

ReGenESyS - REborned GENeric and Expansible SYstem Simulator
v19.114 (DayOfDead19)

Generated by Doxygen 1.8.13

Contents

1	ReGenESyS	1
2	Todo List	1
3	Hierarchical Index	1
3.1	Class Hierarchy	1
4	Class Index	8
4.1	Class List	8
5	File Index	13
5.1	File List	13
6	Class Documentation	22
6.1	Access Class Reference	22
6.1.1	Detailed Description	25
6.1.2	Constructor & Destructor Documentation	25
6.1.3	Member Function Documentation	26
6.2	Allocate Class Reference	29
6.2.1	Detailed Description	32
6.2.2	Constructor & Destructor Documentation	32
6.2.3	Member Function Documentation	33
6.3	AnalysisOfVariance_if Class Reference	36
6.3.1	Detailed Description	37
6.3.2	Member Function Documentation	37
6.4	Assign Class Reference	37
6.4.1	Detailed Description	40
6.4.2	Constructor & Destructor Documentation	40
6.4.3	Member Function Documentation	41
6.5	Assign::Assignment Class Reference	47
6.5.1	Detailed Description	48

6.5.2	Constructor & Destructor Documentation	48
6.5.3	Member Function Documentation	48
6.6	Attribute Class Reference	50
6.6.1	Detailed Description	53
6.6.2	Constructor & Destructor Documentation	53
6.6.3	Member Function Documentation	54
6.7	BaseConsoleGenesysApplication Class Reference	57
6.7.1	Detailed Description	58
6.7.2	Constructor & Destructor Documentation	58
6.7.3	Member Function Documentation	58
6.8	Batch Class Reference	68
6.8.1	Detailed Description	71
6.8.2	Constructor & Destructor Documentation	71
6.8.3	Member Function Documentation	72
6.9	SamplerBoostImpl::BoostImplRNG_Parameters Struct Reference	75
6.9.1	Detailed Description	76
6.9.2	Member Data Documentation	76
6.10	BuildSimulationModel03 Class Reference	77
6.10.1	Detailed Description	78
6.10.2	Constructor & Destructor Documentation	79
6.10.3	Member Function Documentation	79
6.11	CelularAutomata Class Reference	79
6.11.1	Detailed Description	80
6.11.2	Constructor & Destructor Documentation	80
6.12	Collector_if Class Reference	80
6.12.1	Detailed Description	82
6.12.2	Member Function Documentation	82
6.13	CollectorDatafile_if Class Reference	86
6.13.1	Detailed Description	87
6.13.2	Member Function Documentation	87

6.14 CollectorDatafileDefaultImpl1 Class Reference	89
6.14.1 Detailed Description	92
6.14.2 Constructor & Destructor Documentation	92
6.14.3 Member Function Documentation	92
6.15 CollectorDefaultImpl1 Class Reference	95
6.15.1 Detailed Description	96
6.15.2 Constructor & Destructor Documentation	96
6.15.3 Member Function Documentation	97
6.16 ComponentManager Class Reference	98
6.16.1 Detailed Description	99
6.16.2 Constructor & Destructor Documentation	99
6.16.3 Member Function Documentation	100
6.17 ConnectionManager Class Reference	106
6.17.1 Detailed Description	106
6.17.2 Constructor & Destructor Documentation	106
6.17.3 Member Function Documentation	107
6.18 Counter Class Reference	115
6.18.1 Detailed Description	117
6.18.2 Constructor & Destructor Documentation	117
6.18.3 Member Function Documentation	118
6.19 Create Class Reference	123
6.19.1 Detailed Description	126
6.19.2 Constructor & Destructor Documentation	126
6.19.3 Member Function Documentation	127
6.20 Decide Class Reference	133
6.20.1 Detailed Description	135
6.20.2 Constructor & Destructor Documentation	135
6.20.3 Member Function Documentation	136
6.21 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Struct Reference	141
6.21.1 Detailed Description	142

6.21.2 Constructor & Destructor Documentation	142
6.21.3 Member Data Documentation	142
6.22 Delay Class Reference	143
6.22.1 Detailed Description	145
6.22.2 Constructor & Destructor Documentation	145
6.22.3 Member Function Documentation	146
6.23 Dispose Class Reference	154
6.23.1 Detailed Description	156
6.23.2 Constructor & Destructor Documentation	156
6.23.3 Member Function Documentation	157
6.24 DropOff Class Reference	161
6.24.1 Detailed Description	164
6.24.2 Constructor & Destructor Documentation	164
6.24.3 Member Function Documentation	165
6.25 Dummy Class Reference	168
6.25.1 Detailed Description	171
6.25.2 Constructor & Destructor Documentation	171
6.25.3 Member Function Documentation	172
6.26 ElementManager Class Reference	175
6.26.1 Detailed Description	176
6.26.2 Constructor & Destructor Documentation	176
6.26.3 Member Function Documentation	177
6.27 ElementManager_if Class Reference	188
6.27.1 Detailed Description	188
6.27.2 Constructor & Destructor Documentation	188
6.28 Enter Class Reference	189
6.28.1 Detailed Description	192
6.28.2 Constructor & Destructor Documentation	193
6.28.3 Member Function Documentation	193
6.29 Entity Class Reference	198

6.29.1 Detailed Description	200
6.29.2 Constructor & Destructor Documentation	200
6.29.3 Member Function Documentation	201
6.30 EntityGroup Class Reference	210
6.30.1 Detailed Description	213
6.30.2 Constructor & Destructor Documentation	213
6.30.3 Member Function Documentation	214
6.31 EntityType Class Reference	219
6.31.1 Detailed Description	222
6.31.2 Constructor & Destructor Documentation	222
6.31.3 Member Function Documentation	223
6.32 Event Class Reference	228
6.32.1 Detailed Description	229
6.32.2 Constructor & Destructor Documentation	229
6.32.3 Member Function Documentation	231
6.33 Exit Class Reference	234
6.33.1 Detailed Description	236
6.33.2 Constructor & Destructor Documentation	236
6.33.3 Member Function Documentation	237
6.34 ExperimentDesign_if Class Reference	240
6.34.1 Detailed Description	241
6.34.2 Member Function Documentation	241
6.35 ExperimentDesignDefaultImpl1 Class Reference	242
6.35.1 Detailed Description	243
6.35.2 Constructor & Destructor Documentation	243
6.35.3 Member Function Documentation	244
6.36 ExperimentDesignDummyImpl Class Reference	245
6.36.1 Detailed Description	246
6.36.2 Constructor & Destructor Documentation	246
6.36.3 Member Function Documentation	247

6.37	FactorOrInteractionContribution Class Reference	248
6.37.1	Detailed Description	248
6.37.2	Constructor & Destructor Documentation	248
6.37.3	Member Function Documentation	249
6.38	Failure Class Reference	250
6.38.1	Detailed Description	252
6.38.2	Constructor & Destructor Documentation	252
6.38.3	Member Function Documentation	253
6.39	File Class Reference	255
6.39.1	Detailed Description	258
6.39.2	Constructor & Destructor Documentation	258
6.39.3	Member Function Documentation	259
6.40	FirstExampleOfSimulation Class Reference	261
6.40.1	Detailed Description	263
6.40.2	Constructor & Destructor Documentation	264
6.40.3	Member Function Documentation	264
6.41	Fitter_if Class Reference	266
6.41.1	Detailed Description	267
6.41.2	Member Function Documentation	267
6.42	FitterDefaultImpl1 Class Reference	271
6.42.1	Detailed Description	273
6.42.2	Constructor & Destructor Documentation	273
6.42.3	Member Function Documentation	273
6.43	Formula Class Reference	276
6.43.1	Detailed Description	278
6.43.2	Constructor & Destructor Documentation	278
6.43.3	Member Function Documentation	279
6.44	FourthExampleOfSimulation Class Reference	284
6.44.1	Detailed Description	286
6.44.2	Constructor & Destructor Documentation	287

6.44.3 Member Function Documentation	287
6.45 Free Class Reference	289
6.45.1 Detailed Description	291
6.45.2 Constructor & Destructor Documentation	291
6.45.3 Member Function Documentation	292
6.46 FullSimulationOfComplexModel Class Reference	295
6.46.1 Detailed Description	297
6.46.2 Constructor & Destructor Documentation	298
6.46.3 Member Function Documentation	298
6.47 ParserManager::GenerateNewParserResult Struct Reference	300
6.47.1 Detailed Description	300
6.47.2 Member Data Documentation	300
6.48 GenesysApplication_if Class Reference	302
6.48.1 Detailed Description	302
6.48.2 Member Function Documentation	302
6.49 GenesysConsole Class Reference	303
6.49.1 Detailed Description	305
6.49.2 Constructor & Destructor Documentation	306
6.49.3 Member Function Documentation	307
6.50 GenesysGUI Class Reference	316
6.50.1 Detailed Description	317
6.50.2 Constructor & Destructor Documentation	317
6.50.3 Member Function Documentation	318
6.51 GenesysShell_if Class Reference	318
6.51.1 Detailed Description	320
6.51.2 Member Function Documentation	320
6.52 Halt Class Reference	324
6.52.1 Detailed Description	327
6.52.2 Constructor & Destructor Documentation	327
6.52.3 Member Function Documentation	328

6.53 Hold Class Reference	331
6.53.1 Detailed Description	334
6.53.2 Constructor & Destructor Documentation	334
6.53.3 Member Function Documentation	335
6.54 HypothesisTester_if Class Reference	339
6.54.1 Detailed Description	340
6.54.2 Member Enumeration Documentation	340
6.54.3 Member Function Documentation	340
6.55 HypothesisTesterBoostImpl Class Reference	343
6.55.1 Detailed Description	345
6.55.2 Constructor & Destructor Documentation	345
6.55.3 Member Function Documentation	345
6.56 HypothesisTesterDefaultImpl1 Class Reference	347
6.56.1 Detailed Description	350
6.56.2 Constructor & Destructor Documentation	350
6.56.3 Member Function Documentation	350
6.57 Integrator_if Class Reference	352
6.57.1 Detailed Description	354
6.57.2 Member Function Documentation	354
6.58 IntegratorDefaultImpl1 Class Reference	356
6.58.1 Detailed Description	357
6.58.2 Constructor & Destructor Documentation	357
6.58.3 Member Function Documentation	358
6.59 Leave Class Reference	360
6.59.1 Detailed Description	362
6.59.2 Constructor & Destructor Documentation	363
6.59.3 Member Function Documentation	363
6.60 LicenceManager Class Reference	368
6.60.1 Detailed Description	368
6.60.2 Constructor & Destructor Documentation	368

6.60.3 Member Function Documentation	369
6.61 LinkedBy Class Reference	372
6.61.1 Detailed Description	373
6.61.2 Constructor & Destructor Documentation	373
6.61.3 Member Function Documentation	374
6.62 List< T > Class Template Reference	375
6.62.1 Detailed Description	376
6.62.2 Member Typedef Documentation	376
6.62.3 Constructor & Destructor Documentation	376
6.62.4 Member Function Documentation	377
6.63 LSODE Class Reference	389
6.63.1 Detailed Description	392
6.63.2 Constructor & Destructor Documentation	392
6.63.3 Member Function Documentation	393
6.64 MarkovChain Class Reference	399
6.64.1 Detailed Description	401
6.64.2 Constructor & Destructor Documentation	401
6.64.3 Member Function Documentation	402
6.65 Match Class Reference	408
6.65.1 Detailed Description	411
6.65.2 Constructor & Destructor Documentation	411
6.65.3 Member Function Documentation	412
6.66 MathMeth Class Reference	415
6.66.1 Detailed Description	415
6.66.2 Constructor & Destructor Documentation	416
6.67 Model Class Reference	416
6.67.1 Detailed Description	417
6.67.2 Constructor & Destructor Documentation	418
6.67.3 Member Function Documentation	419
6.68 ModelChecker_if Class Reference	441

6.68.1 Detailed Description	442
6.68.2 Member Function Documentation	442
6.69 ModelCheckerDefaultImpl1 Class Reference	443
6.69.1 Detailed Description	445
6.69.2 Constructor & Destructor Documentation	445
6.69.3 Member Function Documentation	446
6.70 ModelComponent Class Reference	452
6.70.1 Detailed Description	454
6.70.2 Constructor & Destructor Documentation	454
6.70.3 Member Function Documentation	455
6.71 ModelElement Class Reference	466
6.71.1 Detailed Description	468
6.71.2 Constructor & Destructor Documentation	468
6.71.3 Member Function Documentation	470
6.71.4 Member Data Documentation	482
6.72 ModellInfo Class Reference	483
6.72.1 Detailed Description	484
6.72.2 Constructor & Destructor Documentation	484
6.72.3 Member Function Documentation	485
6.73 ModelManager Class Reference	500
6.73.1 Detailed Description	501
6.73.2 Constructor & Destructor Documentation	501
6.73.3 Member Function Documentation	501
6.74 ModelPersistence_if Class Reference	507
6.74.1 Detailed Description	508
6.74.2 Member Function Documentation	508
6.75 ModelPersistenceDefaultImpl1 Class Reference	510
6.75.1 Detailed Description	512
6.75.2 Constructor & Destructor Documentation	512
6.75.3 Member Function Documentation	512

6.76 ModelSimulation Class Reference	514
6.76.1 Detailed Description	516
6.76.2 Constructor & Destructor Documentation	516
6.76.3 Member Function Documentation	517
6.77 Move Class Reference	523
6.77.1 Detailed Description	526
6.77.2 Constructor & Destructor Documentation	526
6.77.3 Member Function Documentation	527
6.78 ParserManager::NewParser Struct Reference	531
6.78.1 Detailed Description	531
6.78.2 Member Data Documentation	531
6.79 ODEfunction Class Reference	532
6.79.1 Detailed Description	533
6.79.2 Constructor & Destructor Documentation	533
6.79.3 Member Data Documentation	533
6.80 OLD_ODElement Class Reference	534
6.80.1 Detailed Description	536
6.80.2 Constructor & Destructor Documentation	536
6.80.3 Member Function Documentation	537
6.81 OnEventManager Class Reference	540
6.81.1 Detailed Description	541
6.81.2 Constructor & Destructor Documentation	542
6.81.3 Member Function Documentation	542
6.82 Parser_if Class Reference	549
6.82.1 Detailed Description	550
6.82.2 Member Function Documentation	550
6.83 ParserChangesInformation Class Reference	552
6.83.1 Detailed Description	553
6.83.2 Member Typedef Documentation	553
6.83.3 Constructor & Destructor Documentation	553

6.83.4 Member Function Documentation	554
6.84 ParserDefaultImpl1 Class Reference	556
6.84.1 Detailed Description	557
6.84.2 Constructor & Destructor Documentation	557
6.84.3 Member Function Documentation	558
6.85 ParserManager Class Reference	559
6.85.1 Detailed Description	560
6.85.2 Constructor & Destructor Documentation	560
6.85.3 Member Function Documentation	560
6.86 PickStation Class Reference	561
6.86.1 Detailed Description	563
6.86.2 Constructor & Destructor Documentation	563
6.86.3 Member Function Documentation	564
6.87 PickUp Class Reference	567
6.87.1 Detailed Description	570
6.87.2 Constructor & Destructor Documentation	570
6.87.3 Member Function Documentation	571
6.88 Plugin Class Reference	574
6.88.1 Detailed Description	575
6.88.2 Constructor & Destructor Documentation	575
6.88.3 Member Function Documentation	575
6.89 PluginConnector_if Class Reference	578
6.89.1 Detailed Description	579
6.89.2 Member Function Documentation	579
6.90 PluginConnectorDummyImpl1 Class Reference	580
6.90.1 Detailed Description	582
6.90.2 Constructor & Destructor Documentation	582
6.90.3 Member Function Documentation	583
6.91 PluginInformation Class Reference	585
6.91.1 Detailed Description	586

6.91.2 Constructor & Destructor Documentation	586
6.91.3 Member Function Documentation	587
6.92 PluginManager Class Reference	598
6.92.1 Detailed Description	599
6.92.2 Constructor & Destructor Documentation	599
6.92.3 Member Function Documentation	600
6.93 ProbDistrib Class Reference	605
6.93.1 Detailed Description	606
6.93.2 Member Function Documentation	606
6.94 ProcessAnalyser_if Class Reference	611
6.94.1 Detailed Description	612
6.94.2 Member Function Documentation	612
6.95 ProcessAnalyserDefaultImpl1 Class Reference	614
6.95.1 Detailed Description	615
6.95.2 Constructor & Destructor Documentation	616
6.95.3 Member Function Documentation	616
6.96 Queue Class Reference	618
6.96.1 Detailed Description	621
6.96.2 Member Enumeration Documentation	622
6.96.3 Constructor & Destructor Documentation	623
6.96.4 Member Function Documentation	623
6.97 Record Class Reference	633
6.97.1 Detailed Description	636
6.97.2 Constructor & Destructor Documentation	636
6.97.3 Member Function Documentation	638
6.98 Release Class Reference	643
6.98.1 Detailed Description	646
6.98.2 Constructor & Destructor Documentation	647
6.98.3 Member Function Documentation	647
6.99 Remove Class Reference	655

6.99.1 Detailed Description	658
6.99.2 Constructor & Destructor Documentation	658
6.99.3 Member Function Documentation	659
6.100 Request Class Reference	662
6.100.1 Detailed Description	665
6.100.2 Constructor & Destructor Documentation	665
6.100.3 Member Function Documentation	666
6.101 RequirementTester Class Reference	669
6.101.1 Detailed Description	670
6.101.2 Constructor & Destructor Documentation	670
6.102 Resource Class Reference	671
6.102.1 Detailed Description	674
6.102.2 Member Typedef Documentation	674
6.102.3 Member Enumeration Documentation	674
6.102.4 Constructor & Destructor Documentation	675
6.102.5 Member Function Documentation	676
6.103 Sampler_if::RNG_Parameters Struct Reference	686
6.103.1 Detailed Description	686
6.103.2 Constructor & Destructor Documentation	687
6.104 Route Class Reference	687
6.104.1 Detailed Description	690
6.104.2 Member Enumeration Documentation	690
6.104.3 Constructor & Destructor Documentation	691
6.104.4 Member Function Documentation	691
6.105 Sampler_if Class Reference	698
6.105.1 Detailed Description	700
6.105.2 Member Function Documentation	701
6.106 SamplerBoostImpl Class Reference	705
6.106.1 Detailed Description	707
6.106.2 Constructor & Destructor Documentation	707

6.106.3 Member Function Documentation	707
6.107 SamplerDefaultImpl1 Class Reference	710
6.107.1 Detailed Description	713
6.107.2 Constructor & Destructor Documentation	713
6.107.3 Member Function Documentation	713
6.108 ScenarioExperiment_if Class Reference	720
6.108.1 Detailed Description	720
6.109 Schedule Class Reference	721
6.109.1 Detailed Description	721
6.109.2 Constructor & Destructor Documentation	721
6.110 Search Class Reference	722
6.110.1 Detailed Description	725
6.110.2 Constructor & Destructor Documentation	725
6.110.3 Member Function Documentation	726
6.111 SecondExampleOfSimulation Class Reference	729
6.111.1 Detailed Description	731
6.111.2 Constructor & Destructor Documentation	732
6.111.3 Member Function Documentation	732
6.112 Seize Class Reference	734
6.112.1 Detailed Description	736
6.112.2 Constructor & Destructor Documentation	737
6.112.3 Member Function Documentation	738
6.113 Separate Class Reference	748
6.113.1 Detailed Description	750
6.113.2 Constructor & Destructor Documentation	750
6.113.3 Member Function Documentation	751
6.114 Sequence Class Reference	754
6.114.1 Detailed Description	757
6.114.2 Constructor & Destructor Documentation	757
6.114.3 Member Function Documentation	758

6.115 Sequence::SequenceStep Class Reference	760
6.115.1 Detailed Description	761
6.115.2 Member Data Documentation	762
6.116 Set Class Reference	762
6.116.1 Detailed Description	765
6.116.2 Constructor & Destructor Documentation	765
6.116.3 Member Function Documentation	766
6.117 Signal Class Reference	770
6.117.1 Detailed Description	772
6.117.2 Constructor & Destructor Documentation	772
6.117.3 Member Function Documentation	773
6.118 SimulationControl Class Reference	776
6.118.1 Detailed Description	777
6.118.2 Constructor & Destructor Documentation	778
6.118.3 Member Function Documentation	778
6.119 SimulationEvent Class Reference	779
6.119.1 Detailed Description	779
6.119.2 Constructor & Destructor Documentation	779
6.119.3 Member Function Documentation	780
6.120 SimulationReporter_if Class Reference	781
6.120.1 Detailed Description	782
6.120.2 Member Function Documentation	782
6.121 SimulationReporterDefaultImpl1 Class Reference	783
6.121.1 Detailed Description	785
6.121.2 Constructor & Destructor Documentation	785
6.121.3 Member Function Documentation	785
6.122 SimulationResponse Class Reference	788
6.122.1 Detailed Description	789
6.122.2 Constructor & Destructor Documentation	790
6.122.3 Member Function Documentation	790

6.122.4 Member Data Documentation	792
6.123 SimulationScenario Class Reference	793
6.123.1 Detailed Description	793
6.123.2 Constructor & Destructor Documentation	794
6.123.3 Member Function Documentation	794
6.124 Simulator Class Reference	796
6.124.1 Detailed Description	797
6.124.2 Constructor & Destructor Documentation	797
6.124.3 Member Function Documentation	798
6.125 SinkModelComponent Class Reference	803
6.125.1 Detailed Description	806
6.125.2 Constructor & Destructor Documentation	806
6.125.3 Member Function Documentation	806
6.126 SourceModelComponent Class Reference	809
6.126.1 Detailed Description	811
6.126.2 Constructor & Destructor Documentation	811
6.126.3 Member Function Documentation	812
6.126.4 Member Data Documentation	822
6.127 Start Class Reference	824
6.127.1 Detailed Description	826
6.127.2 Constructor & Destructor Documentation	826
6.127.3 Member Function Documentation	827
6.128 Station Class Reference	830
6.128.1 Detailed Description	833
6.128.2 Constructor & Destructor Documentation	833
6.128.3 Member Function Documentation	834
6.129 Statistics_if Class Reference	840
6.129.1 Detailed Description	842
6.129.2 Member Function Documentation	842
6.130 StatisticsCollector Class Reference	849

6.130.1 Detailed Description	851
6.130.2 Constructor & Destructor Documentation	851
6.130.3 Member Function Documentation	852
6.131 StatisticsDatafile_if Class Reference	857
6.131.1 Detailed Description	858
6.131.2 Member Function Documentation	859
6.132 StatisticsDataFileDummyImpl Class Reference	860
6.132.1 Detailed Description	863
6.132.2 Constructor & Destructor Documentation	863
6.132.3 Member Function Documentation	863
6.133 StatisticsDefaultImpl1 Class Reference	868
6.133.1 Detailed Description	870
6.133.2 Constructor & Destructor Documentation	870
6.133.3 Member Function Documentation	871
6.134 Stop Class Reference	875
6.134.1 Detailed Description	877
6.134.2 Constructor & Destructor Documentation	877
6.134.3 Member Function Documentation	878
6.135 Storage Class Reference	881
6.135.1 Detailed Description	884
6.135.2 Constructor & Destructor Documentation	884
6.135.3 Member Function Documentation	884
6.136 Store Class Reference	887
6.136.1 Detailed Description	890
6.136.2 Constructor & Destructor Documentation	890
6.136.3 Member Function Documentation	891
6.137 Submodel Class Reference	894
6.137.1 Detailed Description	897
6.137.2 Constructor & Destructor Documentation	897
6.137.3 Member Function Documentation	898

6.138 TestEnterLeaveRoute Class Reference	901
6.138.1 Detailed Description	903
6.138.2 Constructor & Destructor Documentation	904
6.138.3 Member Function Documentation	904
6.139 TestInputAnalyserTools Class Reference	906
6.139.1 Detailed Description	907
6.139.2 Constructor & Destructor Documentation	907
6.139.3 Member Function Documentation	907
6.140 TestLSODE Class Reference	909
6.140.1 Detailed Description	910
6.140.2 Constructor & Destructor Documentation	911
6.140.3 Member Function Documentation	911
6.141 TestMarkovChain Class Reference	913
6.141.1 Detailed Description	914
6.141.2 Constructor & Destructor Documentation	915
6.141.3 Member Function Documentation	915
6.142 TestMatricesOfAttributesAndVariables Class Reference	917
6.142.1 Detailed Description	919
6.142.2 Constructor & Destructor Documentation	919
6.142.3 Member Function Documentation	919
6.143 TestODE Class Reference	921
6.143.1 Detailed Description	922
6.143.2 Constructor & Destructor Documentation	923
6.143.3 Member Function Documentation	923
6.144 TestParser Class Reference	924
6.144.1 Detailed Description	925
6.144.2 Constructor & Destructor Documentation	925
6.144.3 Member Function Documentation	925
6.145 TestSimulationControlAndSimulationResponse Class Reference	926
6.145.1 Detailed Description	928

6.145.2 Constructor & Destructor Documentation	928
6.145.3 Member Function Documentation	928
6.146 TestStatistics Class Reference	929
6.146.1 Detailed Description	930
6.146.2 Constructor & Destructor Documentation	931
6.146.3 Member Function Documentation	931
6.147 ThirdExampleOfSimulation Class Reference	932
6.147.1 Detailed Description	933
6.147.2 Constructor & Destructor Documentation	934
6.147.3 Member Function Documentation	934
6.148 ToolManager Class Reference	936
6.148.1 Detailed Description	936
6.148.2 Constructor & Destructor Documentation	936
6.148.3 Member Function Documentation	937
6.149 TraceErrorEvent Class Reference	938
6.149.1 Detailed Description	939
6.149.2 Constructor & Destructor Documentation	939
6.149.3 Member Function Documentation	939
6.150 TraceEvent Class Reference	940
6.150.1 Detailed Description	940
6.150.2 Constructor & Destructor Documentation	941
6.150.3 Member Function Documentation	941
6.151 TraceManager Class Reference	942
6.151.1 Detailed Description	943
6.151.2 Constructor & Destructor Documentation	943
6.151.3 Member Function Documentation	943
6.152 TraceSimulationEvent Class Reference	953
6.152.1 Detailed Description	955
6.152.2 Constructor & Destructor Documentation	955
6.152.3 Member Function Documentation	955

6.153 TraceSimulationProcess Class Reference	956
6.153.1 Detailed Description	957
6.153.2 Constructor & Destructor Documentation	957
6.154 Traits< T > Struct Template Reference	958
6.154.1 Detailed Description	958
6.155 Traits< Collector_if > Struct Template Reference	958
6.155.1 Detailed Description	959
6.155.2 Member Typedef Documentation	959
6.156 Traits< ExperimentDesign_if > Struct Template Reference	959
6.156.1 Detailed Description	959
6.156.2 Member Typedef Documentation	960
6.157 Traits< Fitter_if > Struct Template Reference	960
6.157.1 Detailed Description	960
6.157.2 Member Typedef Documentation	960
6.158 Traits< GenesysApplication_if > Struct Template Reference	961
6.158.1 Detailed Description	961
6.158.2 Member Typedef Documentation	961
6.159 Traits< HypothesisTester_if > Struct Template Reference	962
6.159.1 Detailed Description	962
6.159.2 Member Typedef Documentation	962
6.160 Traits< Integrator_if > Struct Template Reference	963
6.160.1 Detailed Description	963
6.160.2 Member Typedef Documentation	963
6.160.3 Member Data Documentation	964
6.161 Traits< Model > Struct Template Reference	964
6.161.1 Detailed Description	965
6.161.2 Member Data Documentation	965
6.162 Traits< ModelChecker_if > Struct Template Reference	965
6.162.1 Detailed Description	966
6.162.2 Member Typedef Documentation	966

6.163 Traits< ModelComponent > Struct Template Reference	966
6.163.1 Detailed Description	967
6.163.2 Member Typedef Documentation	967
6.164 Traits< ModelPersistence_if > Struct Template Reference	967
6.164.1 Detailed Description	968
6.164.2 Member Typedef Documentation	968
6.165 Traits< Parser_if > Struct Template Reference	968
6.165.1 Detailed Description	969
6.165.2 Member Typedef Documentation	969
6.166 Traits< PluginConnector_if > Struct Template Reference	969
6.166.1 Detailed Description	969
6.166.2 Member Typedef Documentation	970
6.167 Traits< ProcessAnalyser_if > Struct Template Reference	970
6.167.1 Detailed Description	970
6.167.2 Member Typedef Documentation	970
6.168 Traits< Sampler_if > Struct Template Reference	971
6.168.1 Detailed Description	971
6.168.2 Member Typedef Documentation	971
6.169 Traits< SimulationReporter_if > Struct Template Reference	972
6.169.1 Detailed Description	972
6.169.2 Member Typedef Documentation	972
6.170 Traits< Statistics_if > Struct Template Reference	973
6.170.1 Detailed Description	973
6.170.2 Member Typedef Documentation	973
6.170.3 Member Data Documentation	974
6.171 Unstore Class Reference	974
6.171.1 Detailed Description	977
6.171.2 Constructor & Destructor Documentation	977
6.171.3 Member Function Documentation	978
6.172 Util Class Reference	981

6.172.1 Detailed Description	983
6.172.2 Member Typedef Documentation	983
6.172.3 Member Enumeration Documentation	983
6.172.4 Member Function Documentation	985
6.172.5 Member Data Documentation	992
6.173 Variable Class Reference	993
6.173.1 Detailed Description	995
6.173.2 Constructor & Destructor Documentation	996
6.173.3 Member Function Documentation	996
6.174 Waiting Class Reference	1003
6.174.1 Detailed Description	1005
6.174.2 Constructor & Destructor Documentation	1005
6.174.3 Member Function Documentation	1006
6.175 WaitingResource Class Reference	1007
6.175.1 Detailed Description	1009
6.175.2 Constructor & Destructor Documentation	1009
6.175.3 Member Function Documentation	1010
6.176 Write Class Reference	1011
6.176.1 Detailed Description	1013
6.176.2 Member Enumeration Documentation	1013
6.176.3 Constructor & Destructor Documentation	1013
6.176.4 Member Function Documentation	1014
6.177 WriteElement Class Reference	1020
6.177.1 Detailed Description	1021
6.177.2 Constructor & Destructor Documentation	1021
6.177.3 Member Data Documentation	1022

7 File Documentation	1022
7.1 .dep.inc File Reference	1022
7.2 .dep.inc	1022
7.3 Access.cpp File Reference	1022
7.4 Access.cpp	1023
7.5 Access.h File Reference	1023
7.6 Access.h	1024
7.7 Allocate.cpp File Reference	1024
7.8 Allocate.cpp	1024
7.9 Allocate.h File Reference	1025
7.10 Allocate.h	1026
7.11 AnalysisOfVariance_if.h File Reference	1026
7.12 AnalysisOfVariance_if.h	1026
7.13 Assign.cpp File Reference	1027
7.14 Assign.cpp	1027
7.15 Assign.h File Reference	1028
7.16 Assign.h	1028
7.17 Attribute.cpp File Reference	1029
7.18 Attribute.cpp	1030
7.19 Attribute.h File Reference	1030
7.20 Attribute.h	1031
7.21 BaseConsoleGenesysApplication.cpp File Reference	1031
7.22 BaseConsoleGenesysApplication.cpp	1031
7.23 BaseConsoleGenesysApplication.h File Reference	1033
7.24 BaseConsoleGenesysApplication.h	1034
7.25 Batch.cpp File Reference	1034
7.26 Batch.cpp	1035
7.27 Batch.h File Reference	1035
7.28 Batch.h	1036
7.29 BuildSimulationModel03.cpp File Reference	1036

7.29.1 Function Documentation	1037
7.30 BuildSimulationModel03.cpp	1038
7.31 BuildSimulationModel03.h File Reference	1041
7.32 BuildSimulationModel03.h	1041
7.33 CellularAutomata.cpp File Reference	1042
7.34 CellularAutomata.cpp	1042
7.35 CellularAutomata.h File Reference	1042
7.36 CellularAutomata.h	1042
7.37 Collector_if.h File Reference	1042
7.37.1 Typedef Documentation	1043
7.37.2 Function Documentation	1043
7.38 Collector_if.h	1044
7.39 CollectorDatafile_if.h File Reference	1045
7.40 CollectorDatafile_if.h	1045
7.41 CollectorDatafileDefaultImpl1.cpp File Reference	1045
7.42 CollectorDatafileDefaultImpl1.cpp	1045
7.43 CollectorDatafileDefaultImpl1.h File Reference	1046
7.44 CollectorDatafileDefaultImpl1.h	1046
7.45 CollectorDefaultImpl1.cpp File Reference	1047
7.46 CollectorDefaultImpl1.cpp	1047
7.47 CollectorDefaultImpl1.h File Reference	1048
7.48 CollectorDefaultImpl1.h	1048
7.49 ComponentManager.cpp File Reference	1048
7.50 ComponentManager.cpp	1049
7.51 ComponentManager.h File Reference	1050
7.52 ComponentManager.h	1050
7.53 ConnectionManager.cpp File Reference	1050
7.54 ConnectionManager.cpp	1051
7.55 ConnectionManager.h File Reference	1051
7.55.1 Typedef Documentation	1051

7.56 ConnectionManager.h	1052
7.57 Counter.cpp File Reference	1052
7.58 Counter.cpp	1052
7.59 Counter.h File Reference	1053
7.60 Counter.h	1054
7.61 Create.cpp File Reference	1054
7.62 Create.cpp	1055
7.63 Create.h File Reference	1056
7.64 Create.h	1056
7.65 Decide.cpp File Reference	1057
7.66 Decide.cpp	1057
7.67 Decide.h File Reference	1058
7.68 Decide.h	1059
7.69 DefaultTraceAndEventHandlers.h File Reference	1059
7.70 DefaultTraceAndEventHandlers.h	1059
7.71 DefineGetterSetter.h File Reference	1060
7.71.1 Typedef Documentation	1061
7.71.2 Function Documentation	1061
7.72 DefineGetterSetter.h	1064
7.73 Delay.cpp File Reference	1065
7.74 Delay.cpp	1065
7.75 Delay.h File Reference	1067
7.76 Delay.h	1067
7.77 Dispose.cpp File Reference	1068
7.78 Dispose.cpp	1068
7.79 Dispose.h File Reference	1069
7.80 Dispose.h	1069
7.81 DropOff.cpp File Reference	1070
7.82 DropOff.cpp	1070
7.83 DropOff.h File Reference	1071

7.84 DropOff.h	1071
7.85 Dummy.cpp File Reference	1071
7.86 Dummy.cpp	1072
7.87 Dummy.h File Reference	1072
7.88 Dummy.h	1073
7.89 ElementManager.cpp File Reference	1073
7.90 ElementManager.cpp	1073
7.91 ElementManager.h File Reference	1076
7.92 ElementManager.h	1076
7.93 ElementManager_if.h File Reference	1077
7.94 ElementManager_if.h	1077
7.95 Enter.cpp File Reference	1077
7.96 Enter.cpp	1078
7.97 Enter.h File Reference	1079
7.98 Enter.h	1079
7.99 Entity.cpp File Reference	1079
7.100 Entity.cpp	1080
7.101 Entity.h File Reference	1082
7.102 Entity.h	1082
7.103 EntityGroup.cpp File Reference	1083
7.104 EntityGroup.cpp	1083
7.105 EntityGroup.h File Reference	1084
7.106 EntityGroup.h	1085
7.107 EntityType.cpp File Reference	1085
7.108 EntityType.cpp	1085
7.109 EntityType.h File Reference	1088
7.110 EntityType.h	1088
7.111 Event.cpp File Reference	1089
7.112 Event.cpp	1089
7.113 Event.h File Reference	1090

7.114Event.h	1090
7.115Exit.cpp File Reference	1091
7.116Exit.cpp	1091
7.117Exit.h File Reference	1092
7.118Exit.h	1092
7.119ExperimentDesign_if.h File Reference	1093
7.120ExperimentDesign_if.h	1093
7.121ExperimentDesignDefaultImpl1.cpp File Reference	1093
7.122ExperimentDesignDefaultImpl1.cpp	1093
7.123ExperimentDesignDefaultImpl1.h File Reference	1094
7.124ExperimentDesignDefaultImpl1.h	1094
7.125ExperimentDesignDummyImpl.cpp File Reference	1095
7.126ExperimentDesignDummyImpl.cpp	1095
7.127ExperimentDesignDummyImpl.h File Reference	1095
7.128ExperimentDesignDummyImpl.h	1096
7.129FactorOrInteractionContribution.cpp File Reference	1096
7.130FactorOrInteractionContribution.cpp	1096
7.131FactorOrInteractionContribution.h File Reference	1097
7.132FactorOrInteractionContribution.h	1097
7.133Failure.cpp File Reference	1097
7.134Failure.cpp	1098
7.135Failure.h File Reference	1098
7.136Failure.h	1099
7.137File.cpp File Reference	1099
7.138File.cpp	1100
7.139File.h File Reference	1100
7.140File.h	1101
7.141FirstExampleOfSimulation.cpp File Reference	1101
7.142FirstExampleOfSimulation.cpp	1102
7.143FirstExampleOfSimulation.h File Reference	1103

7.144FirstExampleOfSimulation.h	1103
7.145Fitter_if.h File Reference	1103
7.146Fitter_if.h	1103
7.147FitterDefaultImpl1.cpp File Reference	1104
7.148FitterDefaultImpl1.cpp	1104
7.149FitterDefaultImpl1.h File Reference	1105
7.150FitterDefaultImpl1.h	1105
7.151Formula.cpp File Reference	1106
7.152Formula.cpp	1106
7.153Formula.h File Reference	1107
7.154Formula.h	1108
7.155FourthExampleOfSimulation.cpp File Reference	1108
7.156FourthExampleOfSimulation.cpp	1109
7.157FourthExampleOfSimulation.h File Reference	1111
7.158FourthExampleOfSimulation.h	1111
7.159Free.cpp File Reference	1112
7.160Free.cpp	1112
7.161Free.h File Reference	1113
7.162Free.h	1113
7.163FullSimulationOfComplexModel.cpp File Reference	1114
7.164FullSimulationOfComplexModel.cpp	1114
7.165FullSimulationOfComplexModel.h File Reference	1116
7.166FullSimulationOfComplexModel.h	1116
7.167Functor.h File Reference	1116
7.168Functor.h	1116
7.169GenesysApplication_if.h File Reference	1117
7.170GenesysApplication_if.h	1117
7.171GenesysConsole.cpp File Reference	1117
7.172GenesysConsole.cpp	1118
7.173GenesysConsole.h File Reference	1121

7.174GenesysConsole.h	1121
7.175GenesysGUI.cpp File Reference	1122
7.176GenesysGUI.cpp	1122
7.177GenesysGUI.h File Reference	1122
7.178GenesysGUI.h	1123
7.179GenesysShell_if.h File Reference	1123
7.180GenesysShell_if.h	1123
7.181Halt.cpp File Reference	1124
7.182Halt.cpp	1124
7.183Halt.h File Reference	1125
7.184Halt.h	1125
7.185Hold.cpp File Reference	1126
7.186Hold.cpp	1126
7.187Hold.h File Reference	1127
7.188Hold.h	1127
7.189HypothesisTester_if.h File Reference	1128
7.190HypothesisTester_if.h	1128
7.191HypothesisTesterBoostImpl.cpp File Reference	1128
7.192HypothesisTesterBoostImpl.cpp	1129
7.193HypothesisTesterBoostImpl.h File Reference	1129
7.194HypothesisTesterBoostImpl.h	1130
7.195HypothesisTesterDefaultImpl1.cpp File Reference	1130
7.196HypothesisTesterDefaultImpl1.cpp	1130
7.197HypothesisTesterDefaultImpl1.h File Reference	1131
7.198HypothesisTesterDefaultImpl1.h	1131
7.199Integrator_if.h File Reference	1132
7.200Integrator_if.h	1132
7.201IntegratorDefaultImpl1.cpp File Reference	1132
7.202IntegratorDefaultImpl1.cpp	1133
7.203IntegratorDefaultImpl1.h File Reference	1134

7.204IntegratorDefaultImpl1.h	1134
7.205Leave.cpp File Reference	1135
7.206Leave.cpp	1135
7.207Leave.h File Reference	1136
7.208Leave.h	1136
7.209LicenceManager.cpp File Reference	1137
7.210LicenceManager.cpp	1137
7.211LicenceManager.h File Reference	1138
7.212LicenceManager.h	1138
7.213LinkedBy.cpp File Reference	1139
7.214LinkedBy.cpp	1139
7.215LinkedBy.h File Reference	1139
7.216LinkedBy.h	1140
7.217List.h File Reference	1140
7.218List.h	1140
7.219LSODE.cpp File Reference	1143
7.220LSODE.cpp	1143
7.221LSODE.h File Reference	1146
7.222LSODE.h	1146
7.223main.cpp File Reference	1147
7.223.1 Function Documentation	1147
7.224main.cpp	1147
7.225MarkovChain.cpp File Reference	1148
7.226MarkovChain.cpp	1148
7.227MarkovChain.h File Reference	1150
7.228MarkovChain.h	1150
7.229Match.cpp File Reference	1151
7.230Match.cpp	1151
7.231Match.h File Reference	1152
7.232Match.h	1152

7.233MathMeth.cpp File Reference	1152
7.234MathMeth.cpp	1153
7.235MathMeth.h File Reference	1153
7.236MathMeth.h	1153
7.237Model.cpp File Reference	1154
7.237.1 Function Documentation	1154
7.238Model.cpp	1154
7.239Model.h File Reference	1159
7.240Model.h	1159
7.241ModelChecker_if.h File Reference	1160
7.242ModelChecker_if.h	1161
7.243ModelCheckerDefaultImpl1.cpp File Reference	1161
7.244ModelCheckerDefaultImpl1.cpp	1161
7.245ModelCheckerDefaultImpl1.h File Reference	1166
7.246ModelCheckerDefaultImpl1.h	1166
7.247ModelComponent.cpp File Reference	1167
7.248ModelComponent.cpp	1167
7.249ModelComponent.h File Reference	1169
7.250ModelComponent.h	1169
7.251ModelElement.cpp File Reference	1170
7.252ModelElement.cpp	1170
7.253ModelElement.h File Reference	1173
7.254ModelElement.h	1173
7.255ModellInfo.cpp File Reference	1174
7.256ModellInfo.cpp	1174
7.257ModellInfo.h File Reference	1176
7.258ModellInfo.h	1177
7.259ModelManager.cpp File Reference	1177
7.260ModelManager.cpp	1178
7.261ModelManager.h File Reference	1179

7.262ModelManager.h	1179
7.263ModelPersistence_if.h File Reference	1179
7.264ModelPersistence_if.h	1180
7.265ModelPersistenceDefaultImpl1.cpp File Reference	1180
7.266ModelPersistenceDefaultImpl1.cpp	1180
7.267ModelPersistenceDefaultImpl1.h File Reference	1184
7.268ModelPersistenceDefaultImpl1.h	1184
7.269ModelSimulation.cpp File Reference	1185
7.270ModelSimulation.cpp	1185
7.271ModelSimulation.h File Reference	1191
7.272ModelSimulation.h	1191
7.273Move.cpp File Reference	1192
7.274Move.cpp	1192
7.275Move.h File Reference	1193
7.276Move.h	1194
7.277OLD_ODEelement.cpp File Reference	1194
7.278OLD_ODEelement.cpp	1194
7.279OLD_ODEelement.h File Reference	1195
7.280OLD_ODEelement.h	1196
7.281OnEventManager.cpp File Reference	1196
7.282OnEventManager.cpp	1197
7.283OnEventManager.h File Reference	1198
7.283.1 Typedef Documentation	1198
7.284OnEventManager.h	1199
7.285Parser_if.h File Reference	1200
7.286Parser_if.h	1200
7.287ParserChangesInformation.cpp File Reference	1201
7.288ParserChangesInformation.cpp	1201
7.289ParserChangesInformation.h File Reference	1202
7.290ParserChangesInformation.h	1202

7.291 ParserDefaultImpl1.cpp File Reference	1203
7.292 ParserDefaultImpl1.cpp	1203
7.293 ParserDefaultImpl1.h File Reference	1203
7.294 ParserDefaultImpl1.h	1204
7.295 ParserManager.cpp File Reference	1204
7.296 ParserManager.cpp	1204
7.297 ParserManager.h File Reference	1205
7.298 ParserManager.h	1205
7.299 PickStation.cpp File Reference	1205
7.300 PickStation.cpp	1206
7.301 PickStation.h File Reference	1206
7.302 PickStation.h	1207
7.303 PickUp.cpp File Reference	1207
7.304 PickUp.cpp	1207
7.305 PickUp.h File Reference	1208
7.306 PickUp.h	1209
7.307 Plugin.cpp File Reference	1209
7.308 Plugin.cpp	1209
7.309 Plugin.h File Reference	1210
7.310 Plugin.h	1211
7.311 PluginConnector_if.h File Reference	1211
7.312 PluginConnector_if.h	1212
7.313 PluginConnectorDummyImpl1.cpp File Reference	1212
7.314 PluginConnectorDummyImpl1.cpp	1212
7.315 PluginConnectorDummyImpl1.h File Reference	1214
7.316 PluginConnectorDummyImpl1.h	1214
7.317 PluginInformation.cpp File Reference	1215
7.318 PluginInformation.cpp	1215
7.319 PluginInformation.h File Reference	1217
7.319.1 Typedef Documentation	1217

7.320PluginInformation.h	1218
7.321PluginManager.cpp File Reference	1219
7.322PluginManager.cpp	1219
7.323PluginManager.h File Reference	1221
7.324PluginManager.h	1221
7.325ProbDistrib.cpp File Reference	1222
7.326ProbDistrib.cpp	1222
7.327ProbDistrib.h File Reference	1224
7.328ProbDistrib.h	1224
7.329ProcessAnalyser_if.h File Reference	1225
7.330ProcessAnalyser_if.h	1225
7.331ProcessAnalyserDefaultImpl1.cpp File Reference	1225
7.332ProcessAnalyserDefaultImpl1.cpp	1226
7.333ProcessAnalyserDefaultImpl1.h File Reference	1226
7.334ProcessAnalyserDefaultImpl1.h	1227
7.335Queue.cpp File Reference	1227
7.336Queue.cpp	1227
7.337Queue.h File Reference	1229
7.338Queue.h	1230
7.339README.md File Reference	1231
7.340README.md	1231
7.341Record.cpp File Reference	1231
7.342Record.cpp	1232
7.343Record.h File Reference	1233
7.344Record.h	1233
7.345Release.cpp File Reference	1234
7.346Release.cpp	1234
7.347Release.h File Reference	1236
7.348Release.h	1237
7.349Remove.cpp File Reference	1238

7.350Remove.cpp	1238
7.351Remove.h File Reference	1239
7.352Remove.h	1239
7.353Request.cpp File Reference	1239
7.354Request.cpp	1240
7.355Request.h File Reference	1240
7.356Request.h	1241
7.357RequirementTester.cpp File Reference	1241
7.358RequirementTester.cpp	1241
7.359RequirementTester.h File Reference	1242
7.360RequirementTester.h	1242
7.361Resource.cpp File Reference	1242
7.362Resource.cpp	1242
7.363Resource.h File Reference	1245
7.364Resource.h	1245
7.365Route.cpp File Reference	1246
7.366Route.cpp	1246
7.367Route.h File Reference	1249
7.368Route.h	1249
7.369Sampler_if.h File Reference	1250
7.370Sampler_if.h	1250
7.371SamplerBoostImpl.cpp File Reference	1250
7.372SamplerBoostImpl.cpp	1250
7.373SamplerBoostImpl.h File Reference	1251
7.374SamplerBoostImpl.h	1252
7.375SamplerDefaultImpl1.cpp File Reference	1252
7.376SamplerDefaultImpl1.cpp	1252
7.377SamplerDefaultImpl1.h File Reference	1254
7.378SamplerDefaultImpl1.h	1255
7.379ScenarioExperiment_if.h File Reference	1255

7.380 ScenarioExperiment_if.h	1256
7.381 Schedule.cpp File Reference	1256
7.382 Schedule.cpp	1256
7.383 Schedule.h File Reference	1256
7.384 Schedule.h	1257
7.385 Search.cpp File Reference	1257
7.386 Search.cpp	1257
7.387 Search.h File Reference	1258
7.388 Search.h	1258
7.389 SecondExampleOfSimulation.cpp File Reference	1259
7.390 SecondExampleOfSimulation.cpp	1259
7.391 SecondExampleOfSimulation.h File Reference	1260
7.392 SecondExampleOfSimulation.h	1260
7.393 Seize.cpp File Reference	1261
7.394 Seize.cpp	1261
7.395 Seize.h File Reference	1264
7.396 Seize.h	1264
7.397 Separate.cpp File Reference	1266
7.398 Separate.cpp	1266
7.399 Separate.h File Reference	1267
7.400 Separate.h	1267
7.401 Sequence.cpp File Reference	1268
7.402 Sequence.cpp	1268
7.403 Sequence.h File Reference	1269
7.404 Sequence.h	1269
7.405 Set.cpp File Reference	1270
7.406 Set.cpp	1270
7.407 Set.h File Reference	1271
7.408 Set.h	1271
7.409 Signal.cpp File Reference	1272

7.410Signal.cpp	1272
7.411Signal.h File Reference	1273
7.412Signal.h	1273
7.413SimulationControl.cpp File Reference	1274
7.414SimulationControl.cpp	1274
7.415SimulationControl.h File Reference	1274
7.416SimulationControl.h	1275
7.417SimulationReporter_if.h File Reference	1275
7.418SimulationReporter_if.h	1275
7.419SimulationReporterDefaultImpl1.cpp File Reference	1276
7.420SimulationReporterDefaultImpl1.cpp	1276
7.421SimulationReporterDefaultImpl1.h File Reference	1279
7.422SimulationReporterDefaultImpl1.h	1279
7.423SimulationResponse.cpp File Reference	1280
7.424SimulationResponse.cpp	1280
7.425SimulationResponse.h File Reference	1280
7.426SimulationResponse.h	1281
7.427SimulationScenario.cpp File Reference	1281
7.428SimulationScenario.cpp	1281
7.429SimulationScenario.h File Reference	1282
7.430SimulationScenario.h	1282
7.431Simulator.cpp File Reference	1283
7.432Simulator.cpp	1283
7.433Simulator.h File Reference	1284
7.434Simulator.h	1284
7.435SinkModelComponent.cpp File Reference	1285
7.436SinkModelComponent.cpp	1285
7.437SinkModelComponent.h File Reference	1286
7.438SinkModelComponent.h	1286
7.439SourceModelComponent.cpp File Reference	1287

7.440SourceModelComponent.cpp	1287
7.441SourceModelComponent.h File Reference	1289
7.442SourceModelComponent.h	1289
7.443Start.cpp File Reference	1290
7.444Start.cpp	1290
7.445Start.h File Reference	1291
7.446Start.h	1291
7.447Station.cpp File Reference	1292
7.448Station.cpp	1292
7.449Station.h File Reference	1294
7.450Station.h	1294
7.451Statistics_if.h File Reference	1295
7.452Statistics_if.h	1295
7.453StatisticsCollector.cpp File Reference	1296
7.453.1 Typedef Documentation	1296
7.454StatisticsCollector.cpp	1297
7.455StatisticsCollector.h File Reference	1298
7.456StatisticsCollector.h	1298
7.457StatisticsDataFile_if.h File Reference	1299
7.458StatisticsDataFile_if.h	1299
7.459StatisticsDataFileDefaultImpl.cpp File Reference	1300
7.460StatisticsDataFileDefaultImpl.cpp	1300
7.461StatisticsDataFileDefaultImpl.h File Reference	1301
7.462StatisticsDataFileDefaultImpl.h	1301
7.463StatisticsDefaultImpl1.cpp File Reference	1302
7.464StatisticsDefaultImpl1.cpp	1302
7.465StatisticsDefaultImpl1.h File Reference	1304
7.466StatisticsDefaultImpl1.h	1304
7.467Stop.cpp File Reference	1305
7.468Stop.cpp	1305

7.469 Stop.h File Reference	1306
7.470 Stop.h	1306
7.471 Storage.cpp File Reference	1307
7.472 Storage.cpp	1307
7.473 Storage.h File Reference	1308
7.474 Storage.h	1308
7.475 Store.cpp File Reference	1309
7.476 Store.cpp	1309
7.477 Store.h File Reference	1310
7.478 Store.h	1310
7.479 Submodel.cpp File Reference	1311
7.480 Submodel.cpp	1311
7.481 Submodel.h File Reference	1312
7.482 Submodel.h	1312
7.483 TestEnterLeaveRoute.cpp File Reference	1313
7.484 TestEnterLeaveRoute.cpp	1313
7.485 TestEnterLeaveRoute.h File Reference	1315
7.486 TestEnterLeaveRoute.h	1315
7.487 TestFunctions.cpp File Reference	1315
7.487.1 Typedef Documentation	1316
7.487.2 Function Documentation	1316
7.487.3 Variable Documentation	1316
7.488 TestFunctions.cpp	1317
7.489 TestFunctions.h File Reference	1318
7.490 TestFunctions.h	1318
7.491 TestInputAnalyserTools.cpp File Reference	1318
7.491.1 Function Documentation	1318
7.492 TestInputAnalyserTools.cpp	1320
7.493 TestInputAnalyserTools.h File Reference	1321
7.494 TestInputAnalyserTools.h	1321

7.495 TestLSODE.cpp File Reference	1322
7.496 TestLSODE.cpp	1322
7.497 TestLSODE.h File Reference	1323
7.498 TestLSODE.h	1323
7.499 TestMarkovChain.cpp File Reference	1324
7.500 TestMarkovChain.cpp	1324
7.501 TestMarkovChain.h File Reference	1325
7.502 TestMarkovChain.h	1325
7.503 TestMatricesOfAttributesAndVariables.cpp File Reference	1326
7.504 TestMatricesOfAttributesAndVariables.cpp	1326
7.505 TestMatricesOfAttributesAndVariables.h File Reference	1328
7.506 TestMatricesOfAttributesAndVariables.h	1328
7.507 TestParser.cpp File Reference	1328
7.508 TestParser.cpp	1328
7.509 TestParser.h File Reference	1329
7.510 TestParser.h	1329
7.511 TestSimulationControlAndSimulationResponse.cpp File Reference	1330
7.512 TestSimulationControlAndSimulationResponse.cpp	1330
7.513 TestSimulationControlAndSimulationResponse.h File Reference	1330
7.514 TestSimulationControlAndSimulationResponse.h	1331
7.515 TestStatistics.cpp File Reference	1331
7.516 TestStatistics.cpp	1331
7.517 TestStatistics.h File Reference	1332
7.518 TestStatistics.h	1332
7.519 ThirdExampleOfSimulation.cpp File Reference	1332
7.520 ThirdExampleOfSimulation.cpp	1333
7.521 ThirdExampleOfSimulton.h File Reference	1334
7.522 ThirdExampleOfSimulton.h	1334
7.523 ToolManager.cpp File Reference	1334
7.524 ToolManager.cpp	1335

7.525 ToolManager.h File Reference	1335
7.526 ToolManager.h	1335
7.527 TraceManager.cpp File Reference	1336
7.528 TraceManager.cpp	1336
7.529 TraceManager.h File Reference	1338
7.529.1 Typedef Documentation	1338
7.530 TraceManager.h	1340
7.531 Traits.h File Reference	1342
7.532 Traits.h	1343
7.533 Unstore.cpp File Reference	1346
7.534 Unstore.cpp	1346
7.535 Unstore.h File Reference	1347
7.536 Unstore.h	1347
7.537 Util.cpp File Reference	1348
7.538 Util.cpp	1348
7.539 Util.h File Reference	1350
7.539.1 Function Documentation	1350
7.540 Util.h	1353
7.541 Variable.cpp File Reference	1355
7.542 Variable.cpp	1355
7.543 Variable.h File Reference	1356
7.544 Variable.h	1357
7.545 Write.cpp File Reference	1357
7.546 Write.cpp	1358
7.547 Write.h File Reference	1359
7.548 Write.h	1360

1 ReGenESyS

Reborn Generic and Expansible System Simulator

Genesys is a result of teaching and research activities of Professor Dr. Ing Rafael Luiz Cancian. It began in early 2002 as a way to teach students the basics and simulation techniques of systems implemented by other commercial simulation tools, such as Arena. In Genesys development he replicated all the SIMAN language, used by Arena software, and Genesys has become a clone of that tool, including its graphical interface. Genesys allowed the inclusion of new simulation components through dynamic link libraries and also the parallel execution of simulation models in a distributed environment. The development of Genesys continued until 2007, when the professor stopped teaching systems simulation classes. Ten years later the professor starts again to teach systems simulation classes and to carry out scientific research in the area. So in 2018 Genesys is reborn as ReGenesys, with new language and programming techniques, and even more ambitious goals.

Developed by [rlcancian](#)

2 Todo List

Member `PluginConnectorDummyImpl1::check (const std::string dynamicLibraryFilename)`

:To implement

Member `PluginInformation::PluginInformation (std::string pluginTypename, StaticLoaderComponent< Instance componentloader)`

:

Class `TraceSimulationProcess`

: CLASS NOT FULLY IMPLEMENTED (to be implemented for process analyser)

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<code>AnalysisOfVariance_if</code>	36
<code>Assign::Assignment</code>	47
<code>CelularAutomata</code>	79
<code>Collector_if</code>	80
<code>CollectorDatafile_if</code>	86
<code>CollectorDatafileDefaultImpl1</code>	89
<code>CollectorDefaultImpl1</code>	95
<code>ComponentManager</code>	98
<code>ConnectionManager</code>	106
<code>ElementManager</code>	175
<code>ElementManager_if</code>	188

Event	228
ExperimentDesign_if	240
ExperimentDesignDefaultImpl1	242
ExperimentDesignDummyImpl	245
FactorOrInteractionContribution	248
Fitter_if	266
FitterDefaultImpl1	271
ParserManager::GenerateNewParserResult	300
GenesysApplication_if	302
BaseConsoleGenesysApplication	57
BuildSimulationModel03	77
FirstExampleOfSimulation	261
FourthExampleOfSimulation	284
FullSimulationOfComplexModel	295
SecondExampleOfSimulation	729
TestEnterLeaveRoute	901
TestLSODE	909
TestMarkovChain	913
TestMatricesOfAttributesAndVariables	917
TestODE	921
TestSimulationControlAndSimulationResponse	926
ThirdExampleOfSimulation	932
GenesysConsole	303
GenesysGUI	316
GenesysShell_if	318
TestInputAnalyserTools	906
TestParser	924
TestStatistics	929
HypothesisTester_if	339
HypothesisTesterBoostImpl	343
HypothesisTesterDefaultImpl1	347
Integrator_if	352

IntegratorDefaultImpl1	356
LicenceManager	368
LinkedBy	372
List< T >	375
List< Assign::Assignment *>	375
List< Connection *>	375
List< Entity *>	375
List< Event *>	375
List< Model *>	375
List< ModelComponent *>	375
List< ModelElement *>	375
List< ODEfunction *>	375
List< Plugin *>	375
List< ResourceEventHandler >	375
List< ShellCommand *>	375
List< SimulationControl *>	375
List< simulationEventHandler >	375
List< simulationEventHandlerMethod >	375
List< SimulationResponse *>	375
List< StatisticsCollector *>	375
List< std::map< std::string, double > * >	375
List< std::string >	375
List< traceErrorListener >	375
List< traceErrorListenerMethod >	375
List< traceListener >	375
List< traceListenerMethod >	375
List< traceSimulationListener >	375
List< traceSimulationListenerMethod >	375
List< unsigned int >	375
List< Waiting *>	375
List< WriteElement *>	375
MathMeth	415

Model	416
ModelChecker_if	441
ModelCheckerDefaultImpl1	443
ModelElement	466
Attribute	50
Counter	115
Entity	198
EntityGroup	210
EntityType	219
Failure	250
File	255
Formula	276
ModelComponent	452
Access	22
Allocate	29
Assign	37
Batch	68
Decide	133
Delay	143
DropOff	161
Dummy	168
Enter	189
Exit	234
Free	289
Halt	324
Hold	331
Leave	360
LSODE	389
MarkovChain	399
Match	408
Move	523
PickStation	561

PickUp	567
Record	633
Release	643
Remove	655
Request	662
Route	687
Search	722
Seize	734
Separate	748
Signal	770
SinkModelComponent	803
Dispose	154
SourceModelComponent	809
Create	123
Start	824
Stop	875
Store	887
Submodel	894
Unstore	974
Write	1011
OLD_ODElement	534
Queue	618
Resource	671
Sequence	754
Set	762
Station	830
StatisticsCollector	849
Storage	881
Variable	993
ModelInfo	483
ModelManager	500
ModelPersistence_if	507

ModelPersistenceDefaultImpl1	510
ModelSimulation	514
ParserManager::NewParser	531
ODEfunction	532
OnEventManager	540
Parser_if	549
ParserDefaultImpl1	556
ParserChangesInformation	552
ParserManager	559
Plugin	574
PluginConnector_if	578
PluginConnectorDummyImpl1	580
PluginInformation	585
PluginManager	598
ProbDistrib	605
ProcessAnalyser_if	611
ProcessAnalyserDefaultImpl1	614
RequirementTester	669
Sampler_if::RNG_Parameters	686
SamplerBoostImpl::BoostImplRNG_Parameters	75
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters	141
Sampler_if	698
SamplerBoostImpl	705
SamplerDefaultImpl1	710
ScenarioExperiment_if	720
Schedule	721
Sequence::SequenceStep	760
SimulationEvent	779
SimulationReporter_if	781
SimulationReporterDefaultImpl1	783
SimulationResponse	788
SimulationControl	776

SimulationScenario	793
Simulator	796
Statistics_if	840
StatisticsDatafile_if	857
StatisticsDataFileDummyImpl	860
StatisticsDefaultImpl1	868
ToolManager	936
TraceEvent	940
TraceErrorEvent	938
TraceSimulationEvent	953
TraceSimulationProcess	956
TraceManager	942
Traits< T >	958
Traits< Collector_if >	958
Traits< ExperimentDesign_if >	959
Traits< Fitter_if >	960
Traits< GenesysApplication_if >	961
Traits< HypothesisTester_if >	962
Traits< Integrator_if >	963
Traits< Model >	964
Traits< ModelChecker_if >	965
Traits< ModelComponent >	966
Traits< ModelPersistence_if >	967
Traits< Parser_if >	968
Traits< PluginConnector_if >	969
Traits< ProcessAnalyser_if >	970
Traits< Sampler_if >	971
Traits< SimulationReporter_if >	972
Traits< Statistics_if >	973
Util	981
Waiting	1003
WaitingResource	1007

WriteElement	1020
---------------------	-------------

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Access	22
Allocate	29
AnalysisOfVariance_if	36
Assign	37
Assign::Assignment	47
Attribute	50
BaseConsoleGenesysApplication	57
Batch	68
SamplerBoostImpl::BoostImplRNG_Parameters	75
BuildSimulationModel03	77
CelularAutomata	79
Collector_if	80
CollectorDatafile_if	86
CollectorDatafileDefaultImpl1	89
CollectorDefaultImpl1	95
ComponentManager	98
ConnectionManager	106
Counter	115
Create	123
Decide	133
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters	141
Delay	143
Dispose	154
DropOff	161
Dummy	168
ElementManager	175

ElementManager_if	188
Enter	189
Entity	198
EntityGroup	210
EntityType	219
Event	228
Exit	234
ExperimentDesign_if	240
ExperimentDesignDefaultImpl1	242
ExperimentDesignDummyImpl	245
FactorOrInteractionContribution	248
Failure	250
File	255
FirstExampleOfSimulation	261
Fitter_if	266
FitterDefaultImpl1	271
Formula	276
FourthExampleOfSimulation	284
Free	289
FullSimulationOfComplexModel	295
ParserManager::GenerateNewParserResult	300
GenesysApplication_if	302
GenesysConsole	303
GenesysGUI	316
GenesysShell_if	318
Halt	324
Hold	331
HypothesisTester_if	339
HypothesisTesterBoostImpl	343
HypothesisTesterDefaultImpl1	347
Integrator_if	352
IntegratorDefaultImpl1	356

Leave	360
LicenceManager	368
LinkedBy	372
List< T >	375
LSODE	389
MarkovChain	399
Match	408
MathMeth	415
Model	416
ModelChecker_if	441
ModelCheckerDefaultImpl1	443
ModelComponent	452
ModelElement	466
ModellInfo	483
ModelManager	500
ModelPersistence_if	507
ModelPersistenceDefaultImpl1	510
ModelSimulation	514
Move	523
ParserManager::NewParser	531
ODEfunction	532
OLD_ODEelement	534
OnEventManager	540
Parser_if	549
ParserChangesInformation	552
ParserDefaultImpl1	556
ParserManager	559
PickStation	561
PickUp	567
Plugin	574
PluginConnector_if	578
PluginConnectorDummyImpl1	580

PluginInformation	585
PluginManager	598
ProbDistrib	605
ProcessAnalyser_if	611
ProcessAnalyserDefaultImpl1	614
Queue	618
Record	633
Release	643
Remove	655
Request	662
RequirementTester	669
Resource	671
Sampler_if::RNG_Parameters	686
Route	687
Sampler_if	698
SamplerBoostImpl	705
SamplerDefaultImpl1	710
ScenarioExperiment_if	720
Schedule	721
Search	722
SecondExampleOfSimulation	729
Seize	734
Separate	748
Sequence	754
Sequence::SequenceStep	760
Set	762
Signal	770
SimulationControl	776
SimulationEvent	779
SimulationReporter_if	781
SimulationReporterDefaultImpl1	783
SimulationResponse	788

SimulationScenario	793
Simulator	796
SinkModelComponent	803
SourceModelComponent	809
Start	824
Station	830
Statistics_if	840
StatisticsCollector	849
StatisticsDatafile_if	857
StatisticsDataFileDummyImpl	860
StatisticsDefaultImpl1	868
Stop	875
Storage	881
Store	887
Submodel	894
TestEnterLeaveRoute	901
TestInputAnalyserTools	906
TestLSODE	909
TestMarkovChain	913
TestMatricesOfAttributesAndVariables	917
TestODE	921
TestParser	924
TestSimulationControlAndSimulationResponse	926
TestStatistics	929
ThirdExampleOfSimulation	932
ToolManager	936
TraceErrorEvent	938
TraceEvent	940
TraceManager	942
TraceSimulationEvent	953
TraceSimulationProcess	956
Traits< T >	958

Traits< Collector_if >	958
Traits< ExperimentDesign_if >	959
Traits< Fitter_if >	960
Traits< GenesysApplication_if >	961
Traits< HypothesisTester_if >	962
Traits< Integrator_if >	963
Traits< Model >	964
Traits< ModelChecker_if >	965
Traits< ModelComponent >	966
Traits< ModelPersistence_if >	967
Traits< Parser_if >	968
Traits< PluginConnector_if >	969
Traits< ProcessAnalyser_if >	970
Traits< Sampler_if >	971
Traits< SimulationReporter_if >	972
Traits< Statistics_if >	973
Unstore	974
Util	981
Variable	993
Waiting	1003
WaitingResource	1007
Write	1011
WriteElement	1020

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

.dep.inc	1022
Access.cpp	1022
Access.h	1023
Allocate.cpp	1024

Allocate.h	1025
AnalysisOfVariance_if.h	1026
Assign.cpp	1027
Assign.h	1028
Attribute.cpp	1029
Attribute.h	1030
BaseConsoleGenesysApplication.cpp	1031
BaseConsoleGenesysApplication.h	1033
Batch.cpp	1034
Batch.h	1035
BuildSimulationModel03.cpp	1036
BuildSimulationModel03.h	1041
CellularAutomata.cpp	1042
CellularAutomata.h	1042
Collector_if.h	1042
CollectorDatafile_if.h	1045
CollectorDatafileDefaultImpl1.cpp	1045
CollectorDatafileDefaultImpl1.h	1046
CollectorDefaultImpl1.cpp	1047
CollectorDefaultImpl1.h	1048
ComponentManager.cpp	1048
ComponentManager.h	1050
ConnectionManager.cpp	1050
ConnectionManager.h	1051
Counter.cpp	1052
Counter.h	1053
Create.cpp	1054
Create.h	1056
Decide.cpp	1057
Decide.h	1058
DefaultTraceAndEventHandlers.h	1059
DefineGetterSetter.h	1060

Delay.cpp	1065
Delay.h	1067
Dispose.cpp	1068
Dispose.h	1069
DropOff.cpp	1070
DropOff.h	1071
Dummy.cpp	1071
Dummy.h	1072
ElementManager.cpp	1073
ElementManager.h	1076
ElementManager_if.h	1077
Enter.cpp	1077
Enter.h	1079
Entity.cpp	1079
Entity.h	1082
EntityGroup.cpp	1083
EntityGroup.h	1084
EntityType.cpp	1085
EntityType.h	1088
Event.cpp	1089
Event.h	1090
Exit.cpp	1091
Exit.h	1092
ExperimentDesign_if.h	1093
ExperimentDesignDefaultImpl1.cpp	1093
ExperimentDesignDefaultImpl1.h	1094
ExperimentDesignDummyImpl.cpp	1095
ExperimentDesignDummyImpl.h	1095
FactorOrInteractionContribution.cpp	1096
FactorOrInteractionContribution.h	1097
Failure.cpp	1097
Failure.h	1098

File.cpp	1099
File.h	1100
FirstExampleOfSimulation.cpp	1101
FirstExampleOfSimulation.h	1103
Fitter_if.h	1103
FitterDefaultImpl1.cpp	1104
FitterDefaultImpl1.h	1105
Formula.cpp	1106
Formula.h	1107
FourthExampleOfSimulation.cpp	1108
FourthExampleOfSimulation.h	1111
Free.cpp	1112
Free.h	1113
FullSimulationOfComplexModel.cpp	1114
FullSimulationOfComplexModel.h	1116
Functor.h	1116
GenesysApplication_if.h	1117
GenesysConsole.cpp	1117
GenesysConsole.h	1121
GenesysGUI.cpp	1122
GenesysGUI.h	1122
GenesysShell_if.h	1123
Halt.cpp	1124
Halt.h	1125
Hold.cpp	1126
Hold.h	1127
HypothesisTester_if.h	1128
HypothesisTesterBoostImpl.cpp	1128
HypothesisTesterBoostImpl.h	1129
HypothesisTesterDefaultImpl1.cpp	1130
HypothesisTesterDefaultImpl1.h	1131
Integrator_if.h	1132

IntegratorDefaultImpl1.cpp	1132
IntegratorDefaultImpl1.h	1134
Leave.cpp	1135
Leave.h	1136
LicenceManager.cpp	1137
LicenceManager.h	1138
LinkedBy.cpp	1139
LinkedBy.h	1139
List.h	1140
LSODE.cpp	1143
LSODE.h	1146
main.cpp	1147
MarkovChain.cpp	1148
MarkovChain.h	1150
Match.cpp	1151
Match.h	1152
MathMeth.cpp	1152
MathMeth.h	1153
Model.cpp	1154
Model.h	1159
ModelChecker_if.h	1160
ModelCheckerDefaultImpl1.cpp	1161
ModelCheckerDefaultImpl1.h	1166
ModelComponent.cpp	1167
ModelComponent.h	1169
ModelElement.cpp	1170
ModelElement.h	1173
ModellInfo.cpp	1174
ModellInfo.h	1176
ModelManager.cpp	1177
ModelManager.h	1179
ModelPersistence_if.h	1179

ModelPersistenceDefaultImpl1.cpp	1180
ModelPersistenceDefaultImpl1.h	1184
ModelSimulation.cpp	1185
ModelSimulation.h	1191
Move.cpp	1192
Move.h	1193
OLD_ODElement.cpp	1194
OLD_ODElement.h	1195
OnEventManager.cpp	1196
OnEventManager.h	1198
Parser_if.h	1200
ParserChangesInformation.cpp	1201
ParserChangesInformation.h	1202
ParserDefaultImpl1.cpp	1203
ParserDefaultImpl1.h	1203
ParserManager.cpp	1204
ParserManager.h	1205
PickStation.cpp	1205
PickStation.h	1206
PickUp.cpp	1207
PickUp.h	1208
Plugin.cpp	1209
Plugin.h	1210
PluginConnector_if.h	1211
PluginConnectorDummyImpl1.cpp	1212
PluginConnectorDummyImpl1.h	1214
PluginInformation.cpp	1215
PluginInformation.h	1217
PluginManager.cpp	1219
PluginManager.h	1221
ProbDistrib.cpp	1222
ProbDistrib.h	1224

ProcessAnalyser_if.h	1225
ProcessAnalyserDefaultImpl1.cpp	1225
ProcessAnalyserDefaultImpl1.h	1226
Queue.cpp	1227
Queue.h	1229
Record.cpp	1231
Record.h	1233
Release.cpp	1234
Release.h	1236
Remove.cpp	1238
Remove.h	1239
Request.cpp	1239
Request.h	1240
RequirementTester.cpp	1241
RequirementTester.h	1242
Resource.cpp	1242
Resource.h	1245
Route.cpp	1246
Route.h	1249
Sampler_if.h	1250
SamplerBoostImpl.cpp	1250
SamplerBoostImpl.h	1251
SamplerDefaultImpl1.cpp	1252
SamplerDefaultImpl1.h	1254
ScenarioExperiment_if.h	1255
Schedule.cpp	1256
Schedule.h	1256
Search.cpp	1257
Search.h	1258
SecondExampleOfSimulation.cpp	1259
SecondExampleOfSimulation.h	1260
Seize.cpp	1261

Seize.h	1264
Separate.cpp	1266
Separate.h	1267
Sequence.cpp	1268
Sequence.h	1269
Set.cpp	1270
Set.h	1271
Signal.cpp	1272
Signal.h	1273
SimulationControl.cpp	1274
SimulationControl.h	1274
SimulationReporter_if.h	1275
SimulationReporterDefaultImpl1.cpp	1276
SimulationReporterDefaultImpl1.h	1279
SimulationResponse.cpp	1280
SimulationResponse.h	1280
SimulationScenario.cpp	1281
SimulationScenario.h	1282
Simulator.cpp	1283
Simulator.h	1284
SinkModelComponent.cpp	1285
SinkModelComponent.h	1286
SourceModelComponent.cpp	1287
SourceModelComponent.h	1289
Start.cpp	1290
Start.h	1291
Station.cpp	1292
Station.h	1294
Statistics_if.h	1295
StatisticsCollector.cpp	1296
StatisticsCollector.h	1298
StatisticsDataFile_if.h	1299

StatisticsDataFileDefaultImpl.cpp	1300
StatisticsDataFileDefaultImpl.h	1301
StatisticsDefaultImpl1.cpp	1302
StatisticsDefaultImpl1.h	1304
Stop.cpp	1305
Stop.h	1306
Storage.cpp	1307
Storage.h	1308
Store.cpp	1309
Store.h	1310
Submodel.cpp	1311
Submodel.h	1312
TestEnterLeaveRoute.cpp	1313
TestEnterLeaveRoute.h	1315
TestFunctions.cpp	1315
TestFunctions.h	1318
TestInputAnalyserTools.cpp	1318
TestInputAnalyserTools.h	1321
TestLSODE.cpp	1322
TestLSODE.h	1323
TestMarkovChain.cpp	1324
TestMarkovChain.h	1325
TestMatricesOfAttributesAndVariables.cpp	1326
TestMatricesOfAttributesAndVariables.h	1328
TestParser.cpp	1328
TestParser.h	1329
TestSimulationControlAndSimulationResponse.cpp	1330
TestSimulationControlAndSimulationResponse.h	1330
TestStatistics.cpp	1331
TestStatistics.h	1332
ThirdExampleOfSimulation.cpp	1332
ThirdExampleOfSimultion.h	1334

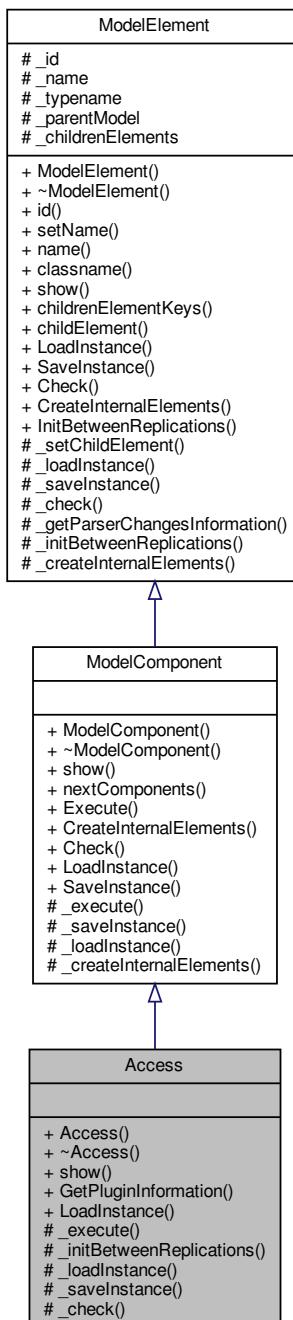
ToolManager.cpp	1334
ToolManager.h	1335
TraceManager.cpp	1336
TraceManager.h	1338
Traits.h	1342
Unstore.cpp	1346
Unstore.h	1347
Util.cpp	1348
Util.h	1350
Variable.cpp	1355
Variable.h	1356
Write.cpp	1357
Write.h	1359

6 Class Documentation

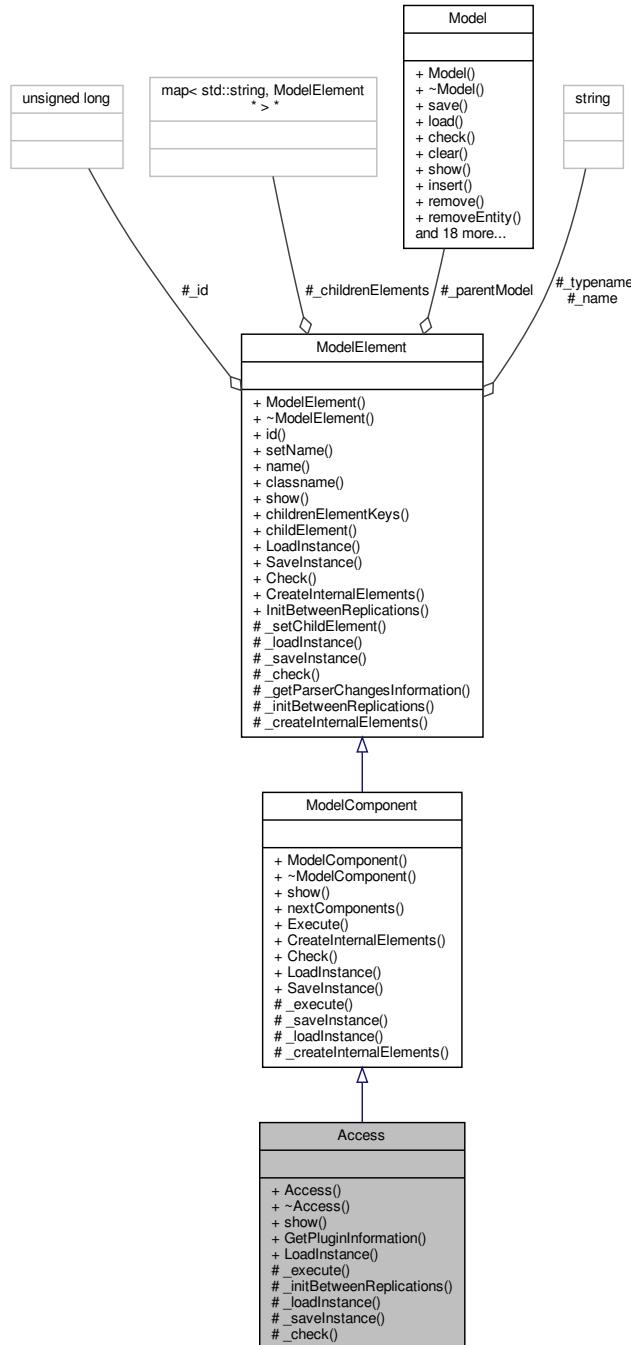
6.1 Access Class Reference

```
#include <Access.h>
```

Inheritance diagram for Access:



Collaboration diagram for Access:



Public Member Functions

- `Access (Model *model, std::string name="")`
- virtual `~Access ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static [PluginInformation * GetPluginInformation \(\)](#)
- static [ModelComponent * LoadInstance \(Model *model, std::map< std::string, std::string > *fields\)](#)

Protected Member Functions

- virtual void [_execute \(Entity *entity\)](#)
- virtual void [_initBetweenReplications \(\)](#)
- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)

Additional Inherited Members

6.1.1 Detailed Description

Access module DESCRIPTION The [Access](#) module allocates one or more cells of a conveyor to an entity for movement from one station to another. Once the entity has control of the cells on the conveyor, it may then be conveyed to the next station. When an entity arrives at an [Access](#) module, it will wait until the appropriate number of contiguous cells on the conveyor are empty and aligned with the entity's station location. TYPICAL USES Parts accessing a conveyor to be sent to a paint booth Glass accessing a conveyor to be transferred to a cutting station PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor that the entity desires.

of Cells Number of contiguous conveyor cells the entity requires for

movement on the conveyor. Queue Type Determines the type of queue used to hold the entities, either an individual Queue, a queue Set, and Internal queue or an Attribute or Expression that evaluate to the queue name. Queue Name Name of the queue that will hold the entity until it accesses the conveyor. Set Name Name of the set of queues. Set Index Defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entries for a queue set containing three members is an expression that evaluates to 1, 2, or 3. Attribute Name Defines the name of the attribute that stores the queue name to which entities will reside. Expression Defines the name of the expression that stores the queue name to which entities will reside.

Definition at line 53 of file [Access.h](#).

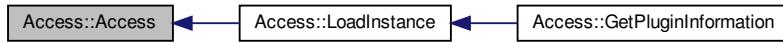
6.1.2 Constructor & Destructor Documentation

6.1.2.1 Access()

```
Access::Access (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Access.cpp](#).

Here is the caller graph for this function:



6.1.2.2 ~Access()

```
virtual Access::~Access ( ) [virtual], [default]
```

6.1.3 Member Function Documentation

6.1.3.1 _check()

```
bool Access::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Access.cpp](#).

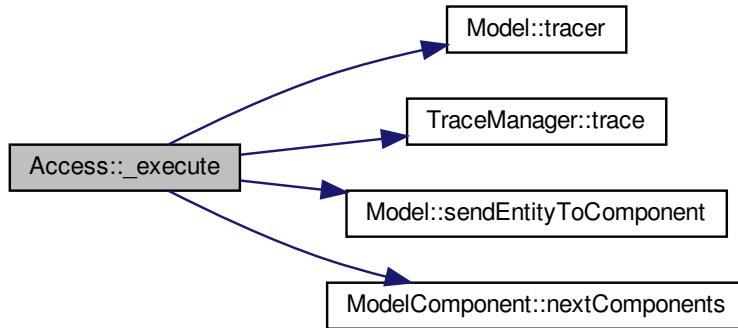
6.1.3.2 _execute()

```
void Access::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Access.cpp](#).

Here is the call graph for this function:



6.1.3.3 `_initBetweenReplications()`

```
void Access::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Access.cpp](#).

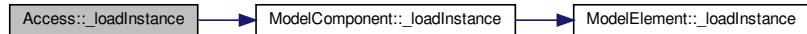
6.1.3.4 `_loadInstance()`

```
bool Access::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

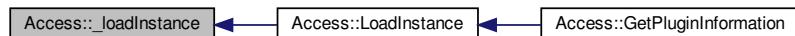
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Access.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



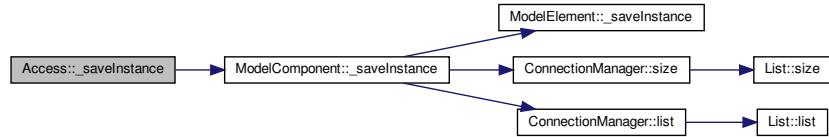
6.1.3.5 `_saveInstance()`

```
std::map< std::string, std::string > * Access::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Access.cpp](#).

Here is the call graph for this function:

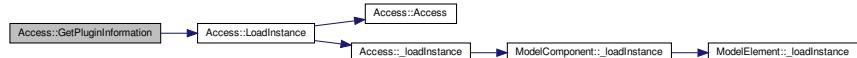


6.1.3.6 `GetPluginInformation()`

```
PluginInformation * Access::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Access.cpp](#).

Here is the call graph for this function:

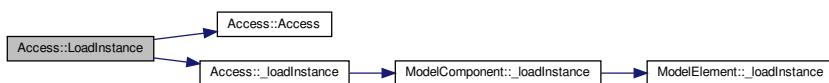


6.1.3.7 `LoadInstance()`

```
ModelComponent * Access::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Access.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



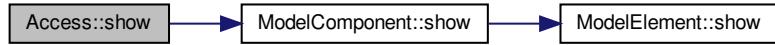
6.1.3.8 show()

```
std::string Access::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [Access.cpp](#).

Here is the call graph for this function:



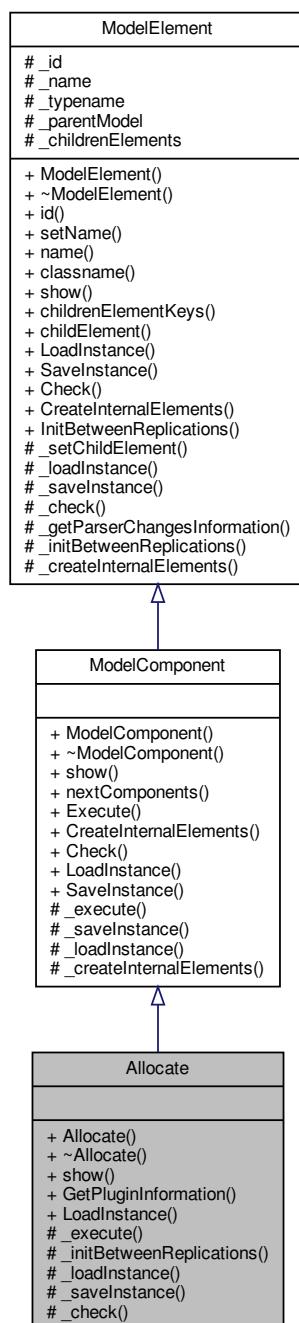
The documentation for this class was generated from the following files:

- [Access.h](#)
- [Access.cpp](#)

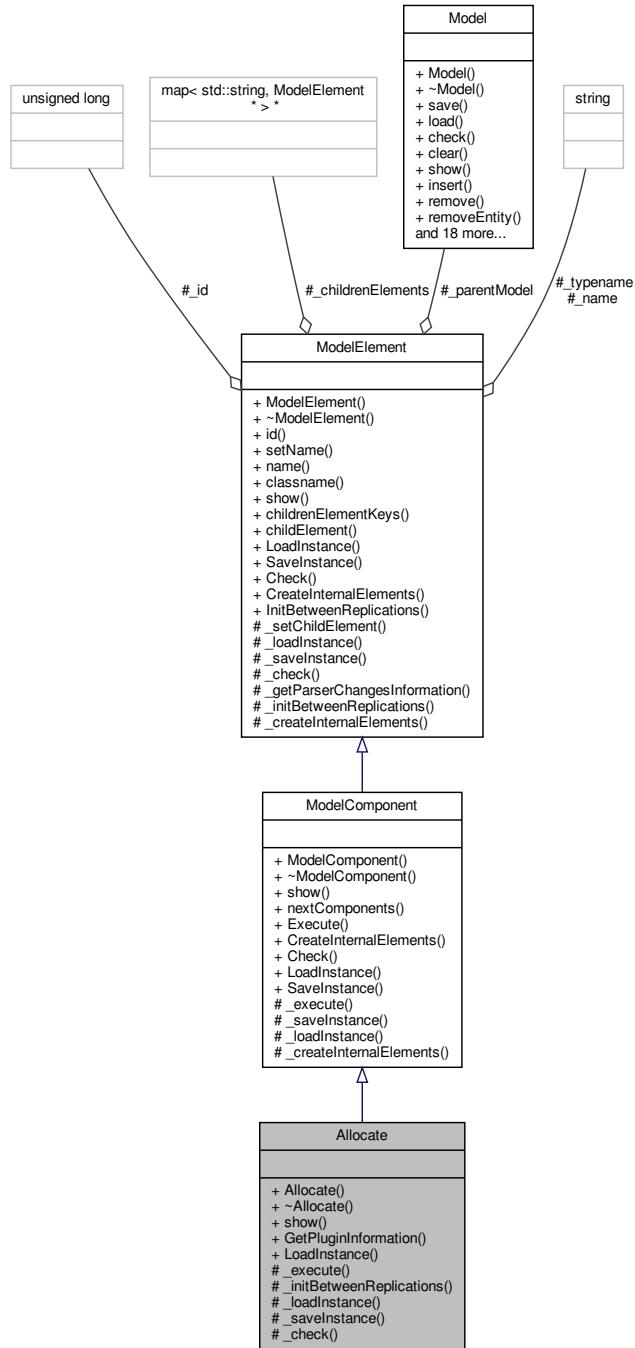
6.2 Allocate Class Reference

```
#include <Allocate.h>
```

Inheritance diagram for Allocate:



Collaboration diagram for Allocate:



Public Member Functions

- `Allocate (Model *model, std::string name="")`
 - `virtual ~Allocate ()=default`
 - `virtual std::string show ()`

Static Public Member Functions

- static [PluginInformation * GetPluginInformation \(\)](#)
- static [ModelComponent * LoadInstance \(Model *model, std::map< std::string, std::string > *fields\)](#)

Protected Member Functions

- virtual void [_execute \(Entity *entity\)](#)
- virtual void [_initBetweenReplications \(\)](#)
- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)

Additional Inherited Members

6.2.1 Detailed Description

Allocate module DESCRIPTION The [Allocate](#) module assigns a transporter to an entity without moving it to the entity's station location. The entity then has control of the transporter to either move it to a particular location or to halt it for a breakdown or failure. A particular transporter unit may be allocated to the entity or a selection rule may be used to determine which of the transporters will be assigned to the entity. TYPICAL USES A mechanic allocates a forklift for scheduled maintenance [Allocate](#) a taxi to pick up waiting passengers PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Transporter Name Name of the transporter to allocate. Selection Rule Rule for determining which of the transporters to allocate to the entity. The selection rule has the following options: Cyclical, Random, Preferred Order, Specific Member, Largest Distance, and Smallest Distance. Save Attribute The attribute name that will store the unit number of the selected transporter. Unit Number Determines the specific transporter in the transporter set to allocate to the entity. It is only visible if selection rule is Specific Member. Priority Priority value of the entity waiting at this module for the transporter specified if one or more entities are waiting for the same transporter anywhere in the model. Entity Location This field is used to determine the transporter closest to or farthest from the requesting entity when using the Largest Distance or Smallest Distance transporter selection rules. For free-path transporters, the entity location must be specified as a station. For guided transporters, the entity location may be specified as a station, intersection, or network link. Queue Type Type of queue used to hold the entities while waiting to allocate the transporter, either an individual Queue, a queue Set, an Internal queue, or an Attribute or Expression that evaluate to the queue name. Queue Name Name of the individual queue. Queue Set Name Name of the queue set that contains the queue being referenced. Set Index The index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entries for a queue set containing three members is an expression that evaluates to 1, 2, or 3. Attribute Name The attribute name that will be evaluated to the queue name. Expression The expression that will be evaluated to the queue name

Definition at line 66 of file [Allocate.h](#).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Allocate()

```
Allocate::Allocate (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Allocate.cpp](#).

Here is the caller graph for this function:



6.2.2.2 ~Allocate()

```
virtual Allocate::~Allocate () [virtual], [default]
```

6.2.3 Member Function Documentation

6.2.3.1 _check()

```
bool Allocate::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Allocate.cpp](#).

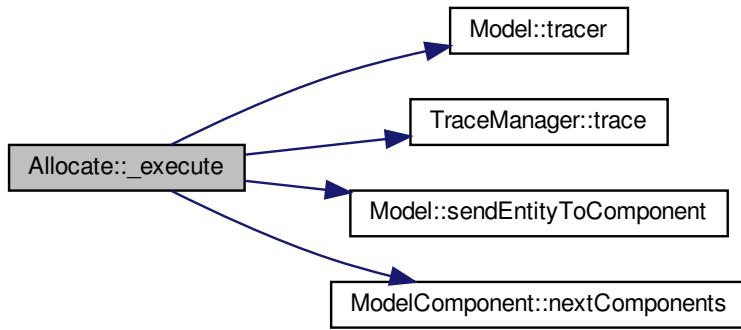
6.2.3.2 _execute()

```
void Allocate::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Allocate.cpp](#).

Here is the call graph for this function:



6.2.3.3 _initBetweenReplications()

```
void Allocate::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Allocate.cpp](#).

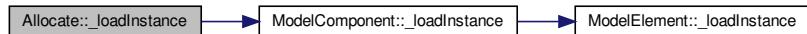
6.2.3.4 _loadInstance()

```
bool Allocate::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Allocate.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



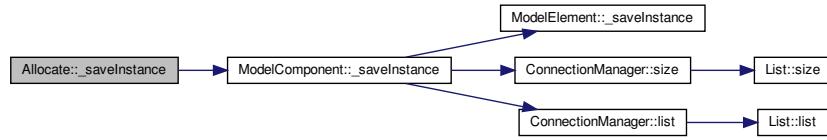
6.2.3.5 `_saveInstance()`

```
std::map< std::string, std::string > * Allocate::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Allocate.cpp](#).

Here is the call graph for this function:

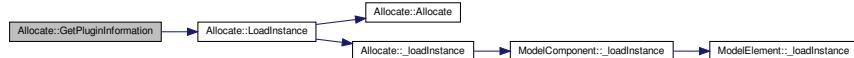


6.2.3.6 `GetPluginInformation()`

```
PluginInformation * Allocate::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Allocate.cpp](#).

Here is the call graph for this function:



6.2.3.7 `LoadInstance()`

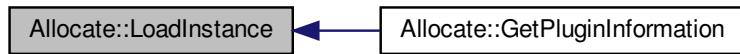
```
ModelComponent * Allocate::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Allocate.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



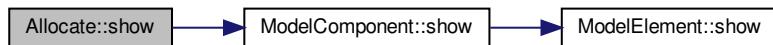
6.2.3.8 show()

```
std::string Allocate::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Allocate.cpp](#).

Here is the call graph for this function:



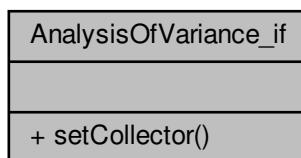
The documentation for this class was generated from the following files:

- [Allocate.h](#)
- [Allocate.cpp](#)

6.3 AnalysisOfVariance_if Class Reference

```
#include <AnalysisOfVariance_if.h>
```

Collaboration diagram for AnalysisOfVariance_if:



Public Member Functions

- virtual void [setCollector \(\)=0](#)

6.3.1 Detailed Description

Definition at line 17 of file [AnalysisOfVariance_if.h](#).

6.3.2 Member Function Documentation

6.3.2.1 setCollector()

```
virtual void AnalysisOfVariance_if::setCollector ( ) [pure virtual]
```

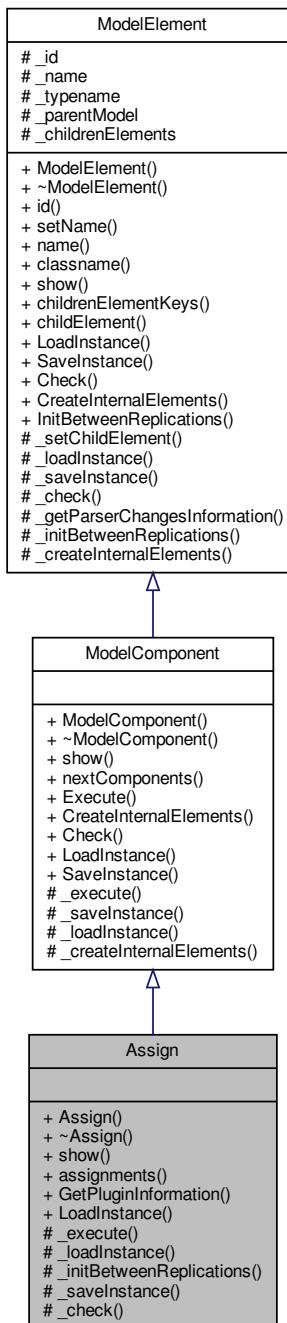
The documentation for this class was generated from the following file:

- [AnalysisOfVariance_if.h](#)

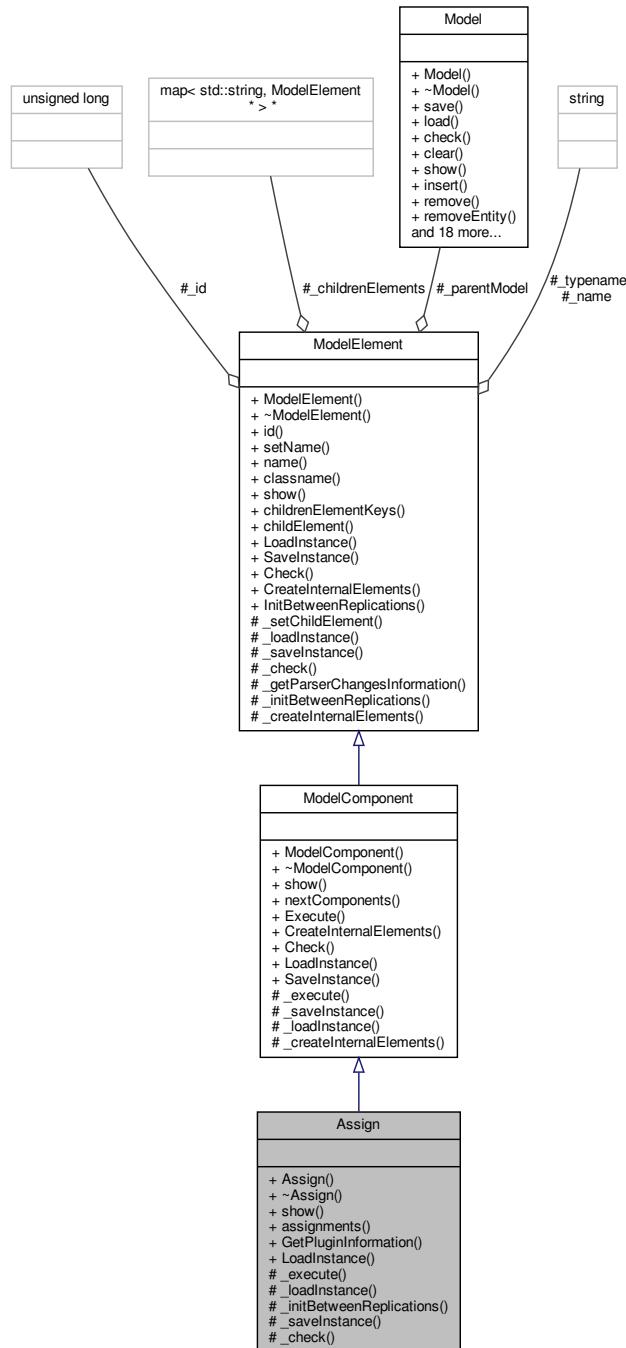
6.4 Assign Class Reference

```
#include <Assign.h>
```

Inheritance diagram for Assign:



Collaboration diagram for Assign:



Classes

- class [Assignment](#)

Public Member Functions

- [Assign \(Model *model, std::string name=""\)](#)

- virtual `~Assign ()=default`
- virtual std::string `show ()`
- `List< Assignment * > * assignments () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.4.1 Detailed Description

Assign module DESCRIPTION This module is used for assigning new values to variables, entity attributes, entity types, entity pictures, or other system variables. Multiple assignments can be made with a single **Assign** module. TYPICAL USES Accumulate the number of subassemblies added to a part Change an entity's type to represent the customer copy of a multi-page form Establish a customer's priority PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Assignments Specifies the one or more assignments that will be made when an entity executes the module. Type Type of assignment to be made. Other can include system variables, such as resource capacity or simulation end time. Variable Name Name of the variable that will be assigned a new value when an entity enters the module. Applies only when Type is **Variable**, **Variable Array** (1D), or **Variable Array** (2D). Row Specifies the row index for a variable array. Column Specifies the column index for a variable array. Attribute Name Name of the entity attribute that will be assigned a new value when the entity enters the module. Applies only when Type is **Attribute**. Entity Type New entity type that will be assigned to the entity when the entity enters the module. Applies only when Type is **Entity** Type. Entity Picture New entity picture that will be assigned to the entity when the entity enters the module. Applies only when Type is **Entity** Picture. Other Identifies the special system variable that will be assigned a new value when an entity enters the module. Applies only when Type is Other. New Value **Assignment** value of the attribute, variable, or other system variable. Does not apply when Type is **Entity** Type or **Entity** Picture.

Definition at line 58 of file [Assign.h](#).

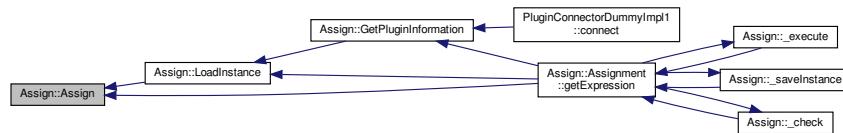
6.4.2 Constructor & Destructor Documentation

6.4.2.1 Assign()

```
Assign::Assign (
    Model * model,
    std::string name = "")
```

Definition at line 21 of file [Assign.cpp](#).

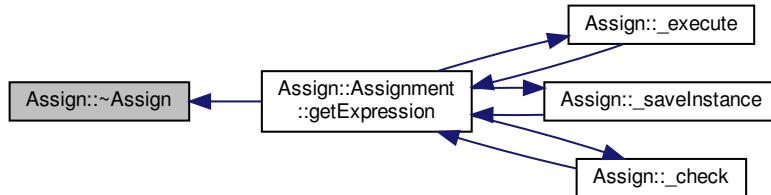
Here is the caller graph for this function:



6.4.2.2 ~Assign()

```
virtual Assign::~Assign () [virtual], [default]
```

Here is the caller graph for this function:



6.4.3 Member Function Documentation

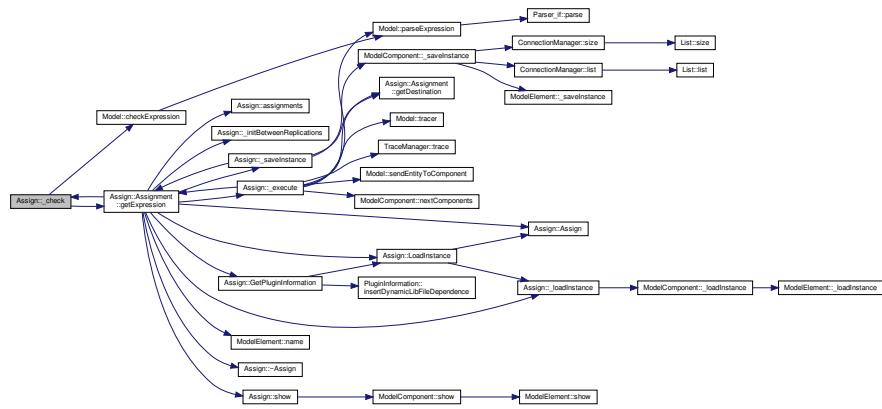
6.4.3.1 _check()

```
bool Assign::_check (
    std::string * errorMessage) [protected], [virtual]
```

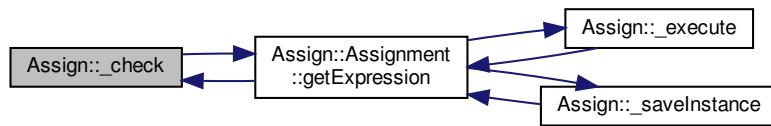
Reimplemented from [ModelElement](#).

Definition at line 96 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



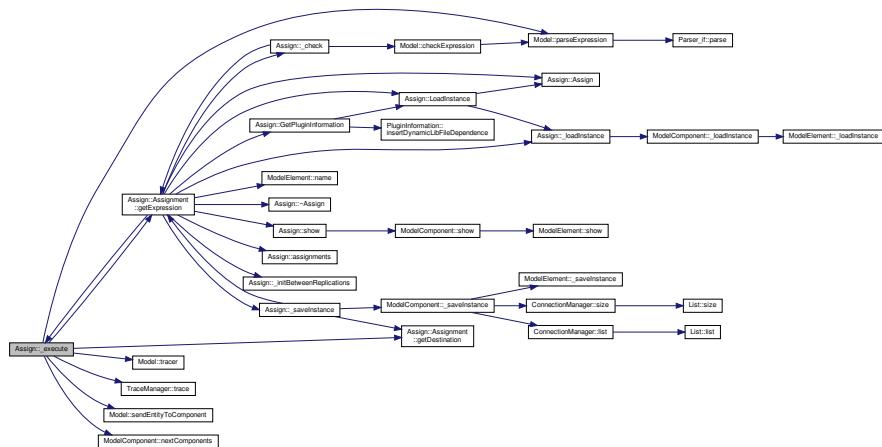
6.4.3.2 `_execute()`

```
void Assign::_execute (
    Entity * entity )  [protected], [virtual]
```

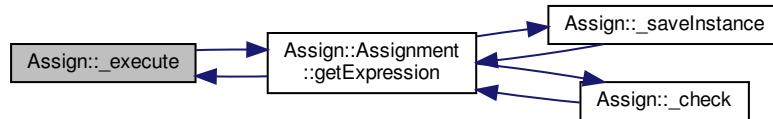
Implements [ModelComponent](#).

Definition at line 50 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



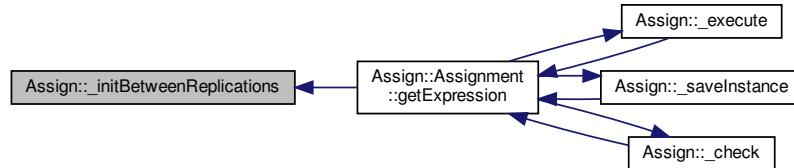
6.4.3.3 _initBetweenReplications()

```
void Assign::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 63 of file [Assign.cpp](#).

Here is the caller graph for this function:



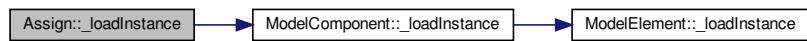
6.4.3.4 _loadInstance()

```
bool Assign::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

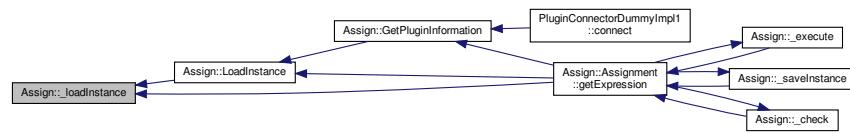
Reimplemented from [ModelComponent](#).

Definition at line 66 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



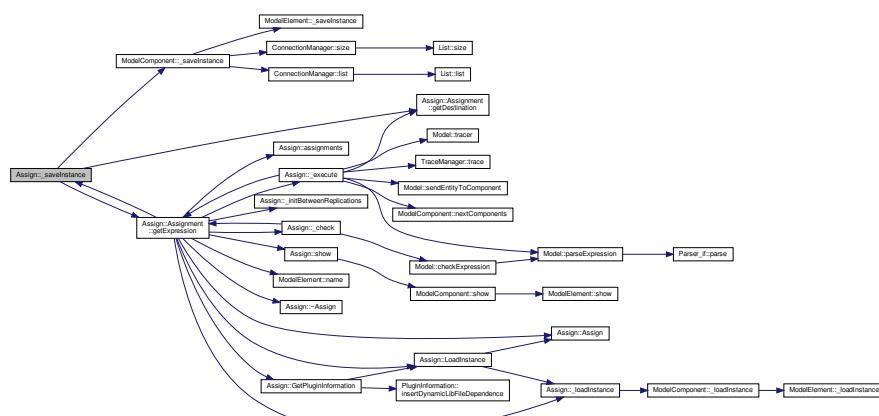
6.4.3.5 `_saveInstance()`

```
std::map< std::string, std::string > * Assign::_saveInstance ( ) [protected], [virtual]
```

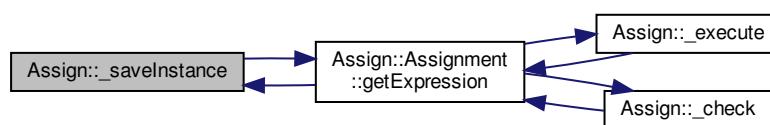
Reimplemented from [ModelComponent](#).

Definition at line 81 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

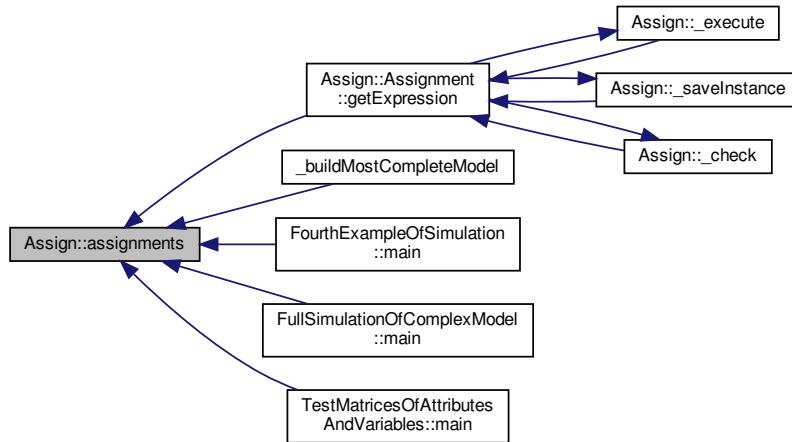


6.4.3.6 assignments()

```
List< Assign::Assignment * > * Assign::assignments ( ) const
```

Definition at line 29 of file [Assign.cpp](#).

Here is the caller graph for this function:

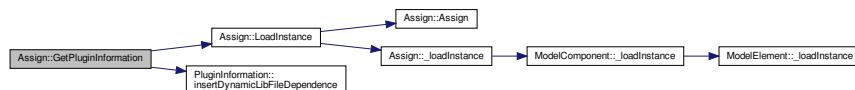


6.4.3.7 GetPluginInformation()

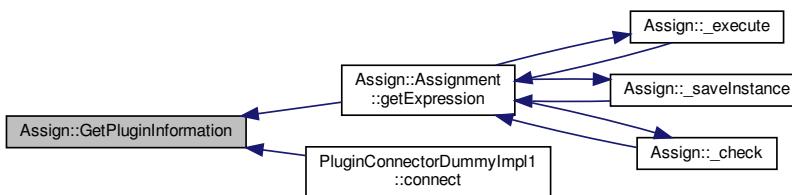
```
PluginInformation * Assign::GetPluginInformation ( ) [static]
```

Definition at line 33 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

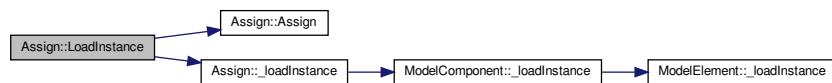


6.4.3.8 LoadInstance()

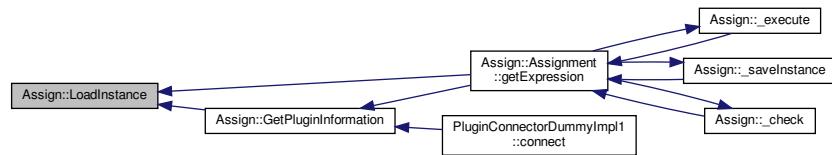
```
ModelComponent * Assign::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 40 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.3.9 show()

```
std::string Assign::show ( ) [virtual]
```

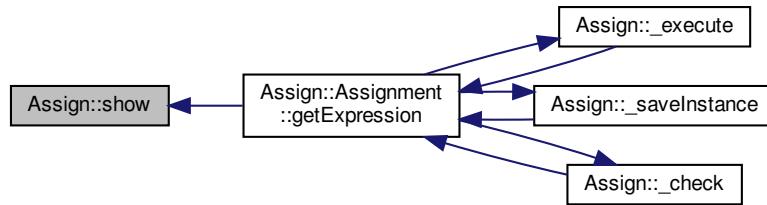
Reimplemented from [ModelComponent](#).

Definition at line 24 of file [Assign.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



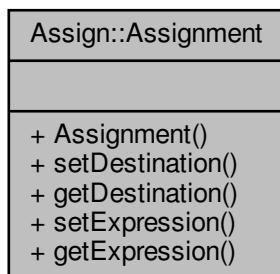
The documentation for this class was generated from the following files:

- [Assign.h](#)
- [Assign.cpp](#)

6.5 Assign::Assignment Class Reference

```
#include <Assign.h>
```

Collaboration diagram for Assign::Assignment:



Public Member Functions

- [Assignment](#) (std::string destination, std::string expression)
- void [setDestination](#) (std::string _destination)
- std::string [getDestination](#) () const
- void [setExpression](#) (std::string _expression)
- std::string [getExpression](#) () const

6.5.1 Detailed Description

While the assign class allows you to perform multiple assignments, the assignment class defines an assignment itself.

Definition at line [63](#) of file [Assign.h](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Assignment()

```
Assign::Assignment::Assignment (
    std::string destination,
    std::string expression ) [inline]
```

Definition at line [65](#) of file [Assign.h](#).

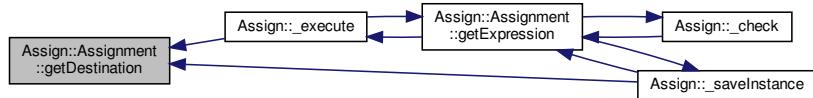
6.5.3 Member Function Documentation

6.5.3.1 getDestination()

```
std::string Assign::Assignment::getDestination ( ) const [inline]
```

Definition at line [75](#) of file [Assign.h](#).

Here is the caller graph for this function:

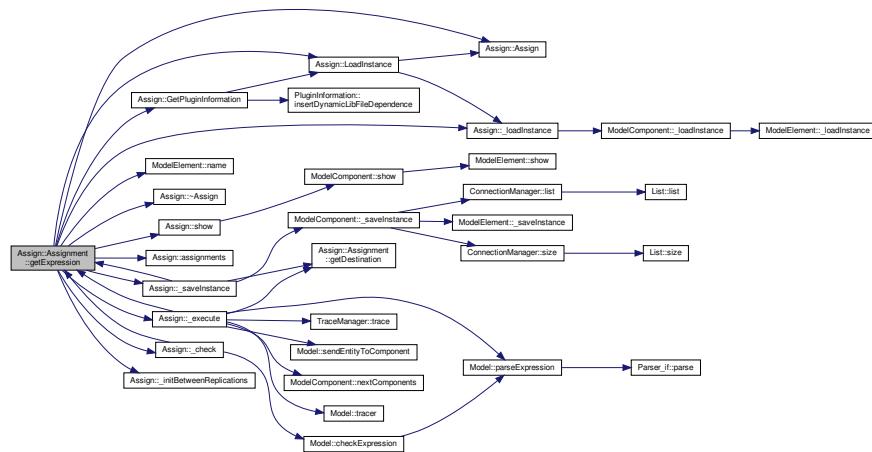


6.5.3.2 getExpression()

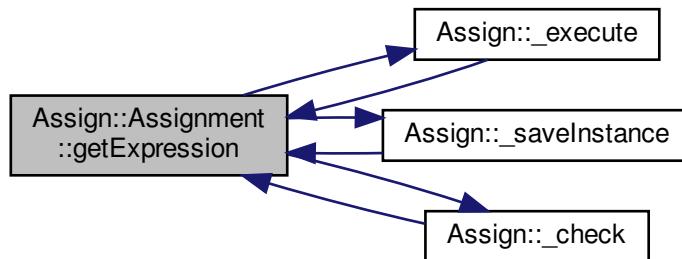
```
std::string Assign::Assignment::getExpression ( ) const [inline]
```

Definition at line 83 of file [Assign.h](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.3 setDestination()

```
void Assign::Assignment::setDestination (
    std::string _destination ) [inline]
```

Definition at line 71 of file [Assign.h](#).

6.5.3.4 setExpression()

```
void Assign::Assignment::setExpression (
    std::string _expression ) [inline]
```

Definition at line 79 of file [Assign.h](#).

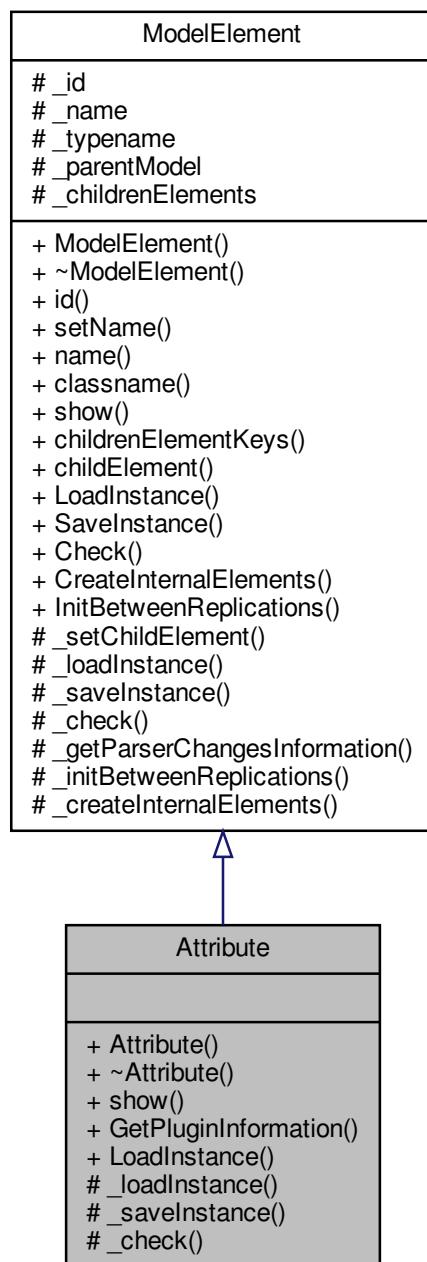
The documentation for this class was generated from the following file:

- [Assign.h](#)

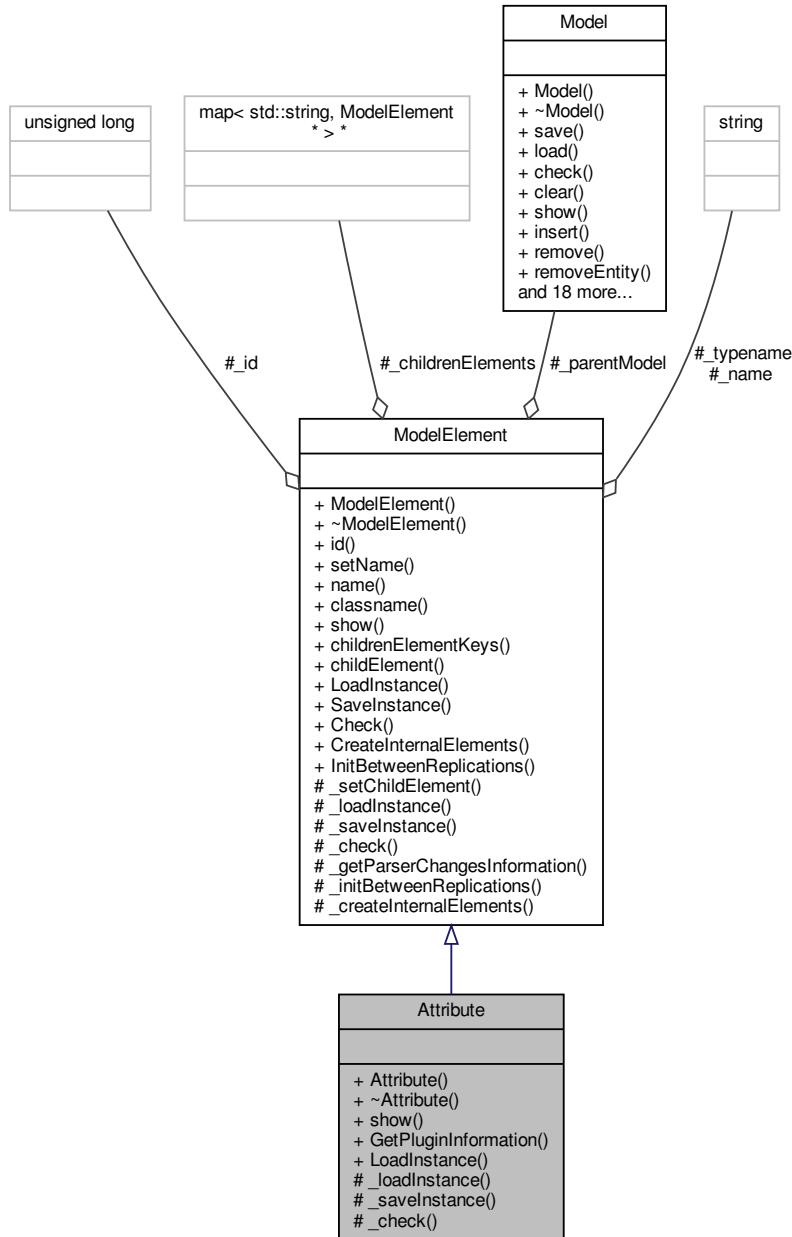
6.6 Attribute Class Reference

```
#include <Attribute.h>
```

Inheritance diagram for Attribute:



Collaboration diagram for Attribute:



Public Member Functions

- **Attribute** (**Model** *model, std::string **name**=“”)
- virtual **~Attribute** ()=default
- virtual std::string **show** ()

Static Public Member Functions

- static **PluginInformation** * **GetPluginInformation** ()
- static **ModelElement** * **LoadInstance** (**Model** *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)

Additional Inherited Members

6.6.1 Detailed Description

Attribute module DESCRIPTION This data module is used to define an attribute's dimension, data type and initial value(s). An attribute is a characteristic of all entities created, but with a specific value that can differ from one entity to another. Attributes can be referenced in other modules (for example, the [Decide](#) module), can be reassigned a new value with the [Assign](#) module, and can be used in any expression. **Attribute** values are unique for each entity, as compared to Variables which are global to the simulation module. There are three methods for manually editing the Initial Values of an **Attribute** module: Using the standard spreadsheet interface. In the module spreadsheet, rightclick on the Initial Values cell and select the Edit via spreadsheet menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the module dialog box. In the module spreadsheet, right-click on any cell and select the Edit via dialog menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the two-dimensional (2-D) spreadsheet interface. In the module spreadsheet, click on the Initial Values cell. TYPICAL USES Due date of an order (entity) Priority of an order (entity) Color of a part (entity) PROMPTS Prompt Description Name The unique name of the attribute being defined. Rows Number of rows in a one- or two-dimensional attribute. Columns Number of columns in a two-dimensional attribute. Data Type The data type of the values stored in the attribute. Valid types are Real and String. The default type is Real. Initial Values Lists the initial value or values of the attribute. You can assign new values to the attribute by using the [Assign](#) module. Initial Value Entity attribute value when entity is created and enters the system.

Definition at line [61](#) of file [Attribute.h](#).

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Attribute()

```
Attribute::Attribute (
    Model * model,
    std::string name = "" )
```

Definition at line [19](#) of file [Attribute.cpp](#).

Here is the caller graph for this function:



6.6.2.2 ~Attribute()

```
virtual Attribute::~Attribute ( ) [virtual], [default]
```

6.6.3 Member Function Documentation

6.6.3.1 _check()

```
bool Attribute::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [52](#) of file [Attribute.cpp](#).

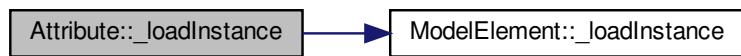
6.6.3.2 _loadInstance()

```
bool Attribute::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [27](#) of file [Attribute.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



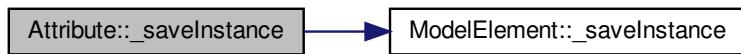
6.6.3.3 `_saveInstance()`

```
std::map< std::string, std::string > * Attribute::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 47 of file [Attribute.cpp](#).

Here is the call graph for this function:

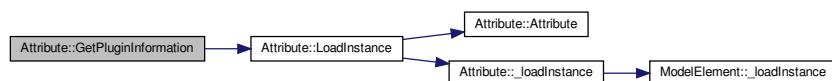


6.6.3.4 `GetPluginInformation()`

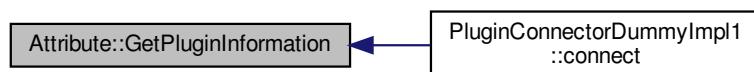
```
PluginInformation * Attribute::GetPluginInformation ( ) [static]
```

Definition at line 31 of file [Attribute.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

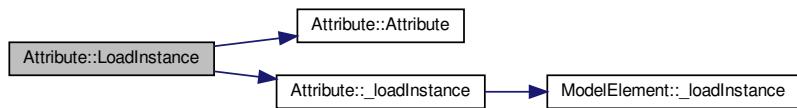


6.6.3.5 LoadInstance()

```
ModelElement * Attribute::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 37 of file [Attribute.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.3.6 show()

```
std::string Attribute::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 23 of file [Attribute.cpp](#).

Here is the call graph for this function:



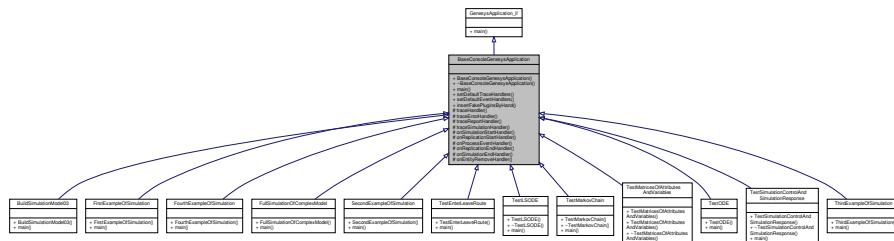
The documentation for this class was generated from the following files:

- [Attribute.h](#)
- [Attribute.cpp](#)

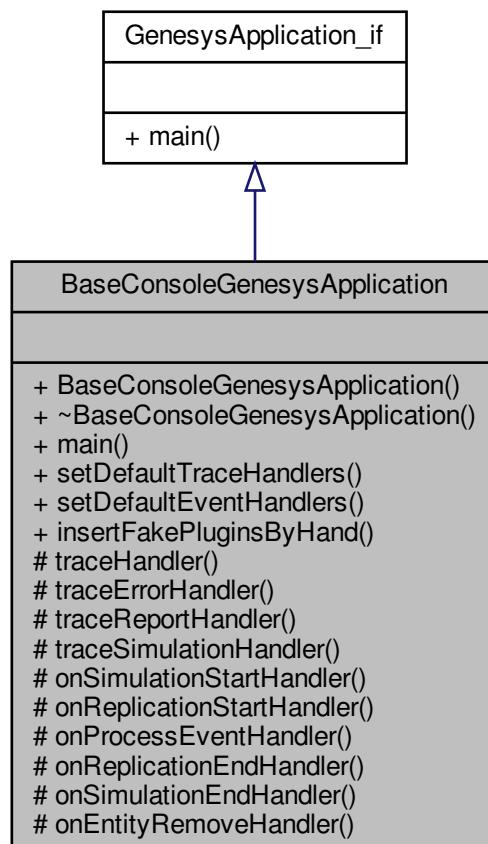
6.7 BaseConsoleGenesysApplication Class Reference

```
#include <BaseConsoleGenesysApplication.h>
```

Inheritance diagram for BaseConsoleGenesysApplication:



Collaboration diagram for BaseConsoleGenesysApplication:



Public Member Functions

- [BaseConsoleGenesysApplication \(\)](#)

- virtual `~BaseConsoleGenesysApplication ()=default`
- virtual int `main (int argc, char **argv)=0`
- void `setDefaultTraceHandlers (TraceManager *tm)`
- void `setDefaultEventHandlers (OnEventManager *oem)`
- void `insertFakePluginsByHand (Simulator *simulator)`

Protected Member Functions

- virtual void `traceHandler (TraceEvent e)`
- virtual void `traceErrorHandler (TraceErrorEvent e)`
- virtual void `traceReportHandler (TraceEvent e)`
- virtual void `traceSimulationHandler (TraceSimulationEvent e)`
- virtual void `onSimulationStartHandler (SimulationEvent *re)`
- virtual void `onReplicationStartHandler (SimulationEvent *re)`
- virtual void `onProcessEventHandler (SimulationEvent *re)`
- virtual void `onReplicationEndHandler (SimulationEvent *re)`
- virtual void `onSimulationEndHandler (SimulationEvent *re)`
- virtual void `onEntityRemoveHandler (SimulationEvent *re)`

6.7.1 Detailed Description

Definition at line 21 of file [BaseConsoleGenesysApplication.h](#).

6.7.2 Constructor & Destructor Documentation

6.7.2.1 BaseConsoleGenesysApplication()

```
BaseConsoleGenesysApplication::BaseConsoleGenesysApplication ( )
```

Definition at line 22 of file [BaseConsoleGenesysApplication.cpp](#).

6.7.2.2 ~BaseConsoleGenesysApplication()

```
virtual BaseConsoleGenesysApplication::~BaseConsoleGenesysApplication ( ) [virtual], [default]
```

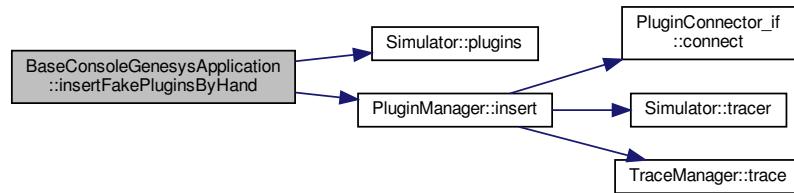
6.7.3 Member Function Documentation

6.7.3.1 insertFakePluginsByHand()

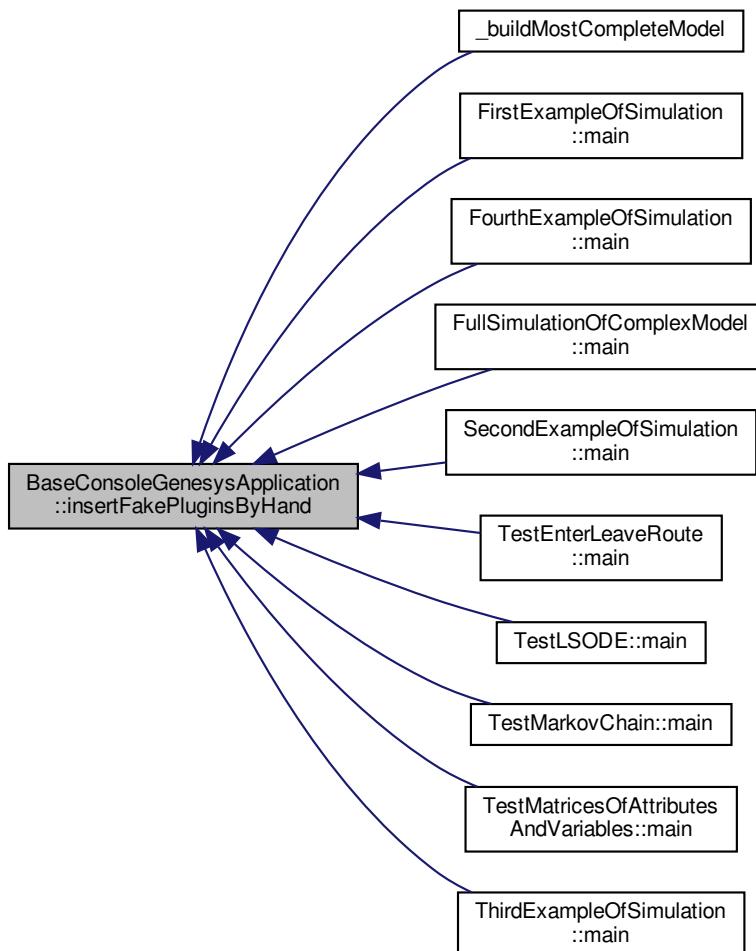
```
void BaseConsoleGenesysApplication::insertFakePluginsByHand (
    Simulator * simulator )
```

Definition at line 87 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.7.3.2 main()

```
virtual int BaseConsoleGenesysApplication::main (
    int argc,
    char ** argv ) [pure virtual]
```

Implements [GenesysApplication_if](#).

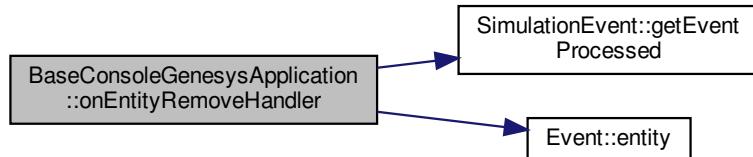
Implemented in [TestMatricesOfAttributesAndVariables](#), [BuildSimulationModel03](#), [TestLSODE](#), [FullSimulation<OfComplexModel](#), [TestMarkovChain](#), [TestSimulationControlAndSimulationResponse](#), [FirstExampleOfSimulation](#), [FourthExampleOfSimulation](#), [SecondExampleOfSimulation](#), [TestEnterLeaveRoute](#), [TestODE](#), and [ThirdExample<OfSimulation](#).

6.7.3.3 onEntityRemoveHandler()

```
void BaseConsoleGenesysApplication::onEntityRemoveHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 67 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:

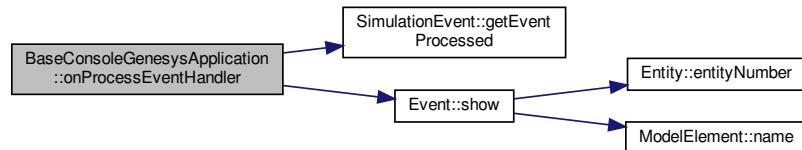


6.7.3.4 onProcessEventHandler()

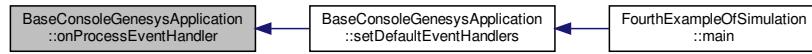
```
void BaseConsoleGenesysApplication::onProcessEventHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 55 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

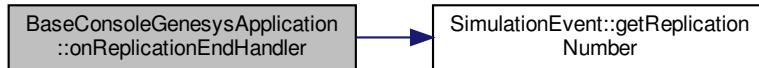


6.7.3.5 onReplicationEndHandler()

```
void BaseConsoleGenesysApplication::onReplicationEndHandler ( SimulationEvent * re ) [protected], [virtual]
```

Definition at line 59 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:

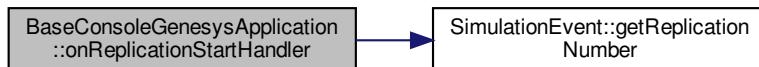


6.7.3.6 onReplicationStartHandler()

```
void BaseConsoleGenesysApplication::onReplicationStartHandler ( SimulationEvent * re ) [protected], [virtual]
```

Definition at line 51 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:

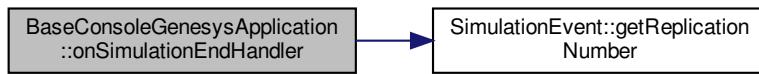


6.7.3.7 onSimulationEndHandler()

```
void BaseConsoleGenesysApplication::onSimulationEndHandler (
    SimulationEvent * re ) [protected], [virtual]
```

Definition at line 63 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



6.7.3.8 onSimulationStartHandler()

```
void BaseConsoleGenesysApplication::onSimulationStartHandler (
    SimulationEvent * re ) [protected], [virtual]
```

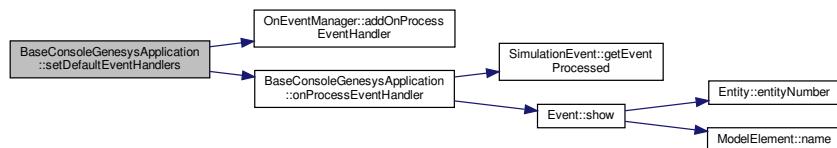
Definition at line 47 of file [BaseConsoleGenesysApplication.cpp](#).

6.7.3.9 setDefaultEventHandlers()

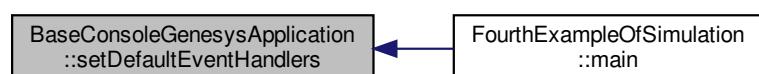
```
void BaseConsoleGenesysApplication::setDefaultEventHandlers (
    OnEventManager * oem )
```

Definition at line 71 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

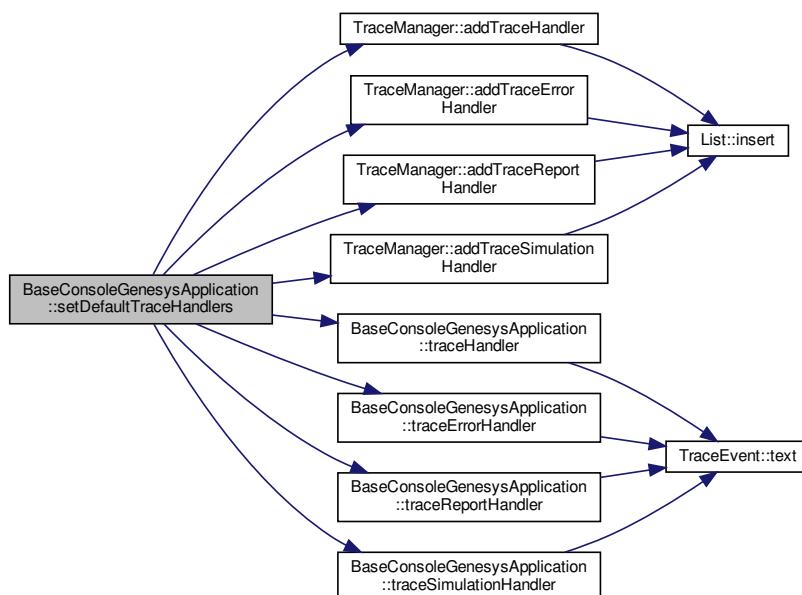


6.7.3.10 setDefaultTraceHandlers()

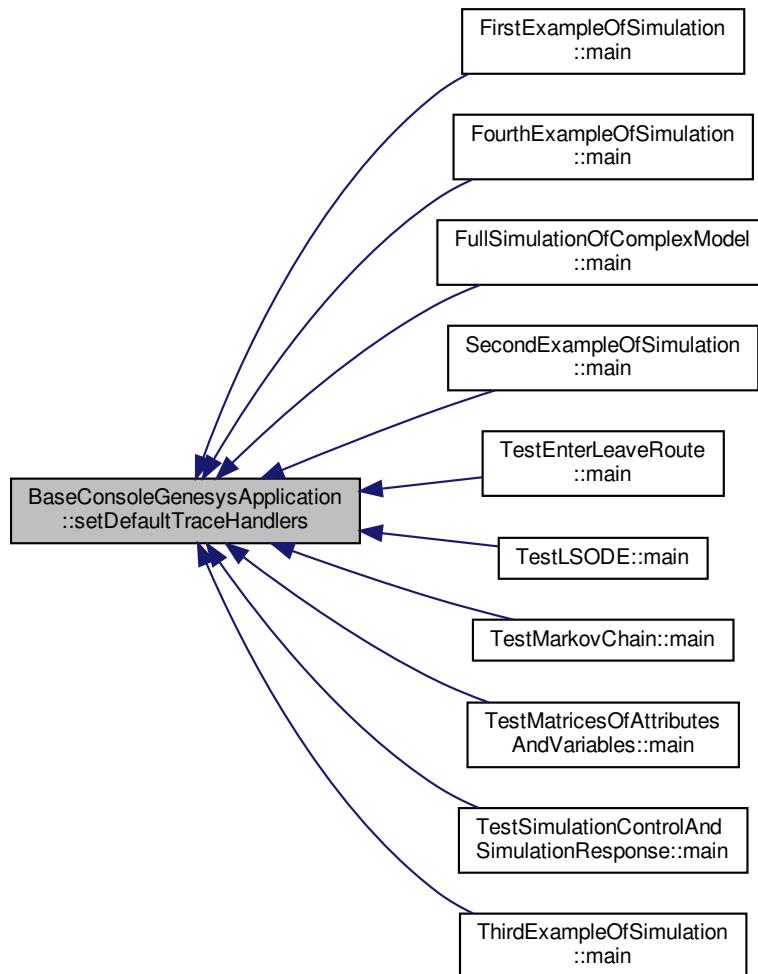
```
void BaseConsoleGenesysApplication::setDefaultTraceHandlers (   
    TraceManager * tm )
```

Definition at line 80 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

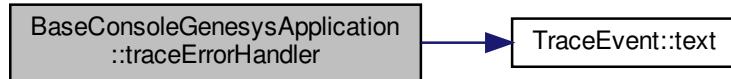


6.7.3.11 traceErrorHandler()

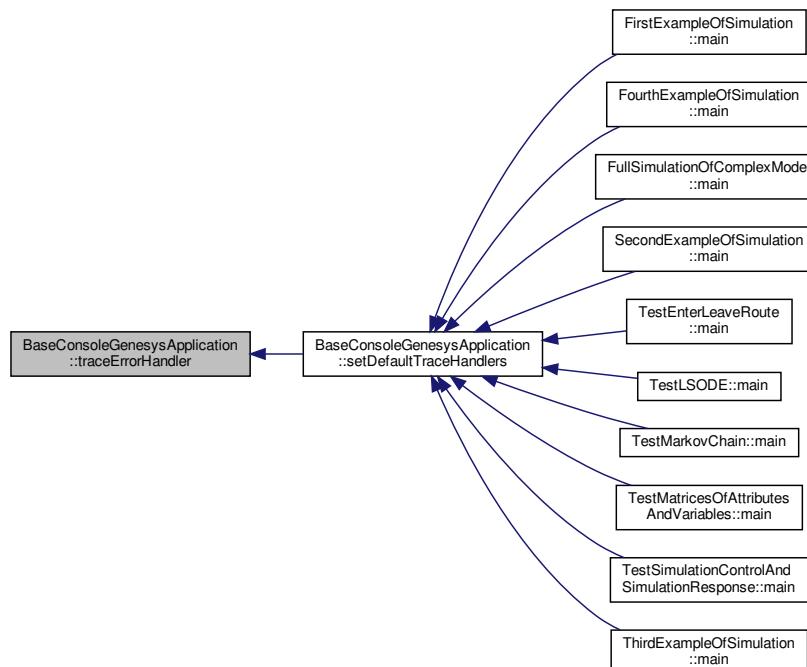
```
void BaseConsoleGenesysApplication::traceErrorHandler (
    TraceErrorEvent e )  [protected], [virtual]
```

Definition at line 32 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

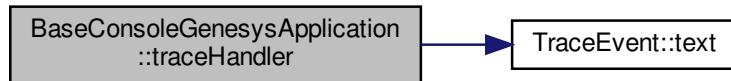


6.7.3.12 traceHandler()

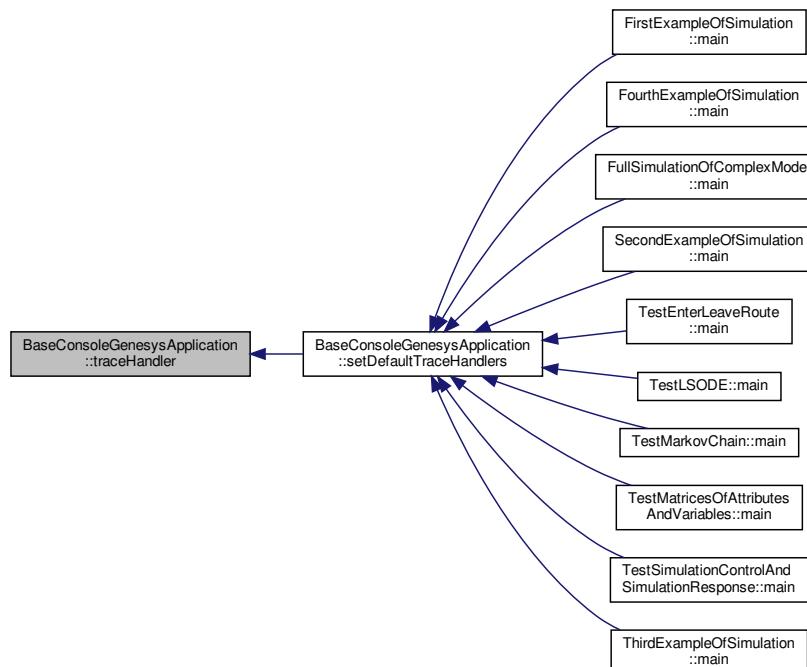
```
void BaseConsoleGenesysApplication::traceHandler (
    TraceEvent e ) [protected], [virtual]
```

Definition at line 28 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



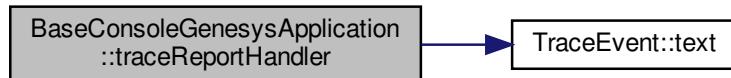
6.7.3.13 traceReportHandler()

```
void BaseConsoleGenesysApplication::traceReportHandler (
```

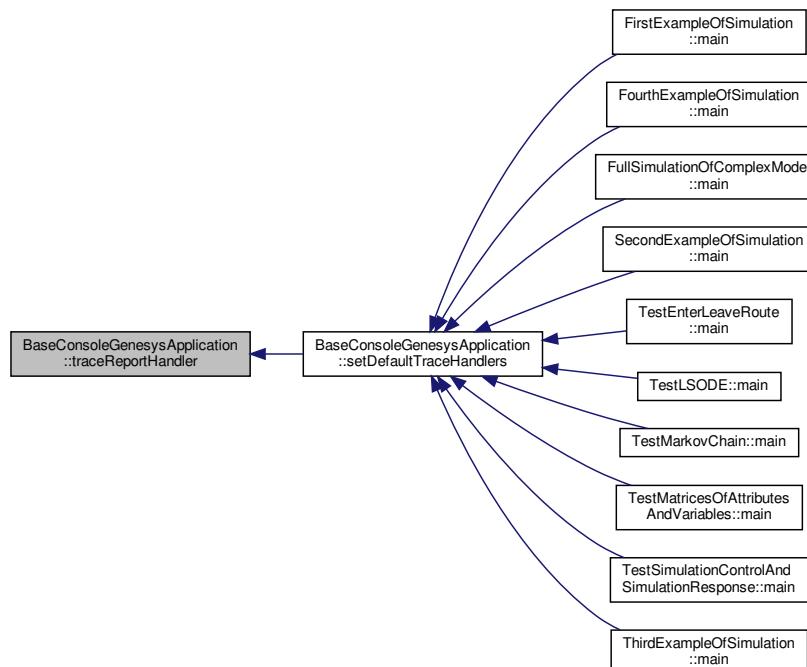
TraceEvent e) [protected], [virtual]

Definition at line 36 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

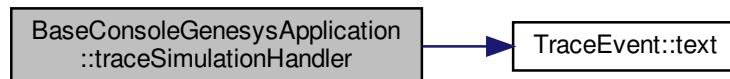


6.7.3.14 traceSimulationHandler()

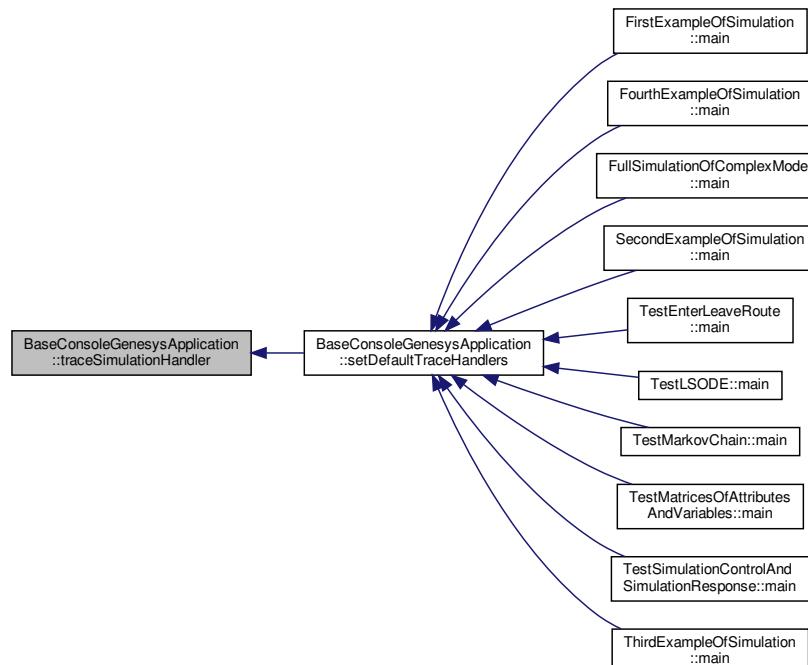
```
void BaseConsoleGenesysApplication::traceSimulationHandler (
    TraceSimulationEvent e ) [protected], [virtual]
```

Definition at line 40 of file [BaseConsoleGenesysApplication.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



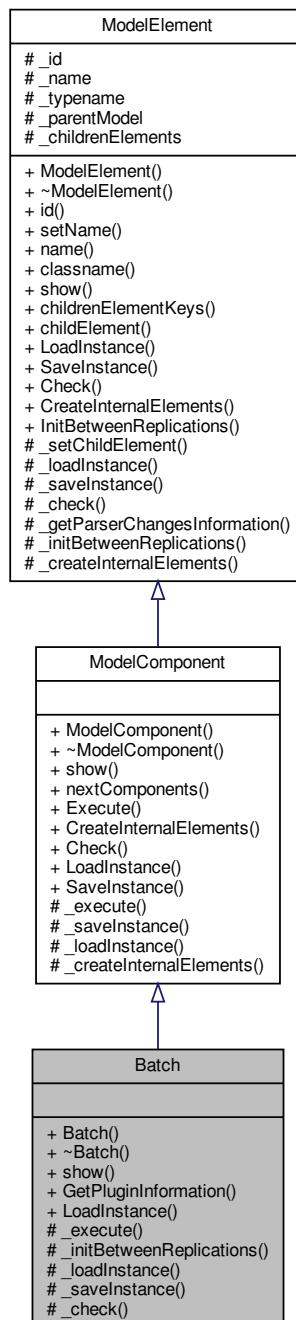
The documentation for this class was generated from the following files:

- [BaseConsoleGenesysApplication.h](#)
- [BaseConsoleGenesysApplication.cpp](#)

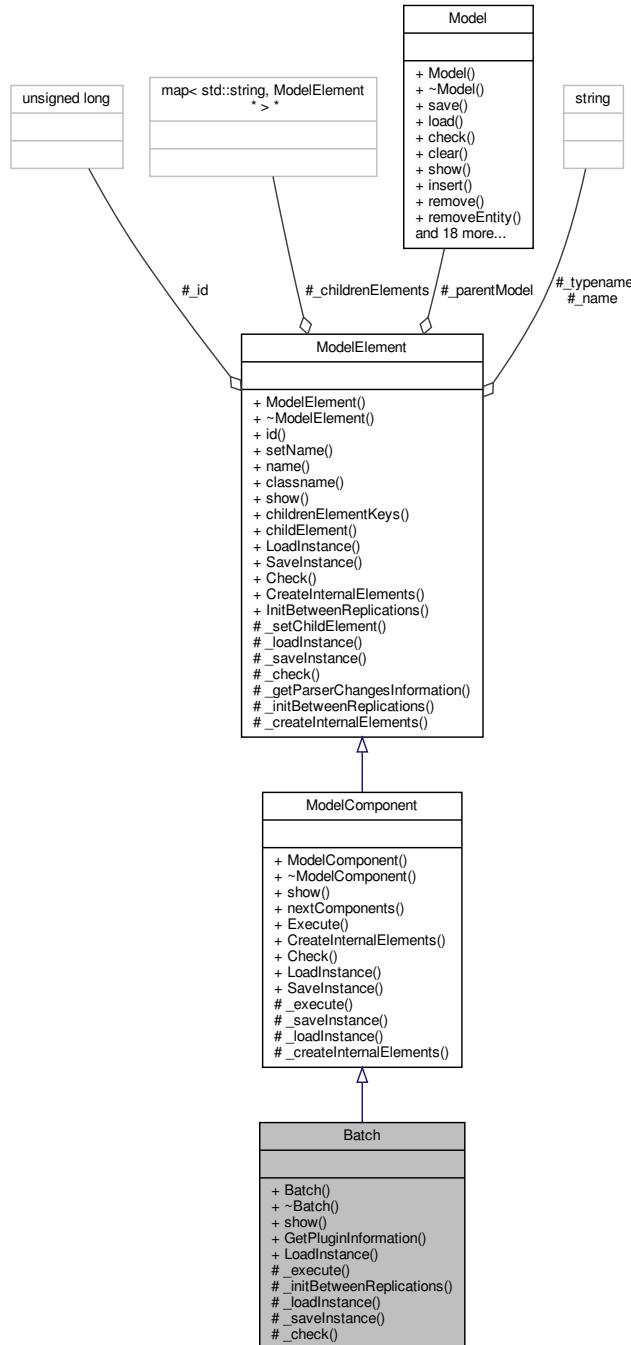
6.8 Batch Class Reference

```
#include <Batch.h>
```

Inheritance diagram for Batch:



Collaboration diagram for Batch:



Public Member Functions

- **Batch (Model *model, std::string name="")**
- virtual **~Batch ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.8.1 Detailed Description

Batch module DESCRIPTION This module is intended as the grouping mechanism within the simulation model. Batches can be permanently or temporarily grouped. Temporary batches must later be split using the **Separate** module. Batches may be made with any specified number of entering entities or may be matched together based on an attribute. Entities arriving at the **Batch** module are placed in a queue until the required number of entities has accumulated. Once accumulated, a new representative entity is created. TYPICAL USES Collect a number of parts before starting processing Reassemble previously separated copies of a form Bring together a patient and his record before commencing an appointment PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of batching entities together. **Batch** Size Number of entities to be batched. Save Criterion Method for assigning representative entity's user-defined attribute values. Rule Determines how incoming entities will be batched. Any **Entity** will take the first "Batch Size" number of entities and put them together. By **Attribute** signifies that the values of the specified attribute must match for entities to be grouped. For example, if **Attribute** Name is Color, all entities must have the same Color value to be grouped; otherwise, they will wait at the module for additional incoming entities. **Attribute** Name Name of the attribute whose value must match the value of the other incoming entities in order for a group to be made. Applies only when Rule is By **Attribute**. Representative **Entity** The entity type for the representative entity.

Definition at line 53 of file `Batch.h`.

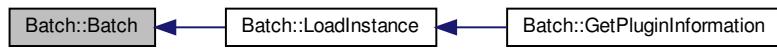
6.8.2 Constructor & Destructor Documentation

6.8.2.1 `Batch()`

```
Batch::Batch (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file `Batch.cpp`.

Here is the caller graph for this function:



6.8.2.2 ~Batch()

```
virtual Batch::~Batch ( ) [virtual], [default]
```

6.8.3 Member Function Documentation

6.8.3.1 _check()

```
bool Batch::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Batch.cpp](#).

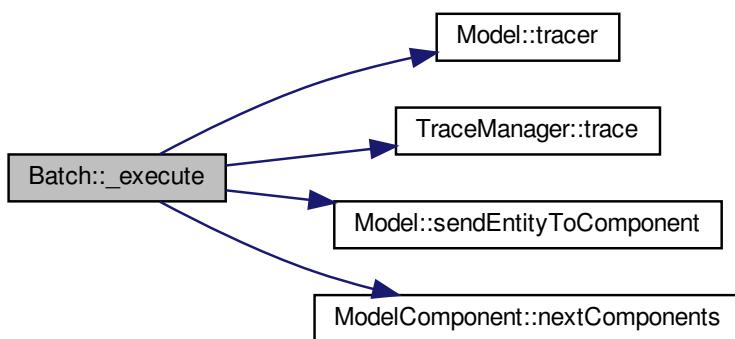
6.8.3.2 _execute()

```
void Batch::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Batch.cpp](#).

Here is the call graph for this function:



6.8.3.3 `_initBetweenReplications()`

```
void Batch::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [49](#) of file [Batch.cpp](#).

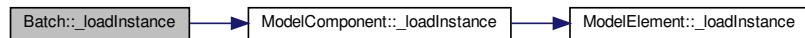
6.8.3.4 `_loadInstance()`

```
bool Batch::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

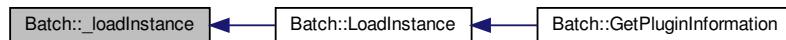
Reimplemented from [ModelComponent](#).

Definition at line [41](#) of file [Batch.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



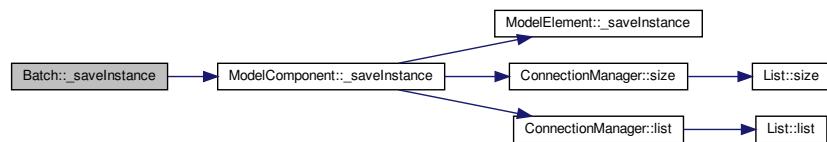
6.8.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Batch::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [52](#) of file [Batch.cpp](#).

Here is the call graph for this function:



6.8.3.6 GetPluginInformation()

```
PluginInformation * Batch::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Batch.cpp](#).

Here is the call graph for this function:

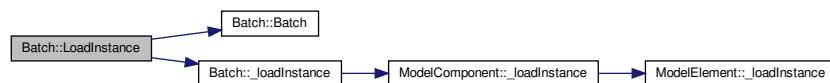


6.8.3.7 LoadInstance()

```
ModelComponent * Batch::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Batch.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



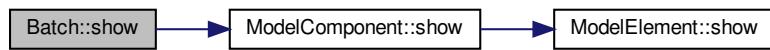
6.8.3.8 show()

```
std::string Batch::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [Batch.cpp](#).

Here is the call graph for this function:



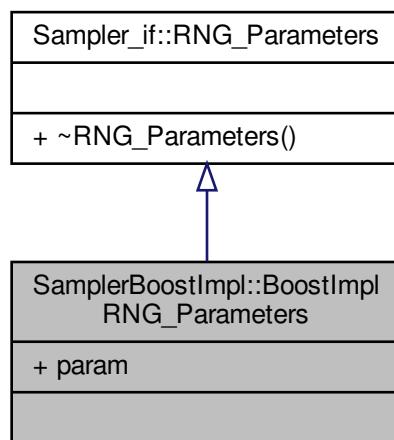
The documentation for this class was generated from the following files:

- [Batch.h](#)
- [Batch.cpp](#)

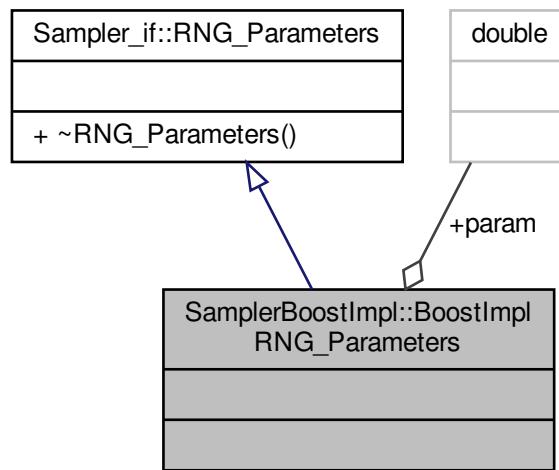
6.9 SamplerBoostImpl::BoostImplRNG_Parameters Struct Reference

```
#include <SamplerBoostImpl.h>
```

Inheritance diagram for SamplerBoostImpl::BoostImplRNG_Parameters:



Collaboration diagram for SamplerBoostImpl::BoostImplRNG_Parameters:



Public Attributes

- `double param`

Additional Inherited Members

6.9.1 Detailed Description

Definition at line 23 of file [SamplerBoostImpl.h](#).

6.9.2 Member Data Documentation

6.9.2.1 param

```
double SamplerBoostImpl::BoostImplRNG_Parameters::param
```

Definition at line 24 of file [SamplerBoostImpl.h](#).

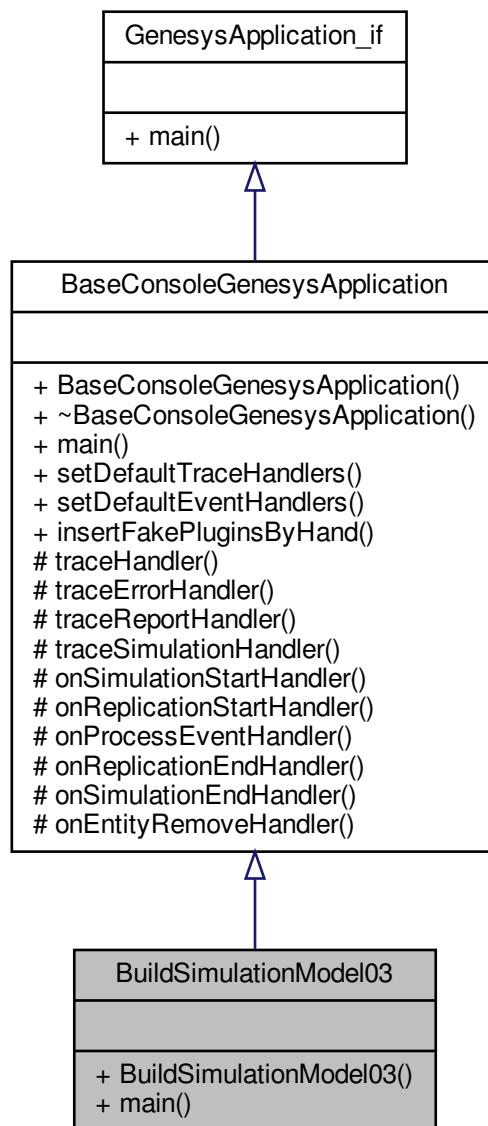
The documentation for this struct was generated from the following file:

- [SamplerBoostImpl.h](#)

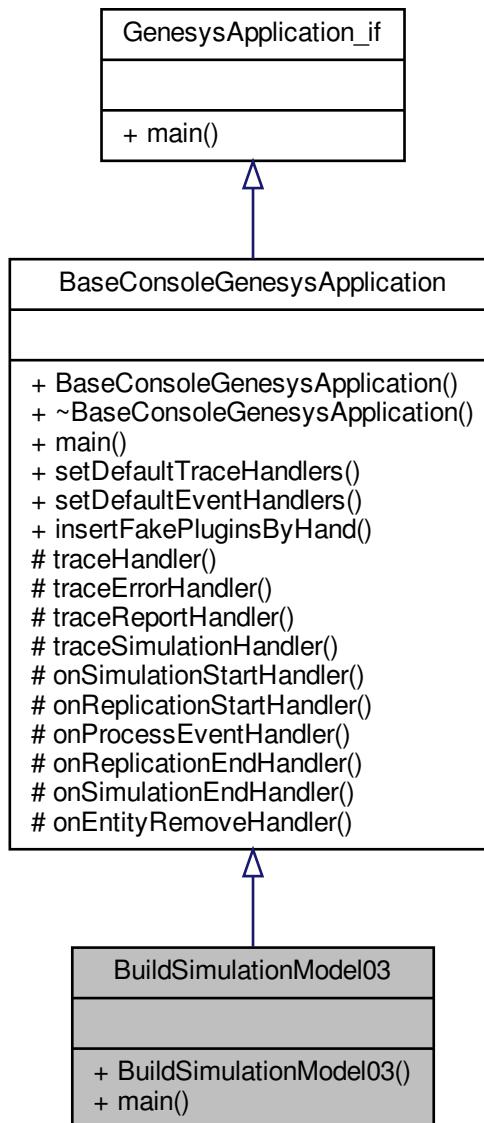
6.10 BuildSimulationModel03 Class Reference

```
#include <BuildSimulationModel03.h>
```

Inheritance diagram for BuildSimulationModel03:



Collaboration diagram for BuildSimulationModel03:



Public Member Functions

- [BuildSimulationModel03 \(\)](#)
- virtual int [main \(int argc, char **argv\)](#)

Additional Inherited Members

6.10.1 Detailed Description

Definition at line 21 of file [BuildSimulationModel03.h](#).

6.10.2 Constructor & Destructor Documentation

6.10.2.1 BuildSimulationModel03()

```
BuildSimulationModel03::BuildSimulationModel03 ( )
```

Definition at line 41 of file [BuildSimulationModel03.cpp](#).

6.10.3 Member Function Documentation

6.10.3.1 main()

```
int BuildSimulationModel03::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 204 of file [BuildSimulationModel03.cpp](#).

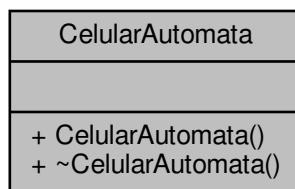
The documentation for this class was generated from the following files:

- [BuildSimulationModel03.h](#)
- [BuildSimulationModel03.cpp](#)

6.11 CelularAutomata Class Reference

```
#include <CellularAutomata.h>
```

Collaboration diagram for CelularAutomata:



Public Member Functions

- [CelularAutomata \(\)](#)
- virtual [~CelularAutomata \(\)=default](#)

6.11.1 Detailed Description

Definition at line [17](#) of file [CellularAutomata.h](#).

6.11.2 Constructor & Destructor Documentation

6.11.2.1 CelularAutomata()

```
CelularAutomata::CelularAutomata ( )
```

Definition at line [16](#) of file [CellularAutomata.cpp](#).

6.11.2.2 ~CelularAutomata()

```
virtual CelularAutomata::~CelularAutomata ( ) [virtual], [default]
```

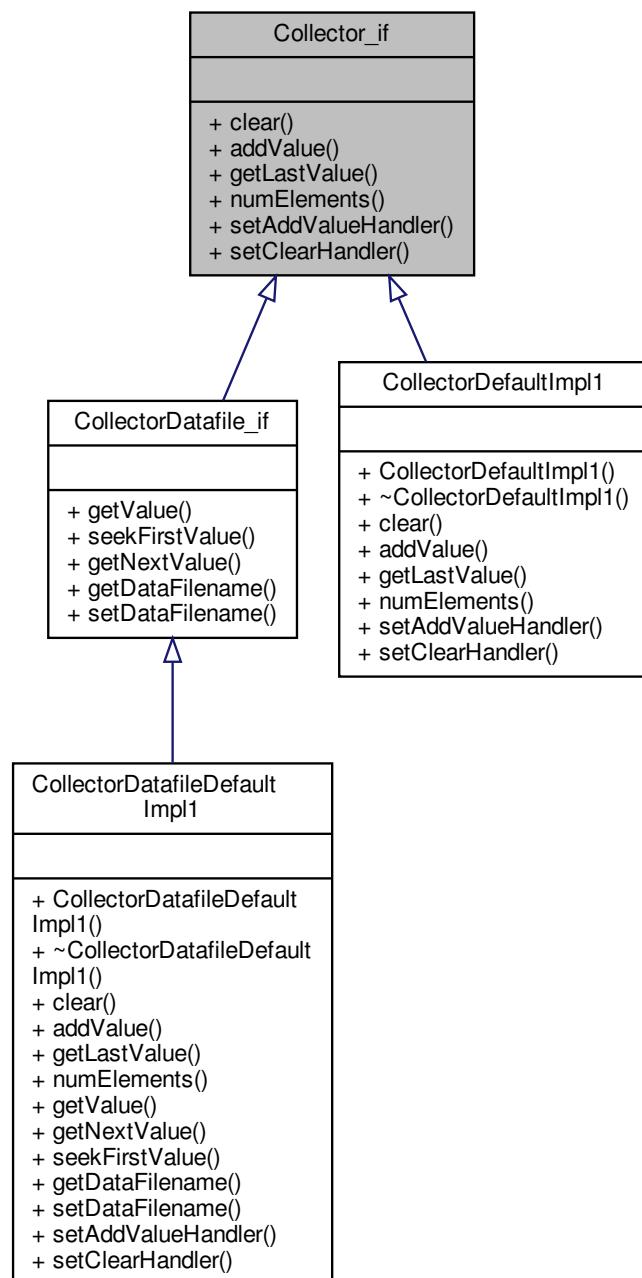
The documentation for this class was generated from the following files:

- [CellularAutomata.h](#)
- [CellularAutomata.cpp](#)

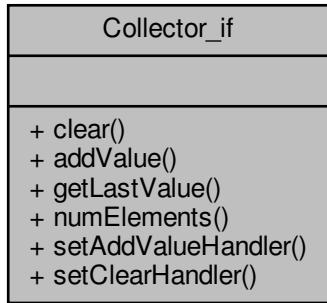
6.12 Collector_if Class Reference

```
#include <Collector_if.h>
```

Inheritance diagram for Collector_if:



Collaboration diagram for Collector_if:



Public Member Functions

- virtual void `clear ()=0`
- virtual void `addValue (double value)=0`
- virtual double `getLastValue ()=0`
- virtual unsigned long `numElements ()=0`
- virtual void `setAddValueHandler (CollectorAddValueHandler addValueHandler)=0`
- virtual void `setClearHandler (CollectorClearHandler clearHandler)=0`

6.12.1 Detailed Description

Interface for collecting values of a single stochastic variable. Values collected can be used as base for statistical analysis.

Definition at line 39 of file [Collector_if.h](#).

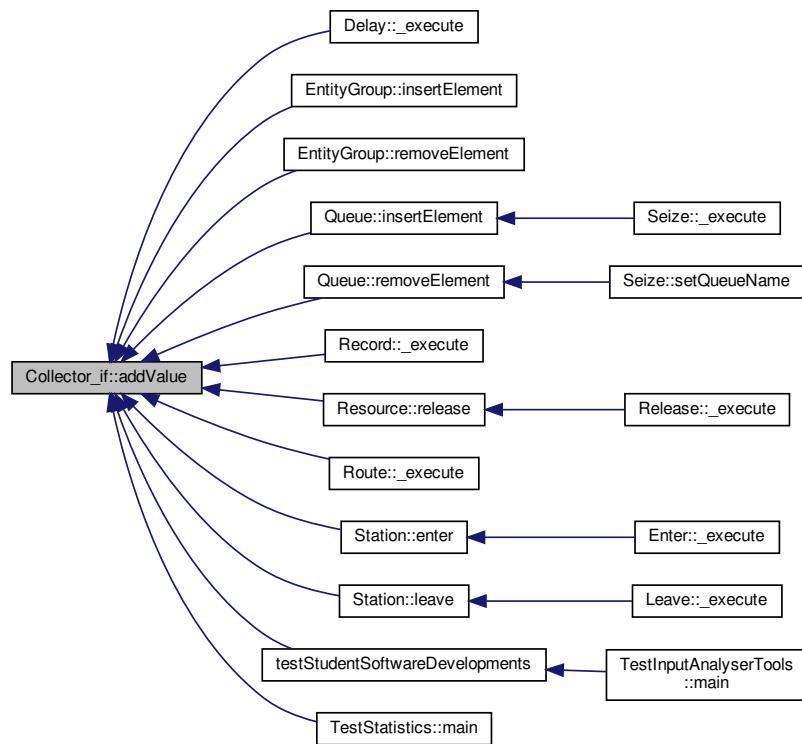
6.12.2 Member Function Documentation

6.12.2.1 addValue()

```
virtual void Collector_if::addValue (
    double value ) [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

Here is the caller graph for this function:

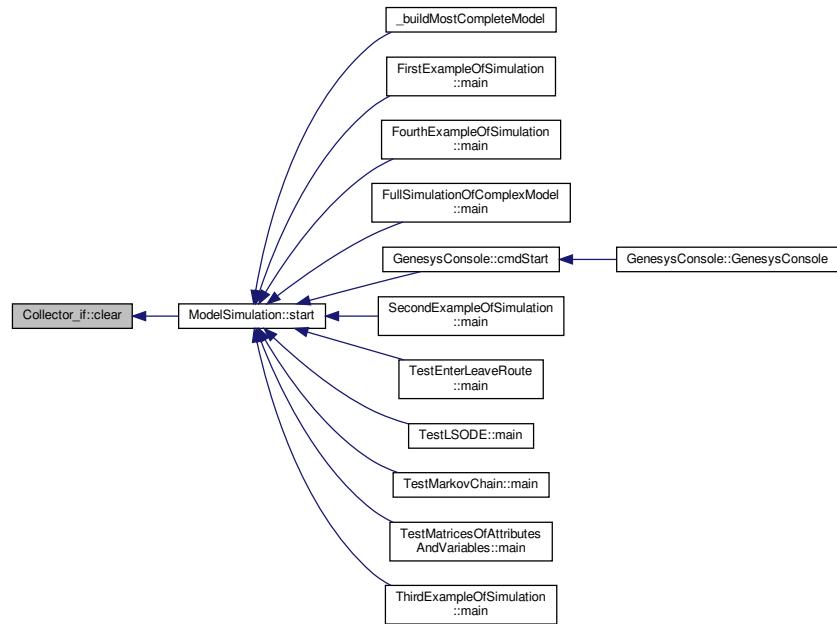


6.12.2.2 clear()

```
virtual void Collector_if::clear ( ) [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

Here is the caller graph for this function:



6.12.2.3 getLastValue()

```
virtual double Collector_if::getLastValue () [pure virtual]
```

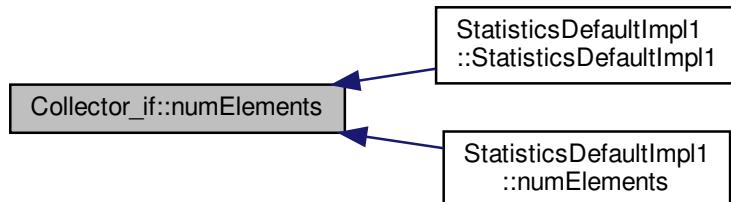
Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

6.12.2.4 numElements()

```
virtual unsigned long Collector_if::numElements () [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

Here is the caller graph for this function:

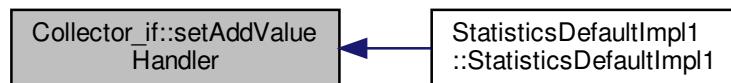


6.12.2.5 setAddValueHandler()

```
virtual void Collector_if::setAddValueHandler (
    CollectorAddValueHandler addValueHandler ) [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

Here is the caller graph for this function:

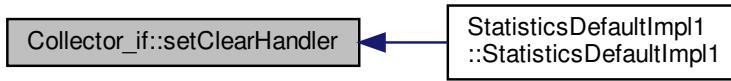


6.12.2.6 setClearHandler()

```
virtual void Collector_if::setClearHandler (
    CollectorClearHandler clearHandler ) [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDefaultImpl1](#).

Here is the caller graph for this function:



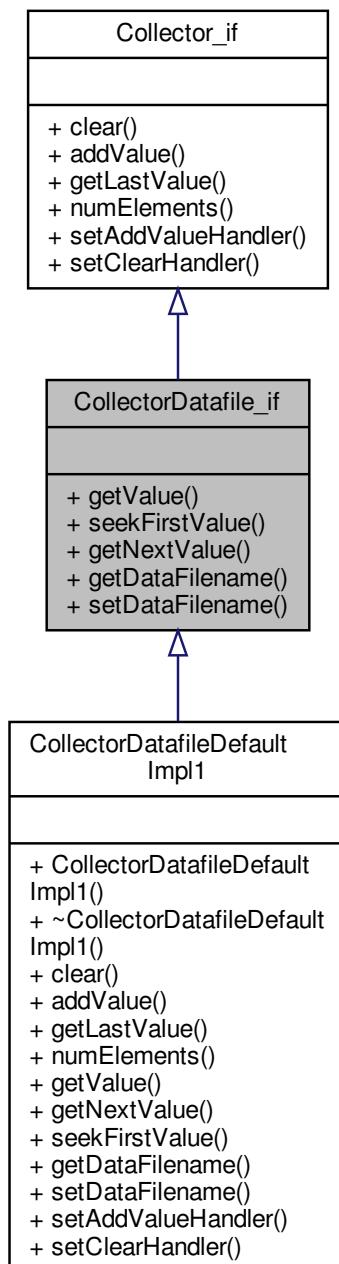
The documentation for this class was generated from the following file:

- [Collector_if.h](#)

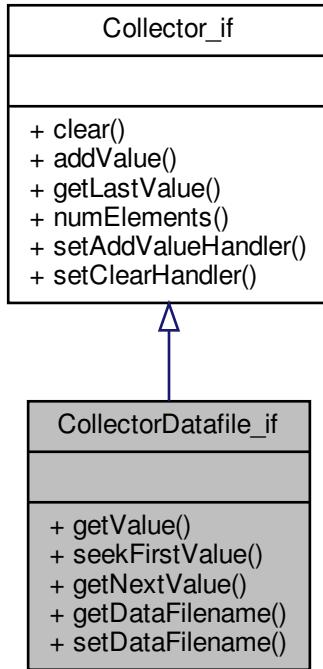
6.13 CollectorDatafile_if Class Reference

```
#include <CollectorDatafile_if.h>
```

Inheritance diagram for CollectorDatafile_if:



Collaboration diagram for CollectorDatafile_if:



Public Member Functions

- virtual double `getValue` (unsigned int rank)=0
- virtual void `seekFirstValue` ()=0
- virtual double `getNextValue` ()=0
- virtual std::string `getDataFilename` ()=0
- virtual void `setDataFilename` (std::string filename)=0

6.13.1 Detailed Description

Interface for collecting values of a stochastic variable that will be stores in a datafile.

Definition at line 22 of file [CollectorDatafile_if.h](#).

6.13.2 Member Function Documentation

6.13.2.1 getDataFilename()

```
virtual std::string CollectorDatafile_if::getDataFilename() [pure virtual]
```

Get the next value in the file and advances the pointer

Implemented in [CollectorDatafileDefaultImpl1](#).

Here is the caller graph for this function:



6.13.2.2 getNextValue()

```
virtual double CollectorDatafile_if::getNextValue() [pure virtual]
```

Set the pointer to the first value in the file

Implemented in [CollectorDatafileDefaultImpl1](#).

6.13.2.3 getValue()

```
virtual double CollectorDatafile_if::getValue( unsigned int rank ) [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#).

6.13.2.4 seekFirstValue()

```
virtual void CollectorDatafile_if::seekFirstValue() [pure virtual]
```

Get a value from a specific position

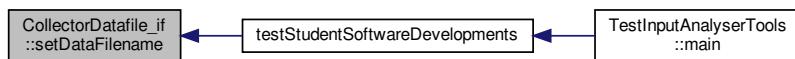
Implemented in [CollectorDatafileDefaultImpl1](#).

6.13.2.5 setDataFilename()

```
virtual void CollectorDatafile_if::setDataFilename (  
    std::string filename ) [pure virtual]
```

Implemented in [CollectorDatafileDefaultImpl1](#).

Here is the caller graph for this function:



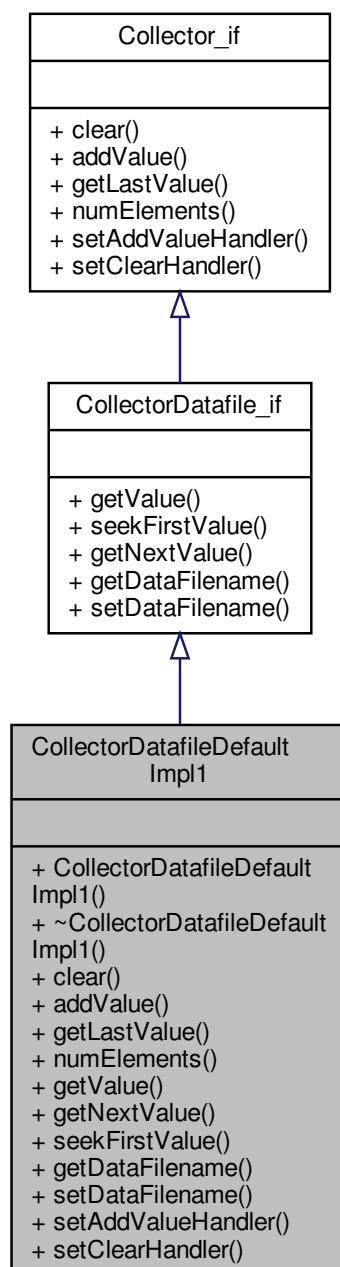
The documentation for this class was generated from the following file:

- [CollectorDatafile_if.h](#)

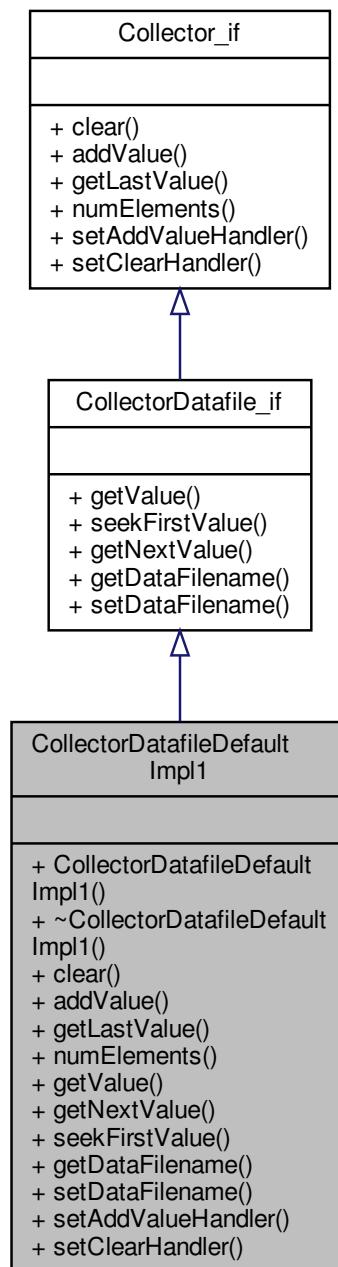
6.14 CollectorDatafileDefaultImpl1 Class Reference

```
#include <CollectorDatafileDefaultImpl1.h>
```

Inheritance diagram for CollectorDatafileDefaultImpl1:



Collaboration diagram for CollectorDatafileDefaultImpl1:



Public Member Functions

- `CollectorDatafileDefaultImpl1 ()`
- virtual `~CollectorDatafileDefaultImpl1 ()=default`
- `void clear ()`
- `void addValue (double value)`
- `double getLastValue ()`

- `unsigned long numElements ()`
- `double getValue (unsigned int num)`
- `double getNextValue ()`
- `void seekFirstValue ()`
- `std::string getDataFilename ()`
- `void setDataFilename (std::string filename)`
- `void setAddValueHandler (CollectorAddValueHandler addValueHandler)`
- `void setClearHandler (CollectorClearHandler clearHandler)`

6.14.1 Detailed Description

Definition at line 21 of file [CollectorDatafileDefaultImpl1.h](#).

6.14.2 Constructor & Destructor Documentation

6.14.2.1 CollectorDatafileDefaultImpl1()

```
CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 ( )
```

Definition at line 16 of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.2.2 ~CollectorDatafileDefaultImpl1()

```
virtual CollectorDatafileDefaultImpl1::~CollectorDatafileDefaultImpl1 ( ) [virtual], [default]
```

6.14.3 Member Function Documentation

6.14.3.1 addValue()

```
void CollectorDatafileDefaultImpl1::addValue ( double value ) [virtual]
```

Implements [Collector_if](#).

Definition at line 23 of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.2 clear()

```
void CollectorDatafileDefaultImpl1::clear ( ) [virtual]
```

Implements [Collector_if](#).

Definition at line 20 of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.3 `getDataFilename()`

```
std::string CollectorDatafileDefaultImpl1::getDataFilename () [virtual]
```

Get the next value in the file and advances the pointer

Implements [CollectorDatafile_if](#).

Definition at line [45](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.4 `getLastValue()`

```
double CollectorDatafileDefaultImpl1::getLastValue () [virtual]
```

Implements [Collector_if](#).

Definition at line [26](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.5 `getNextValue()`

```
double CollectorDatafileDefaultImpl1::getNextValue () [virtual]
```

Set the pointer to the first value in the file

Implements [CollectorDatafile_if](#).

Definition at line [38](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.6 `getValue()`

```
double CollectorDatafileDefaultImpl1::getValue (
    unsigned int num ) [virtual]
```

Implements [CollectorDatafile_if](#).

Definition at line [34](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.7 `numElements()`

```
unsigned long CollectorDatafileDefaultImpl1::numElements () [virtual]
```

Implements [Collector_if](#).

Definition at line [30](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.8 seekFirstValue()

```
void CollectorDatafileDefaultImpl1::seekFirstValue ( ) [virtual]
```

Get a value from a specific position

Implements [CollectorDatafile_if](#).

Definition at line [42](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.9 setAddValueHandler()

```
void CollectorDatafileDefaultImpl1::setAddValueHandler (
    CollectorAddValueHandler addValueHandler ) [virtual]
```

Implements [Collector_if](#).

Definition at line [53](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.10 setClearHandler()

```
void CollectorDatafileDefaultImpl1::setClearHandler (
    CollectorClearHandler clearHandler ) [virtual]
```

Implements [Collector_if](#).

Definition at line [57](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

6.14.3.11 setDataFilename()

```
void CollectorDatafileDefaultImpl1::setDataFilename (
    std::string filename ) [virtual]
```

Implements [CollectorDatafile_if](#).

Definition at line [49](#) of file [CollectorDatafileDefaultImpl1.cpp](#).

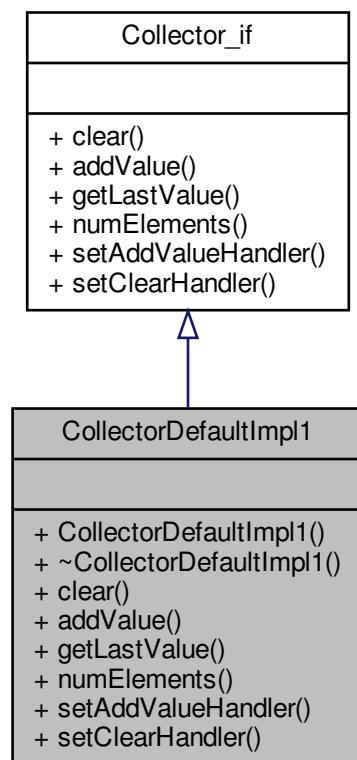
The documentation for this class was generated from the following files:

- [CollectorDatafileDefaultImpl1.h](#)
- [CollectorDatafileDefaultImpl1.cpp](#)

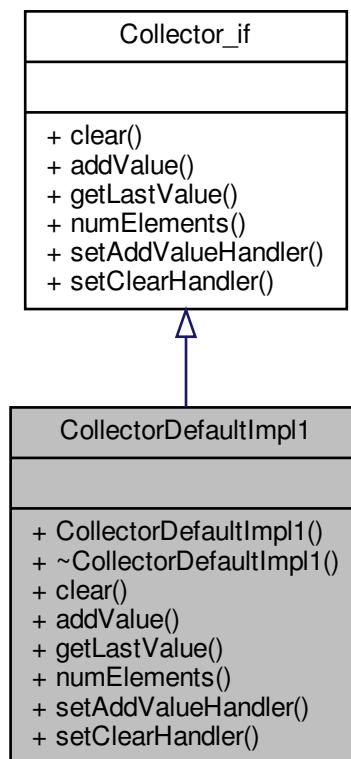
6.15 CollectorDefaultImpl1 Class Reference

```
#include <CollectorDefaultImpl1.h>
```

Inheritance diagram for CollectorDefaultImpl1:



Collaboration diagram for CollectorDefaultImpl1:



Public Member Functions

- `CollectorDefaultImpl1 ()`
- virtual `~CollectorDefaultImpl1 ()=default`
- `void clear ()`
- `void addValue (double value)`
- `double getLastValue ()`
- `unsigned long numElements ()`
- `void setAddValueHandler (CollectorAddValueHandler addValueHandler)`
- `void setClearHandler (CollectorClearHandler clearHandler)`

6.15.1 Detailed Description

Definition at line 19 of file [CollectorDefaultImpl1.h](#).

6.15.2 Constructor & Destructor Documentation

6.15.2.1 CollectorDefaultImpl1()

```
CollectorDefaultImpl1::CollectorDefaultImpl1 ( )
```

Definition at line 16 of file [CollectorDefaultImpl1.cpp](#).

6.15.2.2 ~CollectorDefaultImpl1()

```
virtual CollectorDefaultImpl1::~CollectorDefaultImpl1 ( ) [virtual], [default]
```

6.15.3 Member Function Documentation

6.15.3.1 addValue()

```
void CollectorDefaultImpl1::addValue ( double value ) [virtual]
```

Implements [Collector_if](#).

Definition at line 27 of file [CollectorDefaultImpl1.cpp](#).

6.15.3.2 clear()

```
void CollectorDefaultImpl1::clear ( ) [virtual]
```

Implements [Collector_if](#).

Definition at line 20 of file [CollectorDefaultImpl1.cpp](#).

6.15.3.3 getLastValue()

```
double CollectorDefaultImpl1::getLastValue ( ) [virtual]
```

Implements [Collector_if](#).

Definition at line 35 of file [CollectorDefaultImpl1.cpp](#).

6.15.3.4 numElements()

```
unsigned long CollectorDefaultImpl1::numElements ( ) [virtual]
```

Implements [Collector_if](#).

Definition at line 39 of file [CollectorDefaultImpl1.cpp](#).

6.15.3.5 setAddValueHandler()

```
void CollectorDefaultImpl1::setAddValueHandler (
    CollectorAddValueHandler addValueHandler ) [virtual]
```

Implements [Collector_if](#).

Definition at line 43 of file [CollectorDefaultImpl1.cpp](#).

6.15.3.6 setClearHandler()

```
void CollectorDefaultImpl1::setClearHandler (
    CollectorClearHandler clearHandler ) [virtual]
```

Implements [Collector_if](#).

Definition at line 47 of file [CollectorDefaultImpl1.cpp](#).

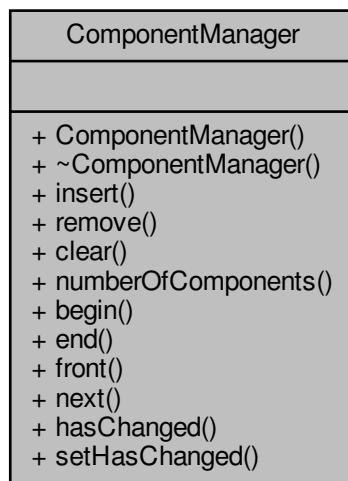
The documentation for this class was generated from the following files:

- [CollectorDefaultImpl1.h](#)
- [CollectorDefaultImpl1.cpp](#)

6.16 ComponentManager Class Reference

```
#include <ComponentManager.h>
```

Collaboration diagram for ComponentManager:



Public Member Functions

- `ComponentManager (Model *model)`
- `virtual ~ComponentManager ()=default`
- `bool insert (ModelComponent *comp)`
- `void remove (ModelComponent *comp)`
- `void clear ()`
- `unsigned int numberOfComponents ()`
- `std::list< ModelComponent * >::iterator begin ()`
- `std::list< ModelComponent * >::iterator end ()`
- `ModelComponent * front ()`
- `ModelComponent * next ()`
- `bool hasChanged () const`
- `void setHasChanged (bool _hasChanged)`

6.16.1 Detailed Description

Definition at line 21 of file [ComponentManager.h](#).

6.16.2 Constructor & Destructor Documentation

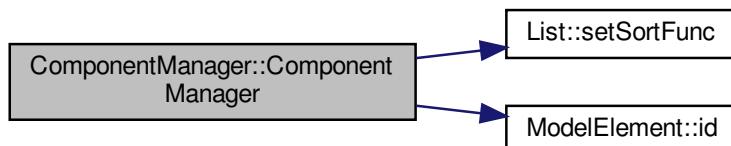
6.16.2.1 ComponentManager()

```
ComponentManager::ComponentManager (
    Model * model )
```

Components are sorted by ID

Definition at line 18 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



6.16.2.2 ~ComponentManager()

```
virtual ComponentManager::~ComponentManager ( ) [virtual], [default]
```

6.16.3 Member Function Documentation

6.16.3.1 begin()

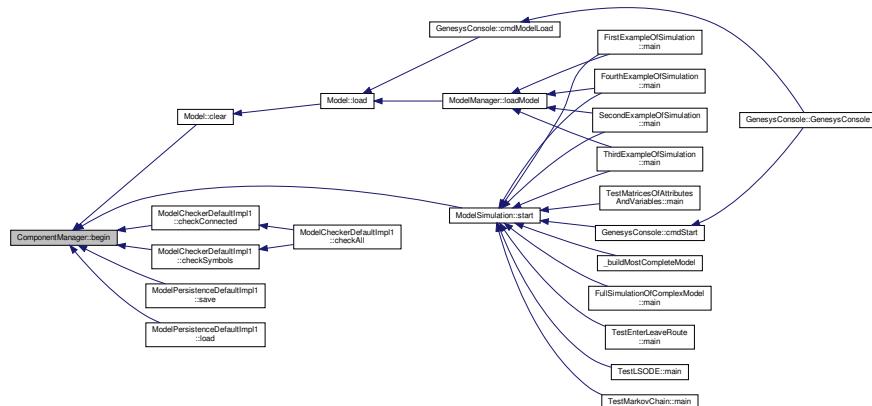
```
std::list< ModelComponent * >::iterator ComponentManager::begin ( )
```

Definition at line 58 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.2 clear()

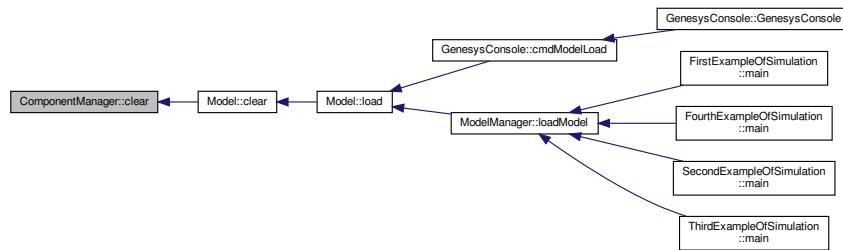
```
void ComponentManager::clear ( )
```

Definition at line 43 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

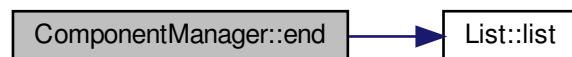


6.16.3.3 end()

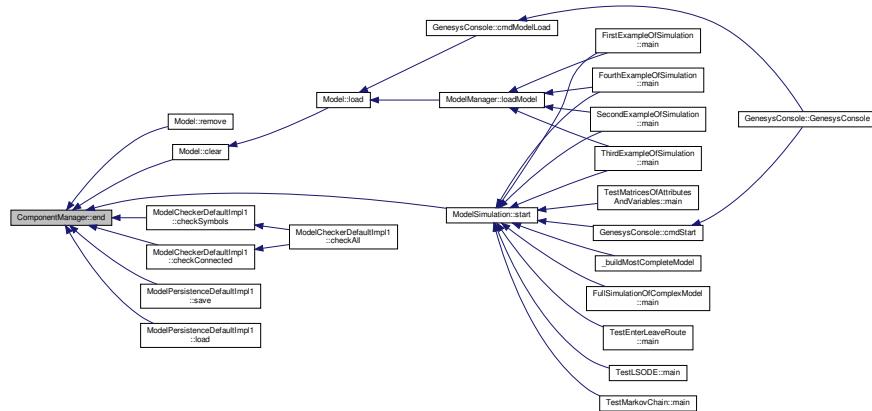
```
std::list< ModelComponent * >::iterator ComponentManager::end ( )
```

Definition at line 62 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.4 front()

```
ModelComponent * ComponentManager::front ( )
```

Definition at line 66 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



6.16.3.5 hasChanged()

```
bool ComponentManager::hasChanged ( ) const
```

Definition at line 74 of file [ComponentManager.cpp](#).

Here is the caller graph for this function:

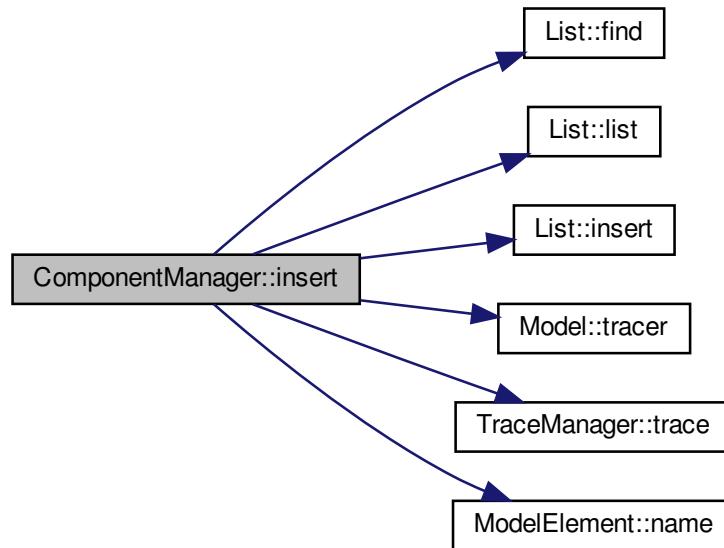


6.16.3.6 insert()

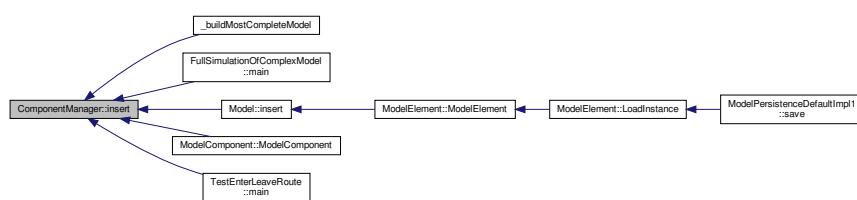
```
bool ComponentManager::insert (
    ModelComponent * comp )
```

Definition at line 26 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.7 next()

```
ModelComponent * ComponentManager::next ( )
```

Definition at line 70 of file [ComponentManager.cpp](#).

Here is the call graph for this function:

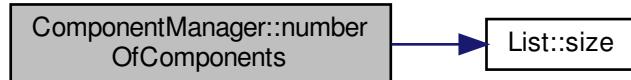


6.16.3.8 numberOfComponents()

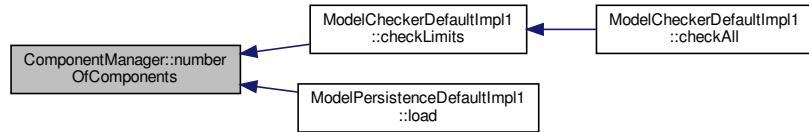
```
unsigned int ComponentManager::numberOfComponents ( )
```

Definition at line 54 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

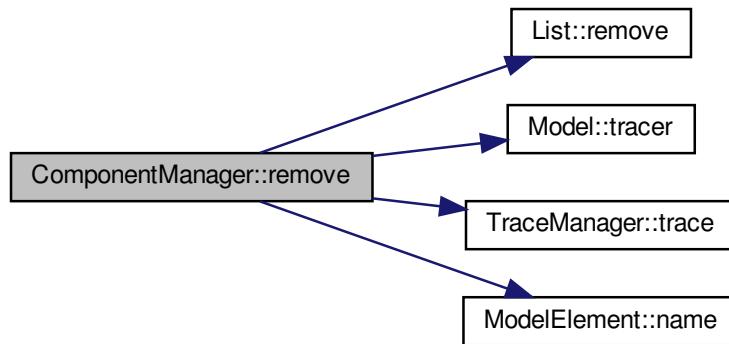


6.16.3.9 remove()

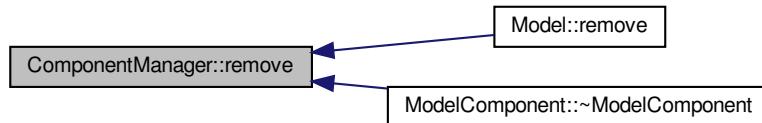
```
void ComponentManager::remove (
    ModelComponent * comp )
```

Definition at line 37 of file [ComponentManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3.10 setHasChanged()

```
void ComponentManager::setHasChanged (
    bool _hasChanged )
```

Definition at line 78 of file [ComponentManager.cpp](#).

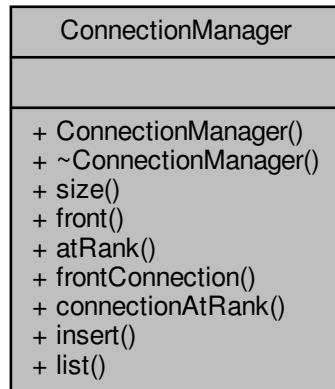
The documentation for this class was generated from the following files:

- [ComponentManager.h](#)
- [ComponentManager.cpp](#)

6.17 ConnectionManager Class Reference

```
#include <ConnectionManager.h>
```

Collaboration diagram for ConnectionManager:



Public Member Functions

- [ConnectionManager \(\)](#)
- virtual [~ConnectionManager \(\)=default](#)
- [unsigned int size \(\)](#)
- [ModelComponent * front \(\)](#)
- [ModelComponent * atRank \(unsigned int rank\)](#)
- [Connection * frontConnection \(\)](#)
- [Connection * connectionAtRank \(unsigned int rank\)](#)
- [void insert \(ModelComponent *component, unsigned int inputNumber=0\)](#)
- [std::list< Connection * > * list \(\) const](#)

6.17.1 Detailed Description

Definition at line [24](#) of file [ConnectionManager.h](#).

6.17.2 Constructor & Destructor Documentation

6.17.2.1 ConnectionManager()

```
ConnectionManager::ConnectionManager ( )
```

Definition at line [16](#) of file [ConnectionManager.cpp](#).

6.17.2.2 ~ConnectionManager()

```
virtual ConnectionManager::~ConnectionManager ( ) [virtual], [default]
```

6.17.3 Member Function Documentation

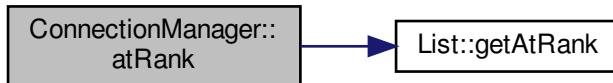
6.17.3.1 atRank()

```
ModelComponent * ConnectionManager::atRank ( unsigned int rank )
```

DEPRECATED. Use getConnectionAtRank instead

Definition at line 28 of file [ConnectionManager.cpp](#).

Here is the call graph for this function:

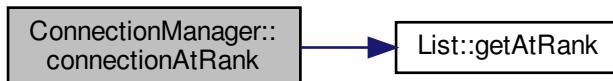


6.17.3.2 connectionAtRank()

```
Connection * ConnectionManager::connectionAtRank ( unsigned int rank )
```

Definition at line 36 of file [ConnectionManager.cpp](#).

Here is the call graph for this function:



6.17.3.3 front()

```
ModelComponent * ConnectionManager::front ( )
```

DEPRECATED. Use frontConnection instead

Definition at line 24 of file [ConnectionManager.cpp](#).

Here is the call graph for this function:



6.17.3.4 frontConnection()

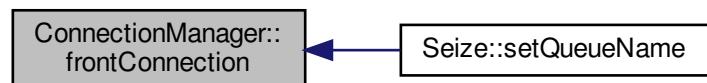
```
Connection * ConnectionManager::frontConnection ( )
```

Definition at line 32 of file [ConnectionManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

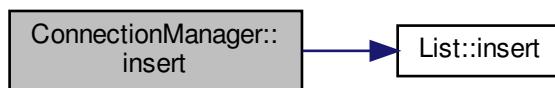


6.17.3.5 insert()

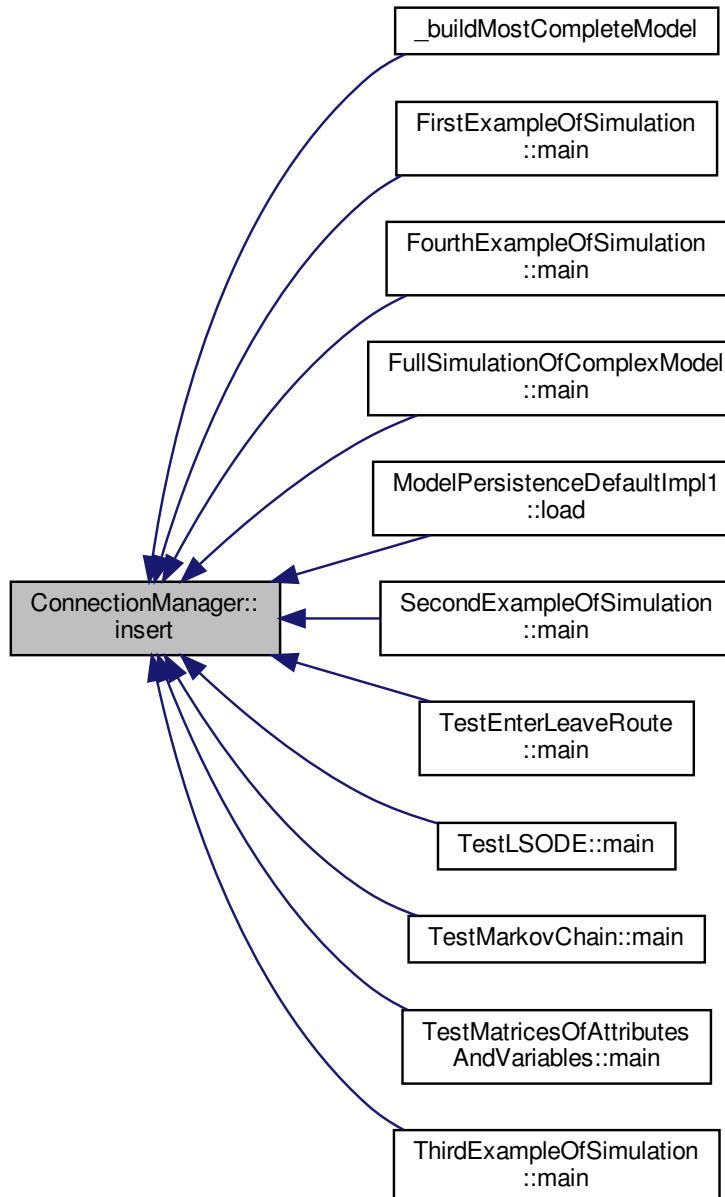
```
void ConnectionManager::insert (
    ModelComponent * component,
    unsigned int inputNumber = 0 )
```

Definition at line [40](#) of file [ConnectionManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.17.3.6 list()

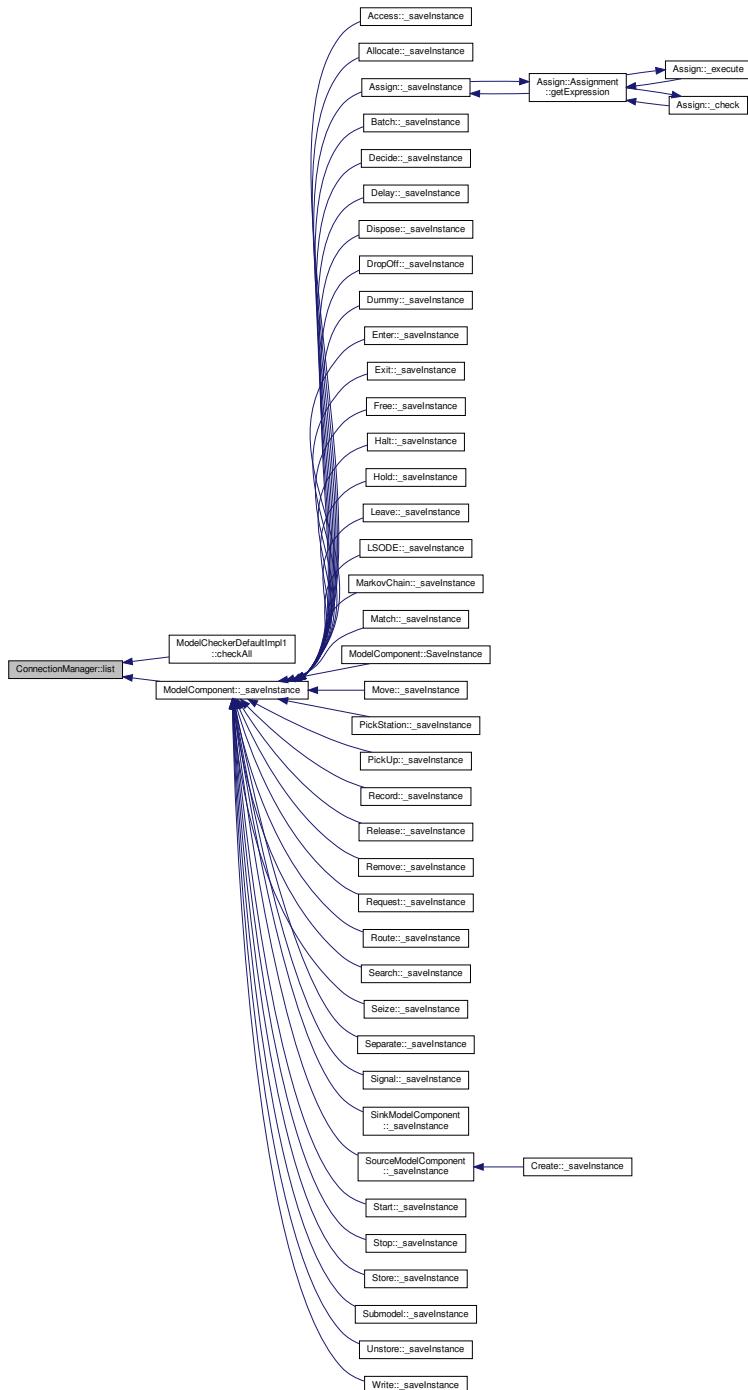
```
std::list< Connection * > * ConnectionManager::list ( ) const
```

Definition at line 44 of file [ConnectionManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.17.3.7 `size()`

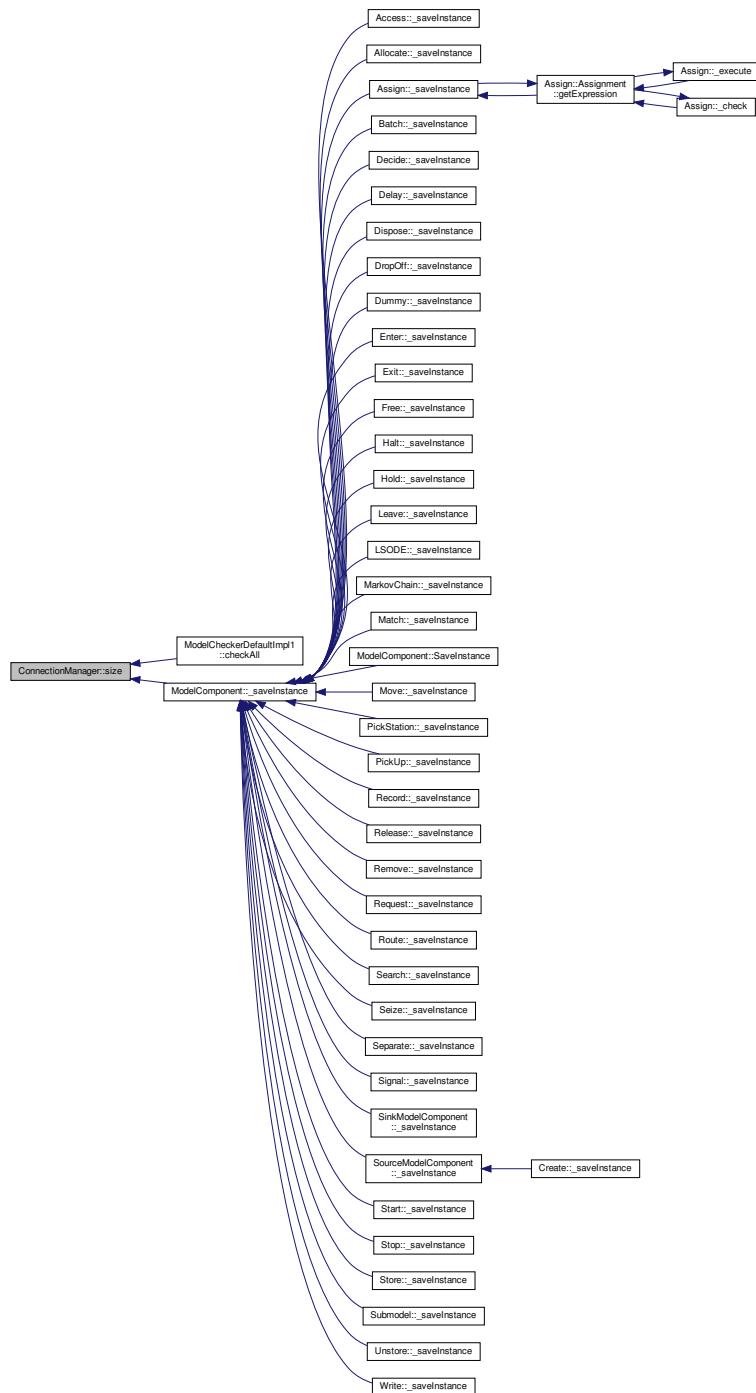
```
unsigned int ConnectionManager::size( )
```

Definition at line 20 of file [ConnectionManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



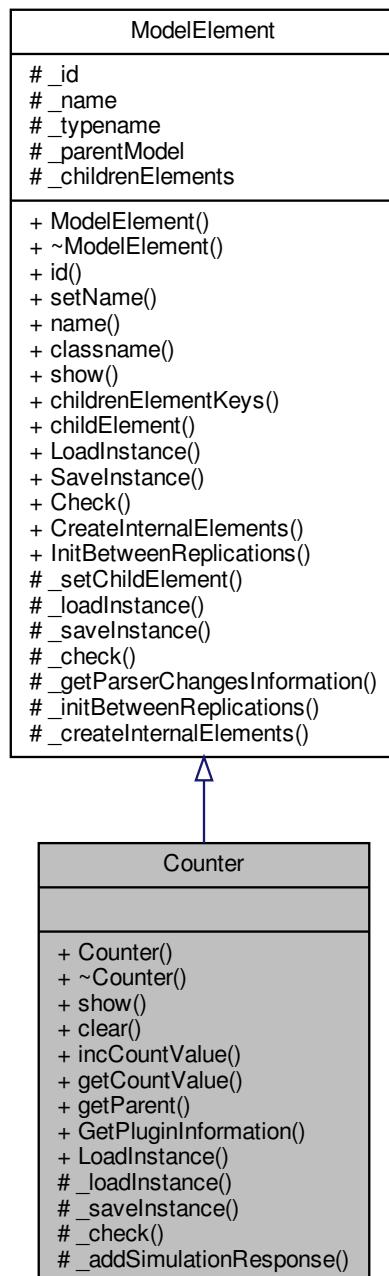
The documentation for this class was generated from the following files:

- [ConnectionManager.h](#)
- [ConnectionManager.cpp](#)

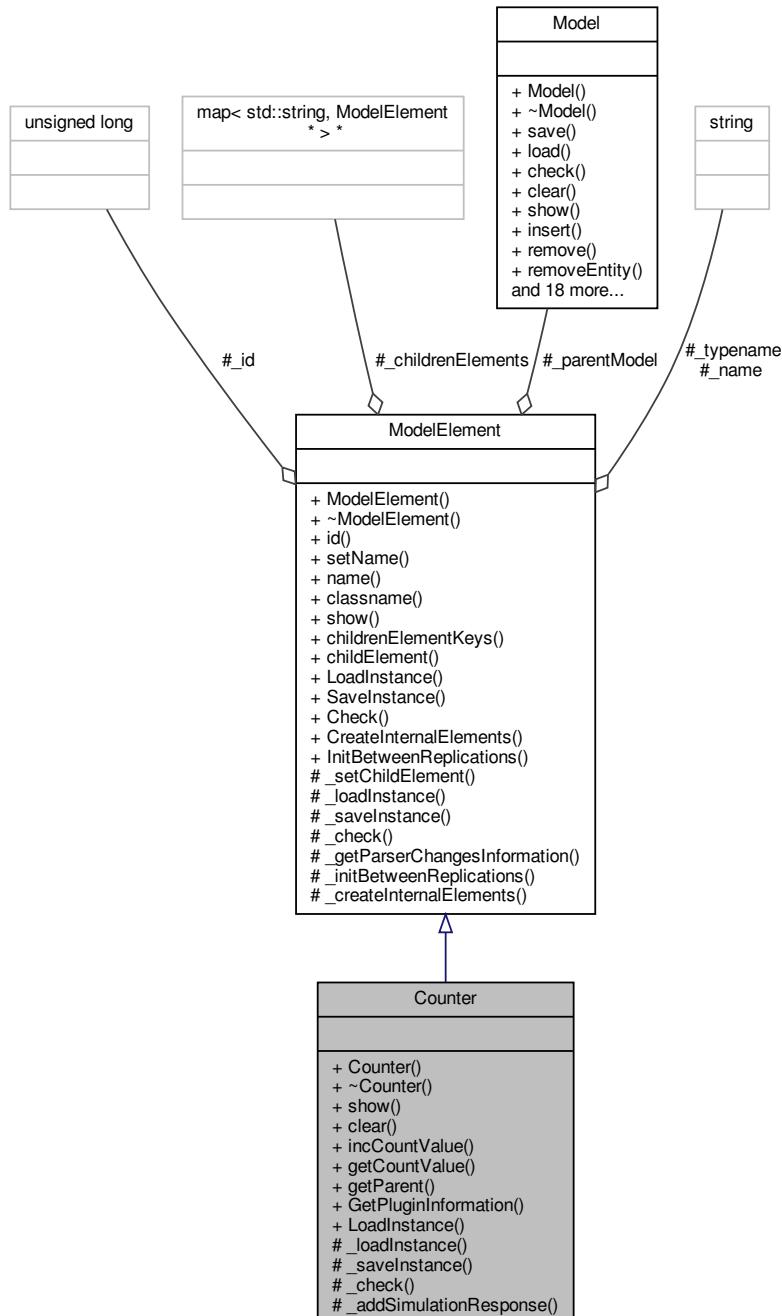
6.18 Counter Class Reference

```
#include <Counter.h>
```

Inheritance diagram for Counter:



Collaboration diagram for Counter:



Public Member Functions

- `Counter (Model *model, std::string name="", ModelElement *parent=nullptr)`
- `virtual ~Counter ()=default`
- `virtual std::string show ()`
- `void clear ()`
- `void incCountValue (int value=1)`
- `unsigned long getCountValue () const`
- `ModelElement * getParent () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- void `_addSimulationResponse ()`

Additional Inherited Members

6.18.1 Detailed Description

The `Counter` element is used to count events, and its internal count value is added by a configurable amount, usually incremented by one.

Definition at line 24 of file `Counter.h`.

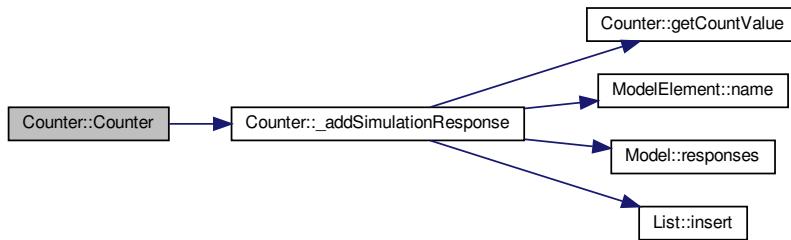
6.18.2 Constructor & Destructor Documentation

6.18.2.1 Counter()

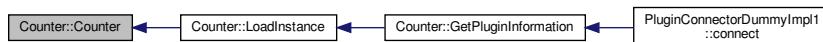
```
Counter::Counter (
    Model * model,
    std::string name = "",
    ModelElement * parent = nullptr )
```

Definition at line 18 of file `Counter.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.2.2 ~Counter()

```
virtual Counter::~Counter() [virtual], [default]
```

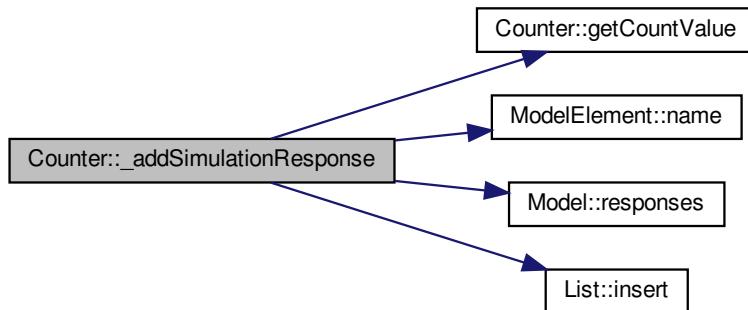
6.18.3 Member Function Documentation

6.18.3.1 _addSimulationResponse()

```
void Counter::_addSimulationResponse() [protected]
```

Definition at line 24 of file [Counter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.3.2 _check()

```
bool Counter::_check(
    std::string * errorMessage) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 80 of file [Counter.cpp](#).

6.18.3.3 `_loadInstance()`

```
bool Counter::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

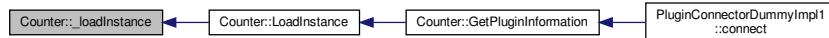
Reimplemented from [ModelElement](#).

Definition at line 71 of file [Counter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



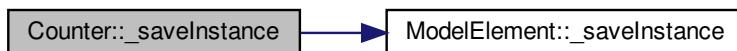
6.18.3.4 `_saveInstance()`

```
std::map< std::string, std::string * > * Counter::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 75 of file [Counter.cpp](#).

Here is the call graph for this function:

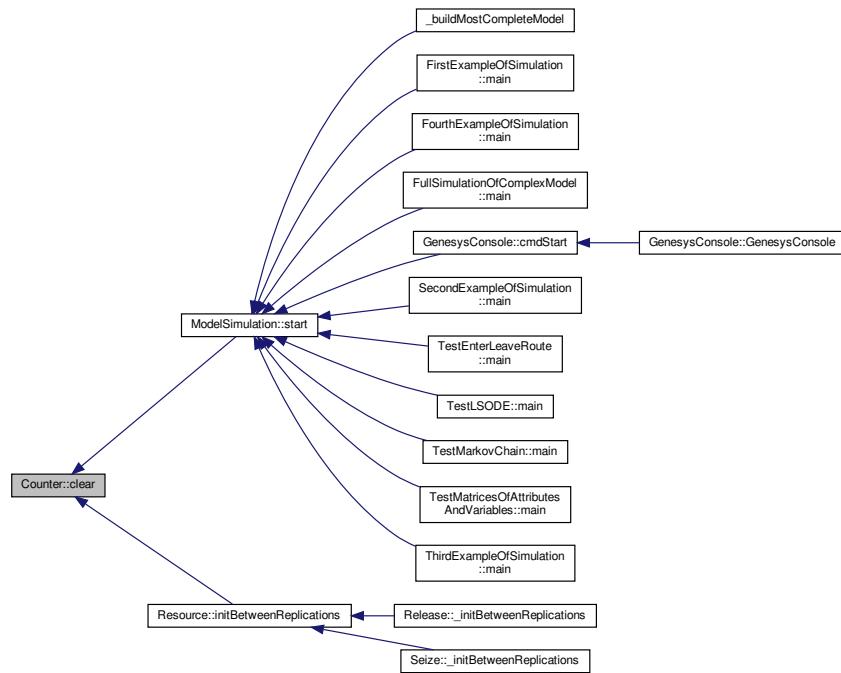


6.18.3.5 clear()

```
void Counter::clear ( )
```

Definition at line 39 of file [Counter.cpp](#).

Here is the caller graph for this function:

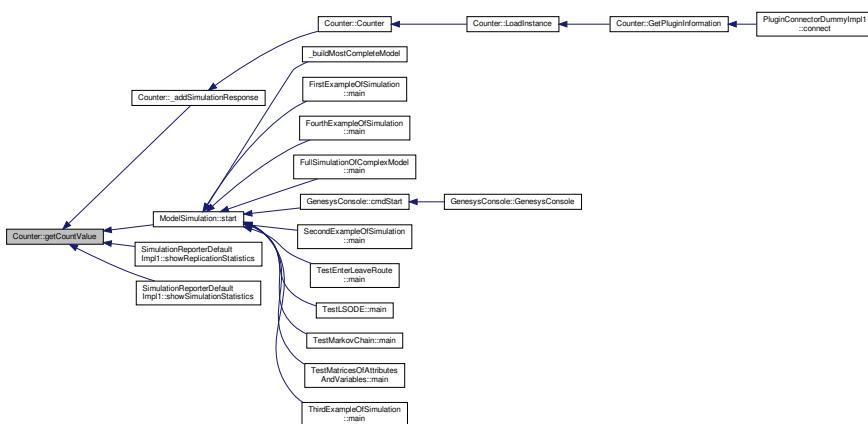


6.18.3.6 getCountValue()

```
unsigned long Counter::getCountValue ( ) const
```

Definition at line 47 of file [Counter.cpp](#).

Here is the caller graph for this function:

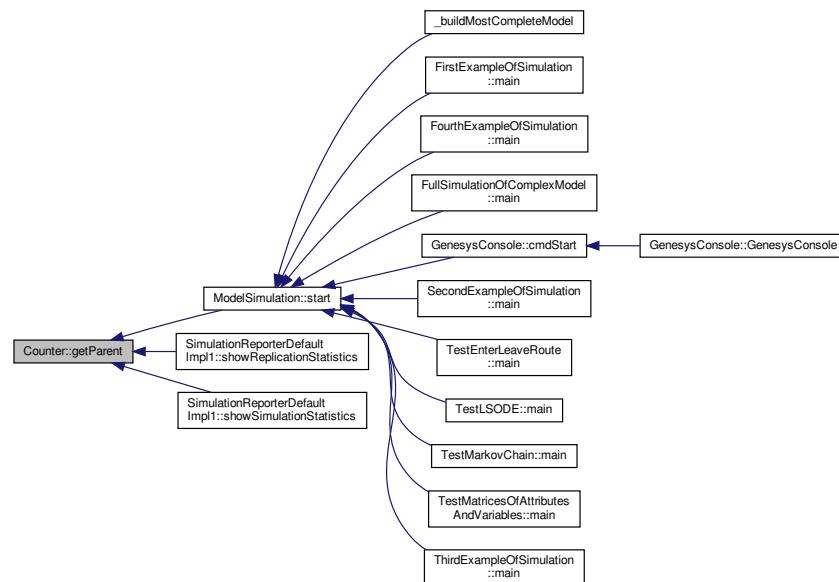


6.18.3.7 getParent()

```
ModelElement * Counter::getParent () const
```

Definition at line 51 of file [Counter.cpp](#).

Here is the caller graph for this function:

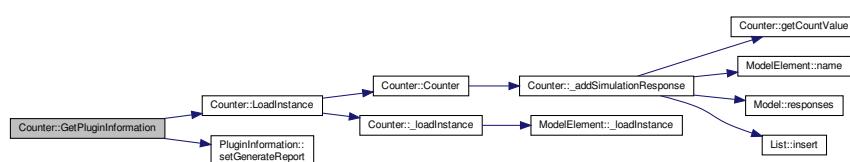


6.18.3.8 GetPluginInformation()

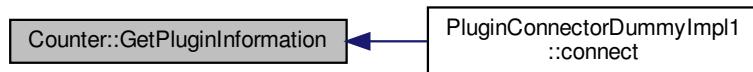
```
PluginInformation * Counter::GetPluginInformation () [static]
```

Definition at line 55 of file [Counter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

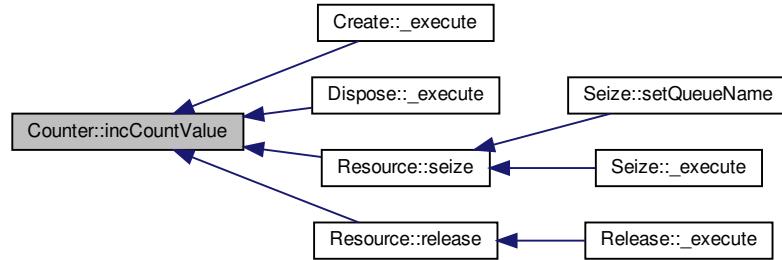


6.18.3.9 incCountValue()

```
void Counter::incCountValue (
    int value = 1 )
```

Definition at line 43 of file [Counter.cpp](#).

Here is the caller graph for this function:

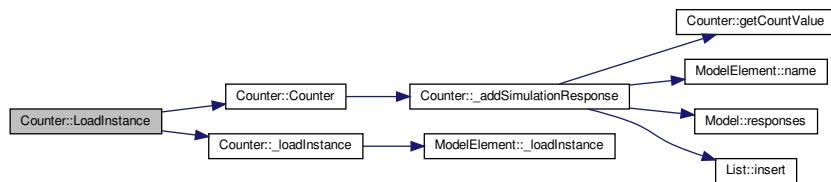


6.18.3.10 LoadInstance()

```
ModelElement * Counter::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 61 of file [Counter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.3.11 show()

```
std::string Counter::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [33](#) of file [Counter.cpp](#).

Here is the call graph for this function:



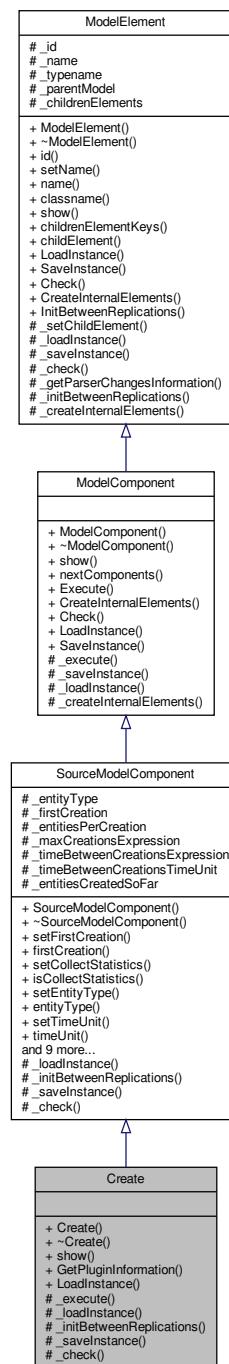
The documentation for this class was generated from the following files:

- [Counter.h](#)
- [Counter.cpp](#)

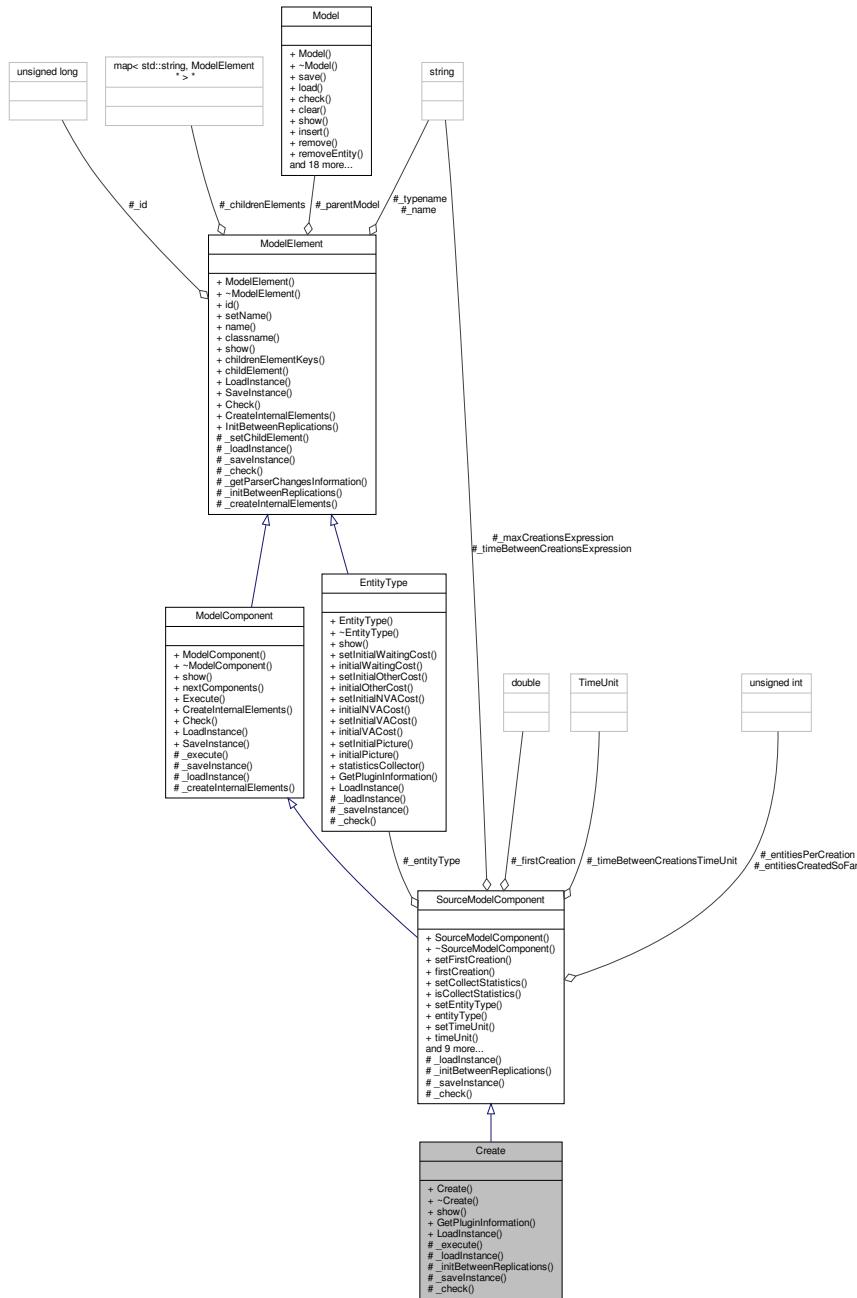
6.19 Create Class Reference

```
#include <Create.h>
```

Inheritance diagram for Create:



Collaboration diagram for Create:



Public Member Functions

- **Create** (`Model *model, std::string name=""`)
- virtual **~Create** ()=default
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void [_execute \(Entity *entity\)](#)
- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual void [_initBetweenReplications \(\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)

Additional Inherited Members

6.19.1 Detailed Description

Create is the most basic component to include the first entities into the model, and therefore is a source component (derived from [SourceModelComponent](#)) **Create** module DESCRIPTION This module is intended as the starting point for entities in a simulation model. Entities are created using a schedule or based on a time between arrivals. Entities then leave the module to begin processing through the system. The entity type is specified in this module. TYPIC← AL USES The start of a part's production in a manufacturing line A document's arrival (for example, order, check, application) into a business process A customer's arrival at a service process (for example, retail store, restaurant, information desk) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Entity Type Name of the entity type to be generated. Type Type of arrival stream to be generated. Types include Random (uses an exponential distribution, user specifies mean), Schedule (uses an exponential distribution, mean determined from the specified [Schedule](#) module), Constant (user specifies constant value; for example, 100), or Expression (drop-down list of various distributions). Value Determines the mean of the exponential distribution (if Random is used) or the constant value (if Constant is used) for the time between arrivals. Applies only when Type is Random or Constant. Schedule Name Identifies the name of the schedule to be used. The schedule defines the arrival pattern for entities arriving to the system. Applies only when Type is [Schedule](#). Expression Any distribution or value specifying the time between arrivals. Applies only when Type is Expression. Units Time units used for interarrival and first creation times. Does not apply when Type is [Schedule](#). Entities per Arrival Number of entities that will enter the system at a given time with each arrival. Max Arrivals Maximum number of entities that this module will generate. When this value is reached, the creation of new entities by this module ceases. First Creation Starting time for the first entity to arrive into the system. Does not apply when Type is [Schedule](#).

Definition at line [68](#) of file [Create.h](#).

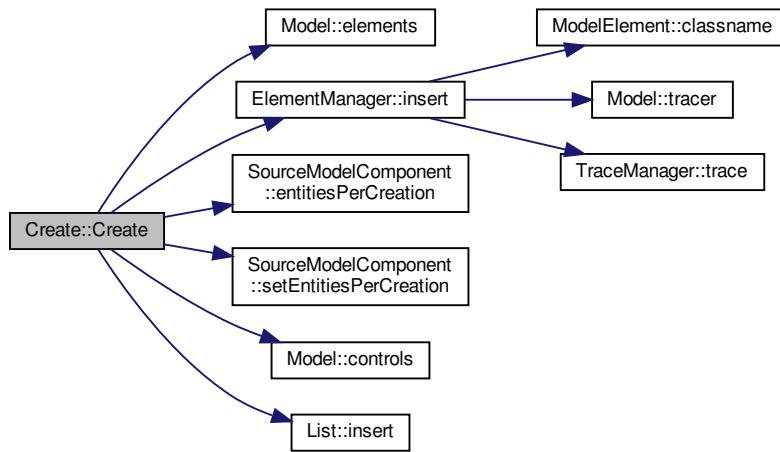
6.19.2 Constructor & Destructor Documentation

6.19.2.1 Create()

```
Create::Create (
    Model * model,
    std::string name = "")
```

Definition at line [21](#) of file [Create.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.19.2.2 ~Create()

```
virtual Create::~Create ( ) [virtual], [default]
```

6.19.3 Member Function Documentation

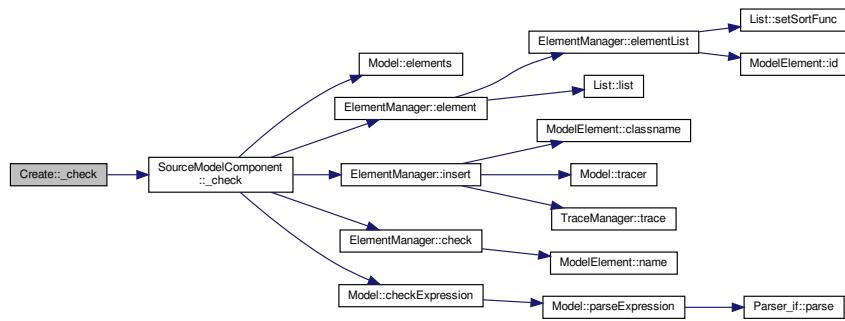
6.19.3.1 _check()

```
bool Create::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [SourceModelComponent](#).

Definition at line 97 of file [Create.cpp](#).

Here is the call graph for this function:



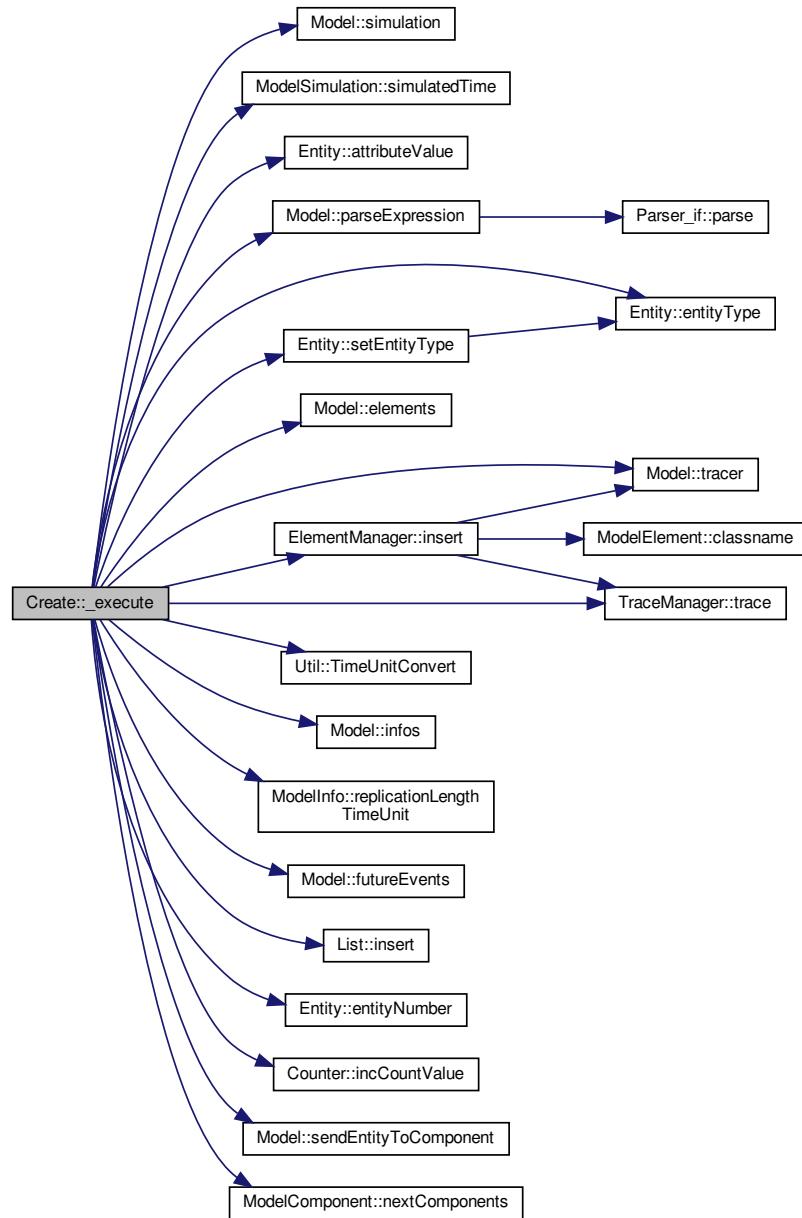
6.19.3.2 `_execute()`

```
void Create::_execute (
    Entity * entity )  [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line [40](#) of file [Create.cpp](#).

Here is the call graph for this function:



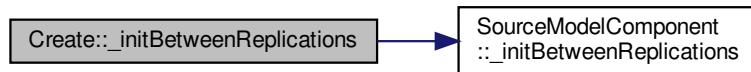
6.19.3.3 `_initBetweenReplications()`

```
void Create::_initBetweenReplications( ) [protected], [virtual]
```

Reimplemented from [SourceModelComponent](#).

Definition at line 88 of file [Create.cpp](#).

Here is the call graph for this function:



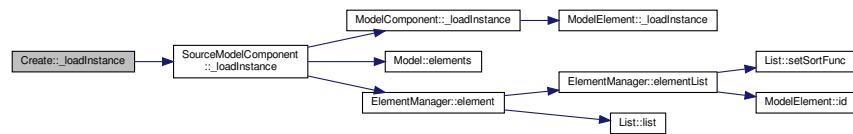
6.19.3.4 _loadInstance()

```
bool Create::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

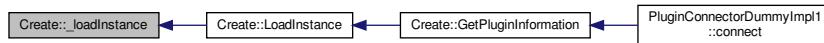
Reimplemented from [SourceModelComponent](#).

Definition at line 84 of file [Create.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



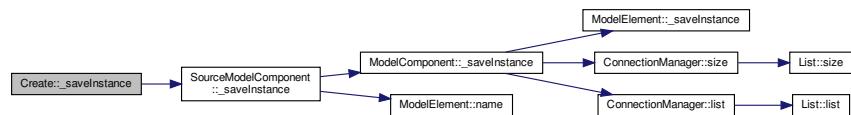
6.19.3.5 _saveInstance()

```
std::map< std::string, std::string * > * Create::_saveInstance () [protected], [virtual]
```

Reimplemented from [SourceModelComponent](#).

Definition at line 92 of file [Create.cpp](#).

Here is the call graph for this function:

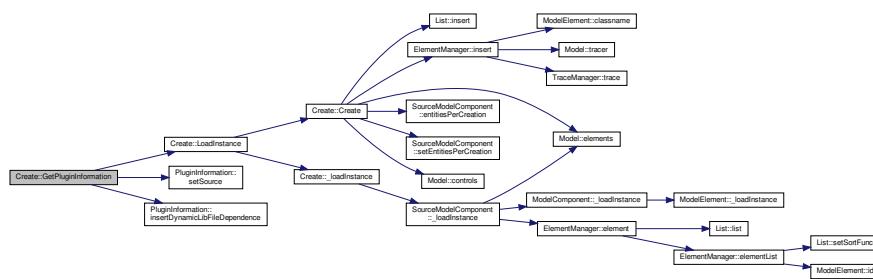


6.19.3.6 GetPluginInformation()

```
PluginInformation * Create::GetPluginInformation() [static]
```

Definition at line 65 of file [Create.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

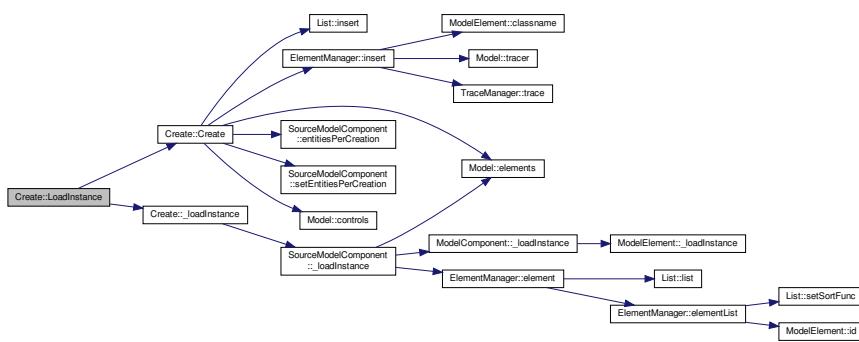


6.19.3.7 LoadInstance()

```
ModelComponent * Create::LoadInstance(
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 74 of file [Create.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



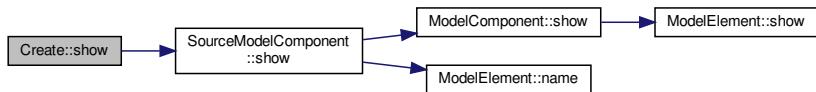
6.19.3.8 show()

```
std::string Create::show ( ) [virtual]
```

Reimplemented from [SourceModelComponent](#).

Definition at line 36 of file [Create.cpp](#).

Here is the call graph for this function:



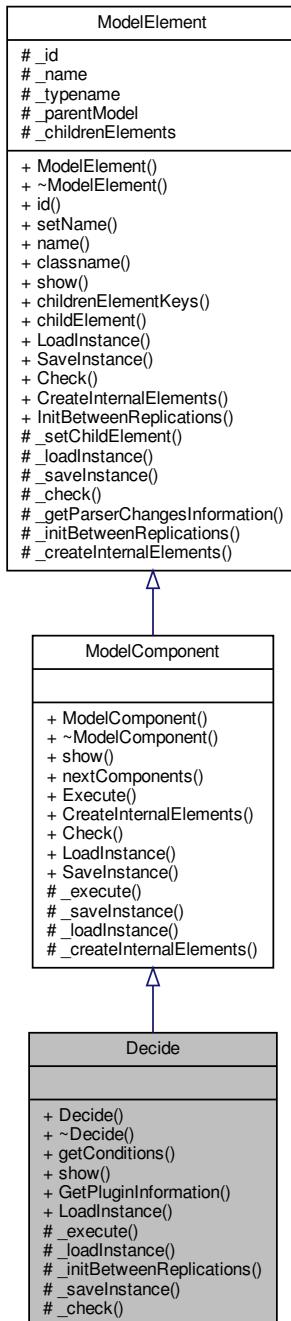
The documentation for this class was generated from the following files:

- Create.h
 - Create.cpp

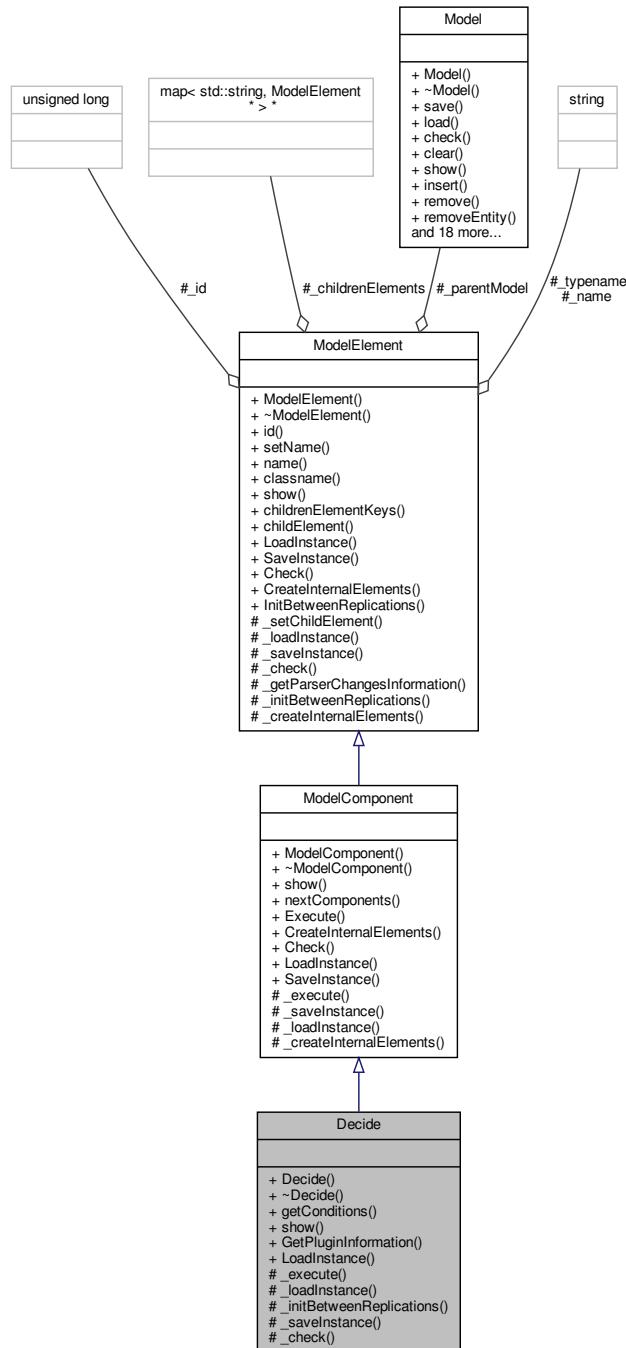
6.20 Decide Class Reference

```
#include <Decide.h>
```

Inheritance diagram for Decide:



Collaboration diagram for Decide:



Public Member Functions

- `Decide (Model *model, std::string name="")`
- virtual `~Decide ()=default`
- `List< std::string > * getConditions () const`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.20.1 Detailed Description

Decide module DESCRIPTION This module allows for decision-making processes in the system. It includes options to make decisions based on one or more conditions (for example, if entity type is Gold Card) or based on one or more probabilities (for example, 75%, true; 25%, false). Conditions can be based on attribute values (for example, Priority), variable values (for example, Number Denied), the entity type, or an expression (for example, NQ(ProcessA.Queue)). There are two exit points out of the **Decide** module when its specified type is either 2-way by Chance or 2-way by Condition. There is one exit point for “true” entities and one for “false” entities. When the N-way by Chance or by Condition type is specified, multiple exit points are shown for each condition or probability and a single “else” exit. The number of entities that exit from each type (true/false) is displayed for 2-way by Chance or by Condition modules only. TYPICAL USES Dispatching a faulty part for rework Branching accepted vs. rejected checks Sending priority customers to a dedicated process Prompt Description Name Unique module identifier displayed on the module shape. Type Indicates whether the decision is based on a condition (if X>Y) or by chance/percentage (for example, 60%, yes; 40%, no). The type can be specified as either 2-way or N-way. 2-way allows for one condition or probability (plus the “false” exit). N-way allows for any number of conditions or probabilities to be specified as well as an “else” exit. Conditions Defines one or more conditions used to direct entities to different modules. Applies only when Type is N-way by Condition. Percentages Defines one or more percentages used to direct entities to different modules. Applies only when Type is N-way by Chance. Percent True Value that will be checked to determine the percentage of entities sent out a given True exit. If Types of conditions that are available for evaluation: **Variable**, **Variable Array (1D)**, **Variable Array (2D)**, **Attribute**, **Entity Type**, **Expression**. Named Specifies the name of the variable, attribute, or entity type that will be evaluated when an entity enters the module. Does not apply when Type is Expression. Is Evaluator for the condition. Applies only to **Attribute** and **Variable** conditions. Row Specifies the row index for a variable array. Applies only when Type is N-way by Condition or 2-way by Condition and **Variable** is Array 1-D or Array 2-D. Column Specifies the column index for a variable array. Applies only when Type is N-way by Condition or 2-way by Condition and **Variable** is Array 1-D or Array 2-D. Value Expression that will be either compared to an attribute or variable or that will be evaluated as a single expression to determine if it is true or false. Does not apply to **Entity Type** condition. If Type is Expression, this value must also include the evaluator (for example, Color<>Red).

Definition at line 73 of file [Decide.h](#).

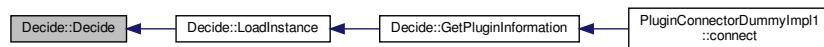
6.20.2 Constructor & Destructor Documentation

6.20.2.1 Decide()

```
Decide::Decide (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Decide.cpp](#).

Here is the caller graph for this function:



6.20.2.2 ~Decide()

```
virtual Decide::~Decide () [virtual], [default]
```

6.20.3 Member Function Documentation

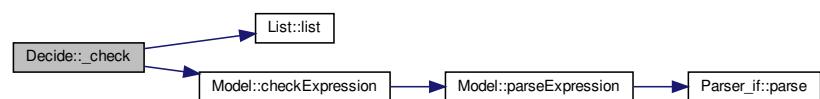
6.20.3.1 _check()

```
bool Decide::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 69 of file [Decide.cpp](#).

Here is the call graph for this function:



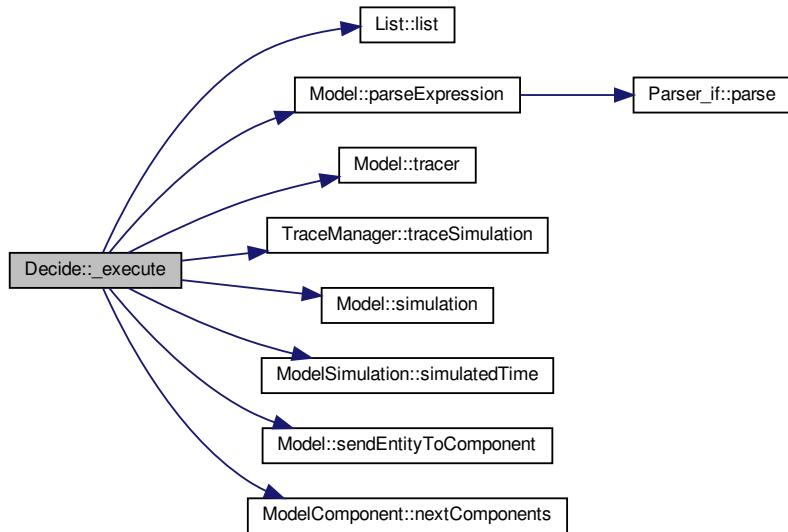
6.20.3.2 `_execute()`

```
void Decide::_execute (
    Entity * entity )  [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 29 of file [Decide.cpp](#).

Here is the call graph for this function:



6.20.3.3 `_initBetweenReplications()`

```
void Decide::_initBetweenReplications ( )  [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 45 of file [Decide.cpp](#).

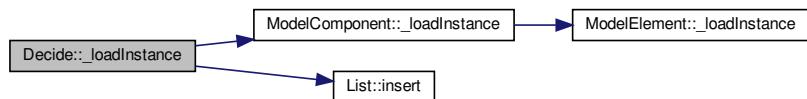
6.20.3.4 `_loadInstance()`

```
bool Decide::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 48 of file [Decide.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



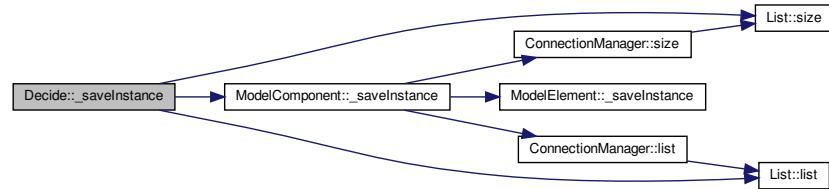
6.20.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Decide::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 59 of file [Decide.cpp](#).

Here is the call graph for this function:

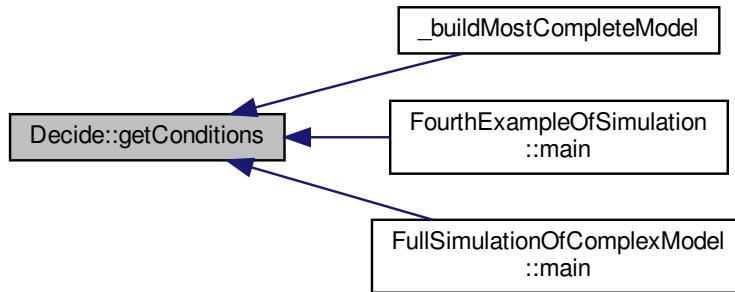


6.20.3.6 getConditions()

```
List< std::string > * Decide::getConditions ( ) const
```

Definition at line 20 of file [Decide.cpp](#).

Here is the caller graph for this function:

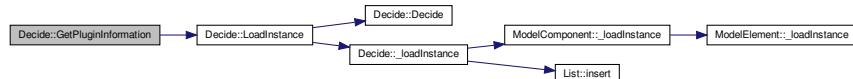


6.20.3.7 GetPluginInformation()

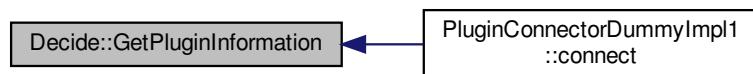
```
PluginInformation * Decide::GetPluginInformation ( ) [static]
```

Definition at line 79 of file [Decide.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

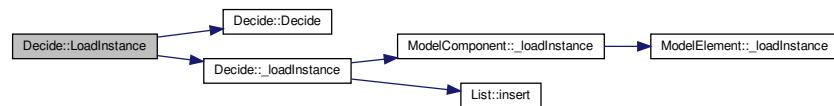


6.20.3.8 LoadInstance()

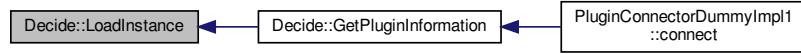
```
ModelComponent * Decide::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 84 of file [Decide.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



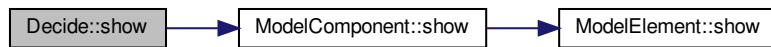
6.20.3.9 show()

```
std::string Decide::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 25 of file [Decide.cpp](#).

Here is the call graph for this function:



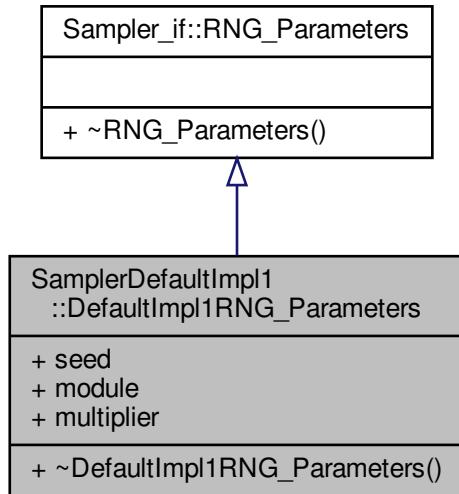
The documentation for this class was generated from the following files:

- [Decide.h](#)
- [Decide.cpp](#)

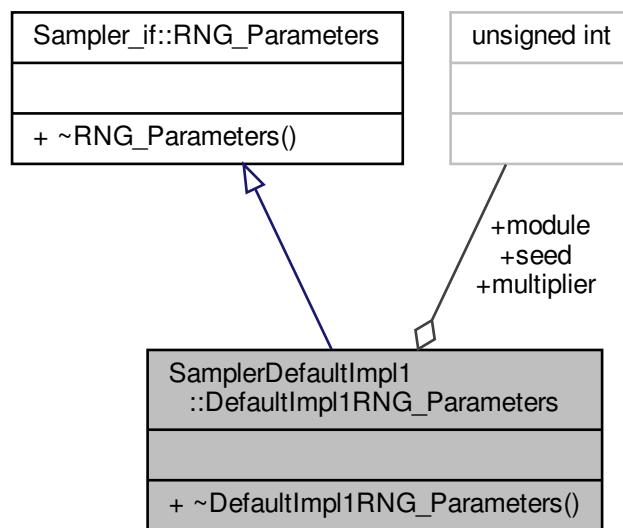
6.21 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Struct Reference

```
#include <SamplerDefaultImpl1.h>
```

Inheritance diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:



Collaboration diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:



Public Member Functions

- `~DefaultImpl1RNG_Parameters ()=default`

Public Attributes

- `unsigned int seed = 666`
- `unsigned int module = 2147483647`
- `unsigned int multiplier = 950706376`

6.21.1 Detailed Description

Definition at line 22 of file [SamplerDefaultImpl1.h](#).

6.21.2 Constructor & Destructor Documentation

6.21.2.1 `~DefaultImpl1RNG_Parameters()`

```
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::~DefaultImpl1RNG_Parameters ( ) [default]
```

6.21.3 Member Data Documentation

6.21.3.1 `module`

```
unsigned int SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::module = 2147483647
```

Definition at line 24 of file [SamplerDefaultImpl1.h](#).

6.21.3.2 `multiplier`

```
unsigned int SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::multiplier = 950706376
```

Definition at line 25 of file [SamplerDefaultImpl1.h](#).

6.21.3.3 `seed`

```
unsigned int SamplerDefaultImpl1::DefaultImpl1RNG_Parameters::seed = 666
```

Definition at line 23 of file [SamplerDefaultImpl1.h](#).

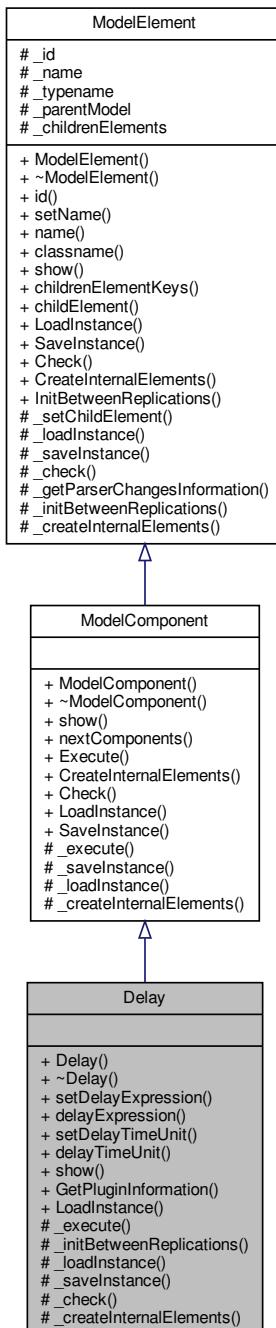
The documentation for this struct was generated from the following file:

- [SamplerDefaultImpl1.h](#)

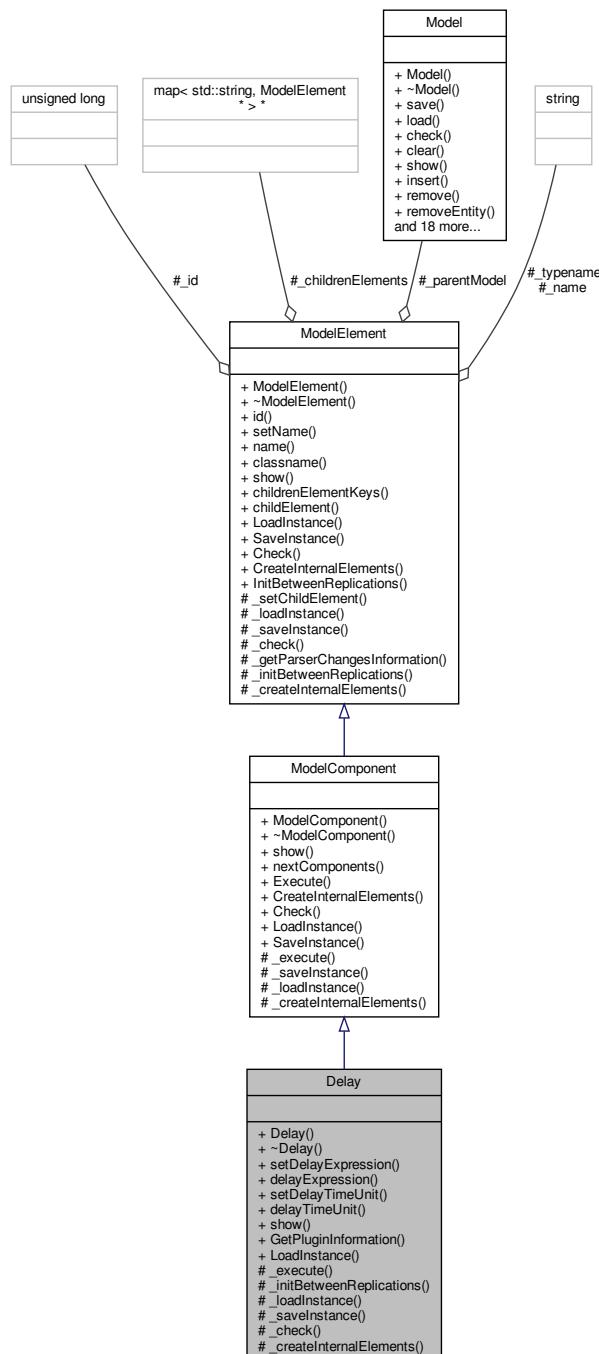
6.22 Delay Class Reference

```
#include <Delay.h>
```

Inheritance diagram for Delay:



Collaboration diagram for Delay:



Public Member Functions

- `Delay (Model *model, std::string name="")`
- `virtual ~Delay ()=default`
- `void setDelayExpression (std::string _delayExpression)`
- `std::string delayExpression () const`
- `void setDelayTimeUnit (Util::TimeUnit _delayTimeUnit)`
- `Util::TimeUnit delayTimeUnit () const`
- `virtual std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

6.22.1 Detailed Description

Delay module DESCRIPTION The **Delay** module delays an entity by a specified amount of time. When an entity arrives at a **Delay** module, the time delay expression is evaluated and the entity remains in the module for the resulting time period. The time is then allocated to the entity's value-added, non-value added, transfer, wait, or other time. Associated costs are calculated and allocated as well. TYPICAL USES Processing a check at a bank Performing a setup on a machine Transferring a document to another department PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Allocation Type of category to which the entity's incurred delay time and cost will be added. **Delay** Time Determines the value of the delay for the entity. Units Time units used for the delay time.

Definition at line 41 of file `Delay.h`.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Delay()

```
Delay::Delay (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file `Delay.cpp`.

Here is the caller graph for this function:



6.22.2.2 ~Delay()

```
virtual Delay::~Delay ( ) [virtual], [default]
```

6.22.3 Member Function Documentation

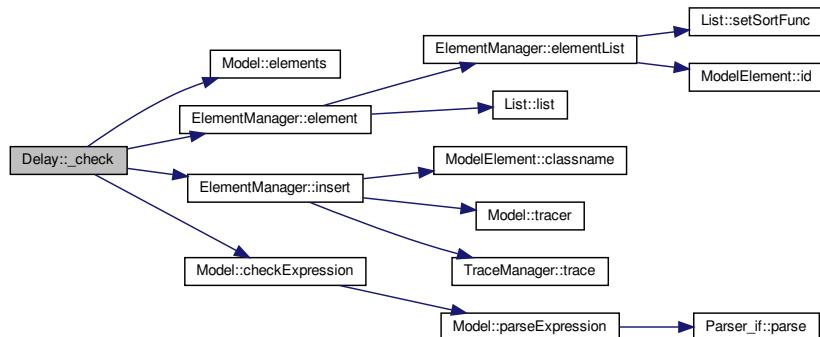
6.22.3.1 _check()

```
bool Delay::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 82 of file [Delay.cpp](#).

Here is the call graph for this function:



6.22.3.2 _createInternalElements()

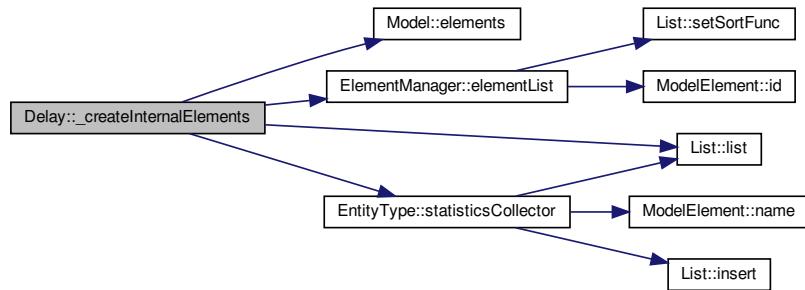
```
void Delay::_createInternalElements ( ) [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelComponent](#).

Definition at line 97 of file [Delay.cpp](#).

Here is the call graph for this function:



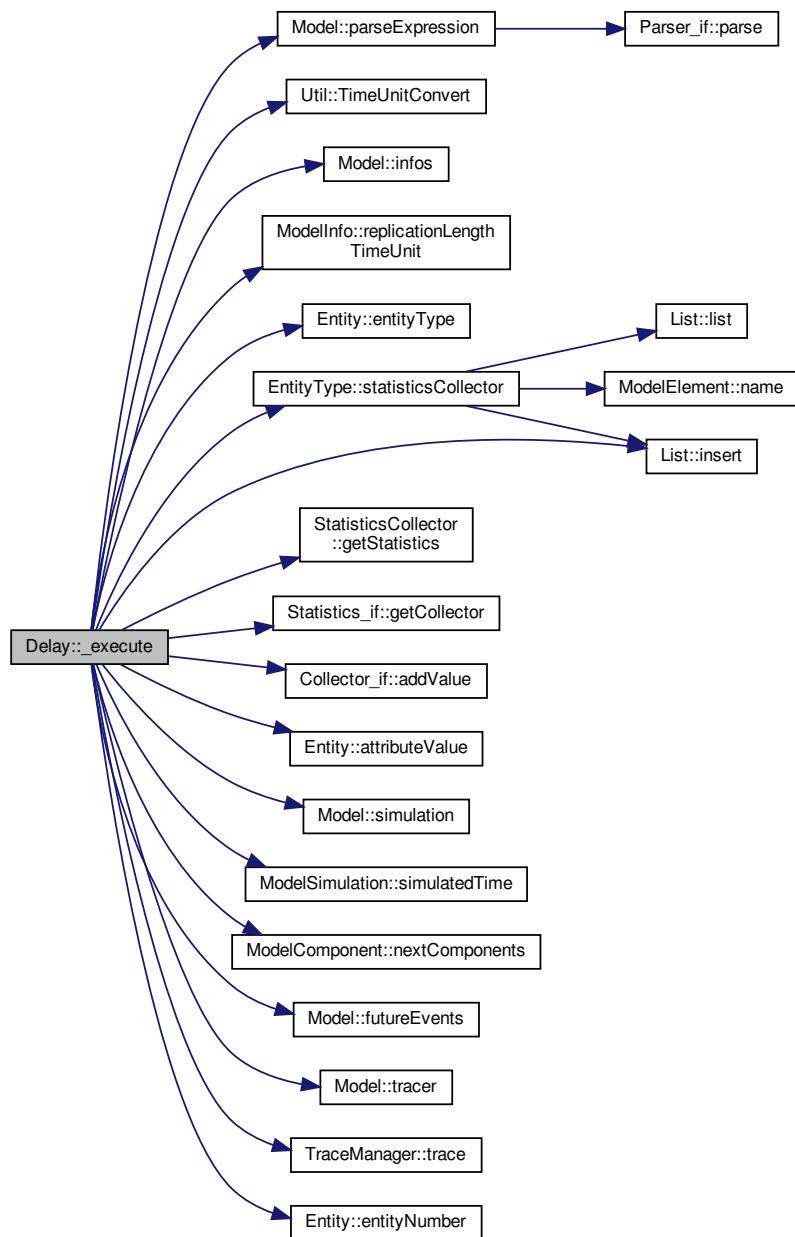
6.22.3.3 `_execute()`

```
void Delay::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 53 of file [Delay.cpp](#).

Here is the call graph for this function:



6.22.3.4 `_initBetweenReplications()`

```
void Delay::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 72 of file [Delay.cpp](#).

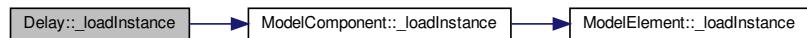
6.22.3.5 `_loadInstance()`

```
bool Delay::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 63 of file [Delay.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



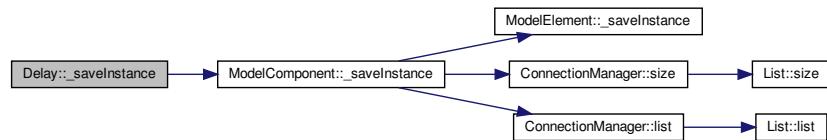
6.22.3.6 `_saveInstance()`

```
std::map< std::string, std::string > * Delay::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 75 of file [Delay.cpp](#).

Here is the call graph for this function:



6.22.3.7 delayExpression()

```
std::string Delay::delayExpression ( ) const
```

Definition at line 41 of file [Delay.cpp](#).

6.22.3.8 delayTimeUnit()

```
Util::TimeUnit Delay::delayTimeUnit ( ) const
```

Definition at line 49 of file [Delay.cpp](#).

6.22.3.9 GetPluginInformation()

```
PluginInformation * Delay::GetPluginInformation ( ) [static]
```

Definition at line 107 of file [Delay.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.3.10 LoadInstance()

```
ModelComponent * Delay::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 27 of file [Delay.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

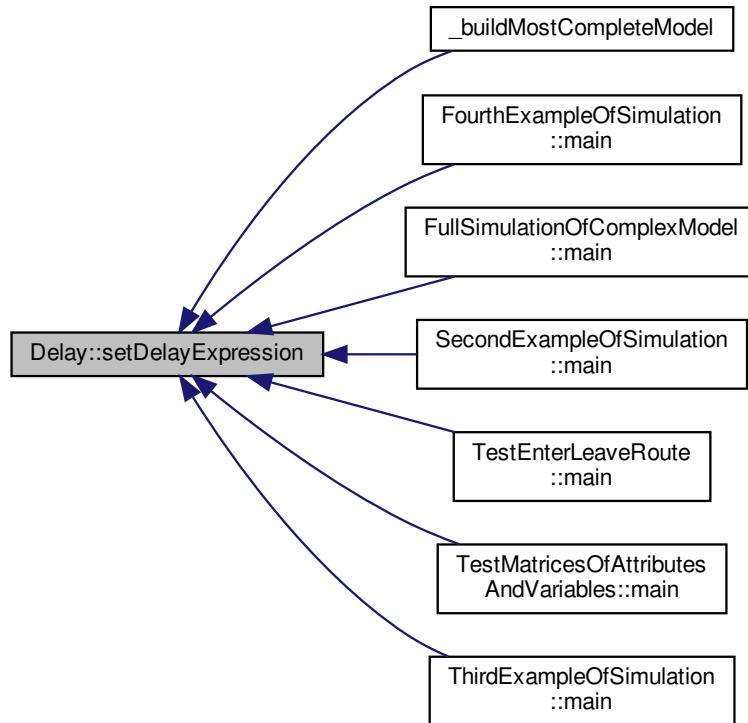


6.22.3.11 setDelayExpression()

```
void Delay::setDelayExpression (
    std::string _delayExpression )
```

Definition at line 37 of file [Delay.cpp](#).

Here is the caller graph for this function:

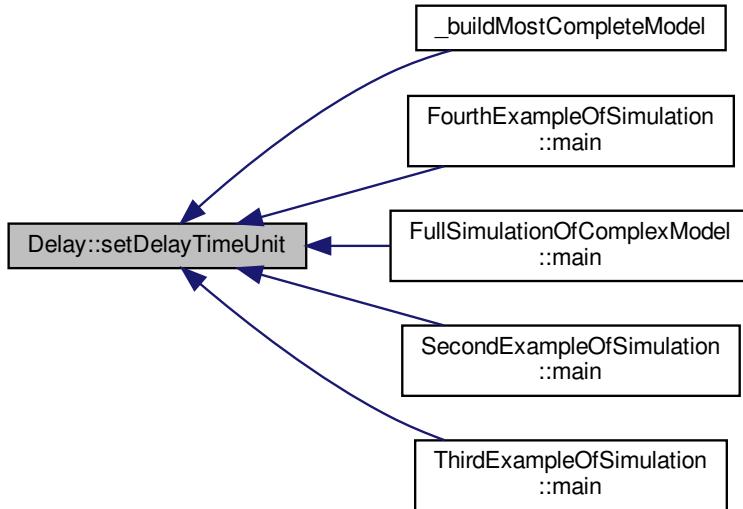


6.22.3.12 setDelayTimeUnit()

```
void Delay::setDelayTimeUnit (
    Util::TimeUnit _delayTimeUnit )
```

Definition at line 45 of file [Delay.cpp](#).

Here is the caller graph for this function:



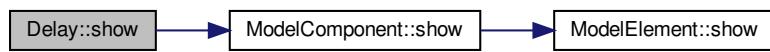
6.22.3.13 show()

```
std::string Delay::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [21](#) of file [Delay.cpp](#).

Here is the call graph for this function:



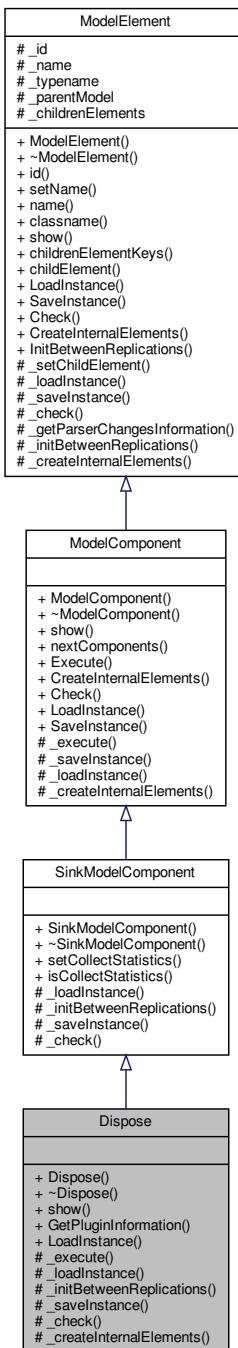
The documentation for this class was generated from the following files:

- [Delay.h](#)
- [Delay.cpp](#)

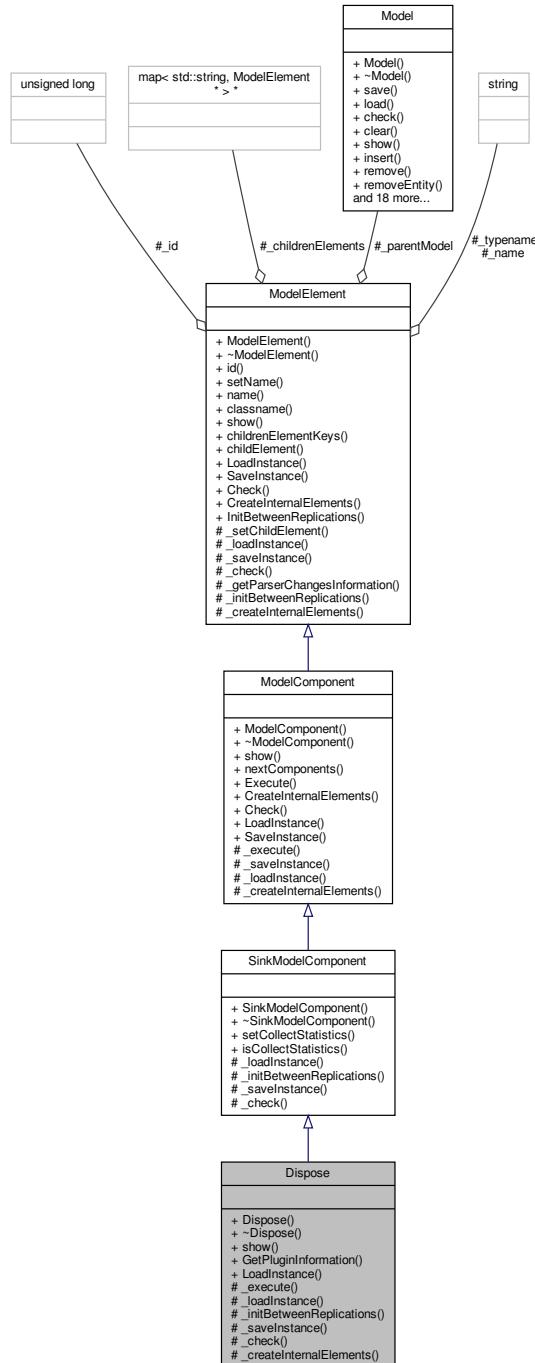
6.23 Dispose Class Reference

```
#include <Dispose.h>
```

Inheritance diagram for Dispose:



Collaboration diagram for Dispose:



Public Member Functions

- **Dispose** (`Model *model, std::string name=""`)
- virtual **~Dispose** ()=default
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

6.23.1 Detailed Description

Dispose module DESCRIPTION This module is intended as the ending point for entities in a simulation model. Entity statistics may be recorded before the entity is disposed of. TYPICAL USES Parts leaving the modeled facility The termination of a business process Customers departing from the store Prompt Description Name Unique module identifier displayed on the module shape. **Record Entity Statistics** Determines whether or not the incoming entity's statistics will be recorded. Statistics include value-added time, non-value-added time, wait time, transfer time, other time, total time, value-added cost, non-value-added cost, wait cost, transfer cost, other cost, and total cost.

Definition at line 38 of file `Dispose.h`.

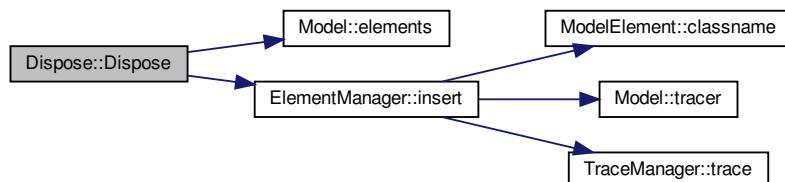
6.23.2 Constructor & Destructor Documentation

6.23.2.1 Dispose()

```
Dispose::Dispose (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file `Dispose.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.23.2.2 ~Dispose()

```
virtual Dispose::~Dispose ( ) [virtual], [default]
```

6.23.3 Member Function Documentation

6.23.3.1 _check()

```
bool Dispose::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [SinkModelComponent](#).

Definition at line 52 of file [Dispose.cpp](#).

6.23.3.2 _createInternalElements()

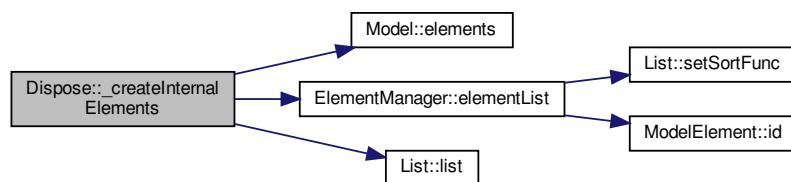
```
void Dispose::_createInternalElements ( ) [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelComponent](#).

Definition at line 57 of file [Dispose.cpp](#).

Here is the call graph for this function:



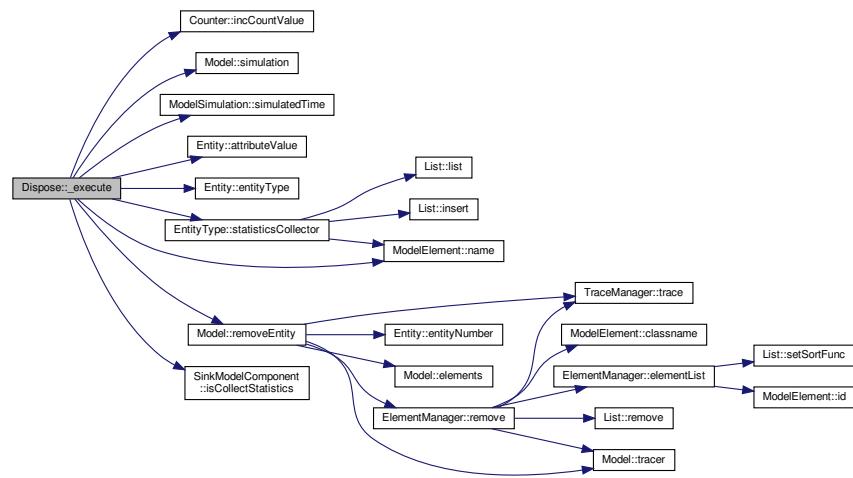
6.23.3.3 `_execute()`

```
void Dispose::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line [28](#) of file [Dispose.cpp](#).

Here is the call graph for this function:



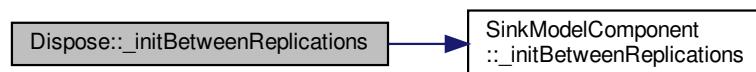
6.23.3.4 `_initBetweenReplications()`

```
void Dispose::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [SinkModelComponent](#).

Definition at line [42](#) of file [Dispose.cpp](#).

Here is the call graph for this function:



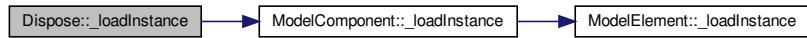
6.23.3.5 `_loadInstance()`

```
bool Dispose::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [SinkModelComponent](#).

Definition at line 38 of file [Dispose.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



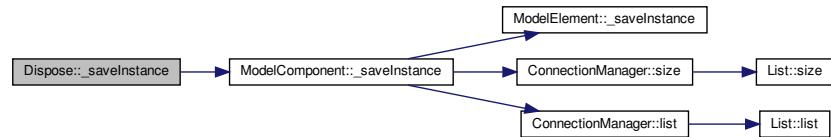
6.23.3.6 `_saveInstance()`

```
std::map< std::string, std::string > * Dispose::_saveInstance () [protected], [virtual]
```

Reimplemented from [SinkModelComponent](#).

Definition at line 46 of file [Dispose.cpp](#).

Here is the call graph for this function:

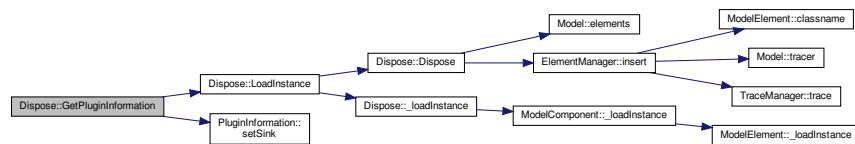


6.23.3.7 GetPluginInformation()

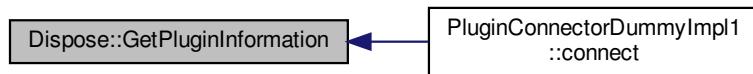
```
PluginInformation * Dispose::GetPluginInformation ( ) [static]
```

Definition at line 65 of file [Dispose.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

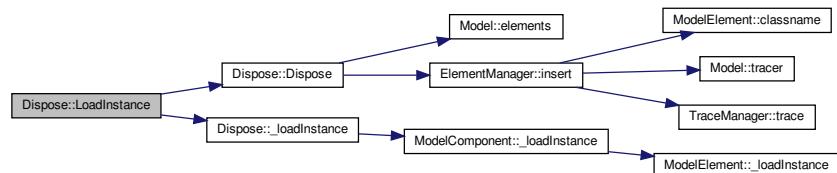


6.23.3.8 LoadInstance()

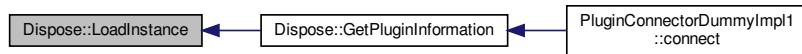
```
ModelComponent * Dispose::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 71 of file [Dispose.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.23.3.9 show()

```
std::string Dispose::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [23](#) of file [Dispose.cpp](#).

Here is the call graph for this function:



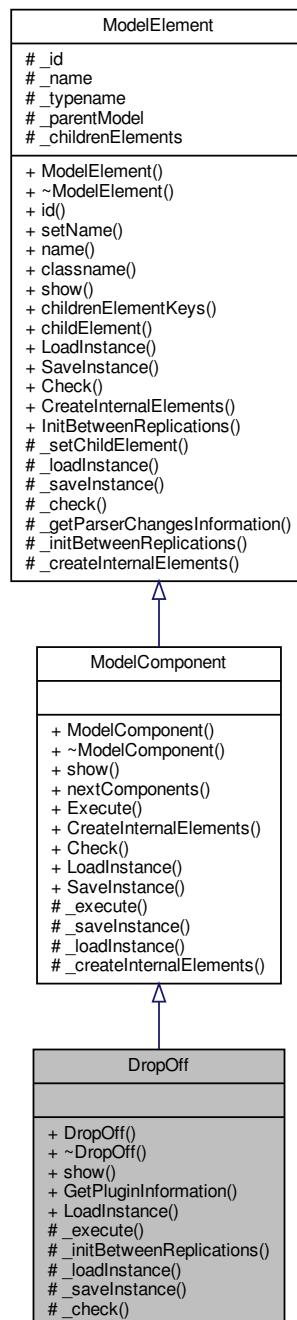
The documentation for this class was generated from the following files:

- [Dispose.h](#)
- [Dispose.cpp](#)

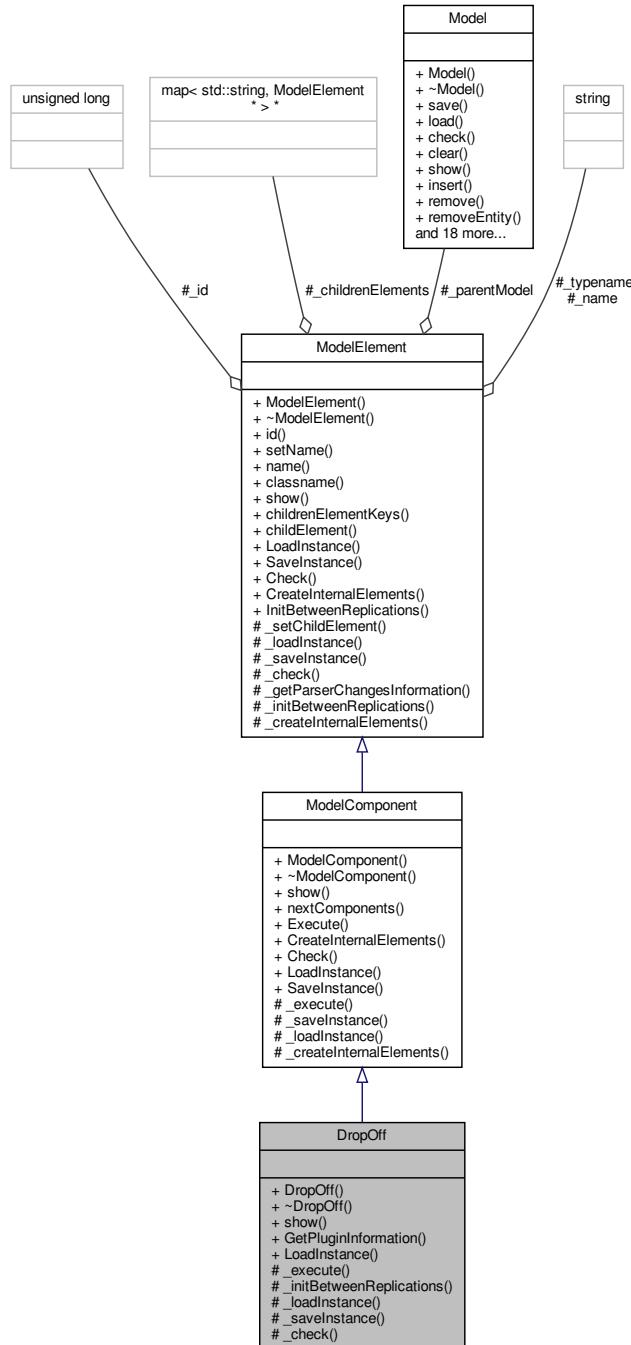
6.24 DropOff Class Reference

```
#include <DropOff.h>
```

Inheritance diagram for DropOff:



Collaboration diagram for DropOff:



Public Member Functions

- `DropOff (Model *model, std::string name="")`
- virtual `~DropOff ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.24.1 Detailed Description

Dropoff module DESCRIPTION The Dropoff module removes a specified number of entities from the entity's group and sends them to another module, as specified by a graphical connection. Group user-defined attribute value and internal attributes may be given to the dropped-off entities based on a specified rule. **TYPICAL USES** Loading shelves with product Separating a form for use in various departments **PROMPTS** Prompt Description Name Unique module identifier displayed on the module shape. Quantity Number of entities that will be dropped off from an incoming representative grouped entity. Starting Rank Starting rank of the entities to be dropped off, based on the entities in the group. Member Attributes Method of determining how to assign the representative entity attribute values (other than costs/times) to the dropped-off original entities. **Attribute** Name Name of representative entity attribute(s) assigned to droppedoff original entities of the group

Definition at line 41 of file `DropOff.h`.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 DropOff()

```
DropOff::DropOff (
    Model * model,
    std::string name = "")
```

Definition at line 17 of file `DropOff.cpp`.

Here is the caller graph for this function:



6.24.2.2 ~DropOff()

```
virtual DropOff::~DropOff ( ) [virtual], [default]
```

6.24.3 Member Function Documentation

6.24.3.1 _check()

```
bool DropOff::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [57](#) of file [DropOff.cpp](#).

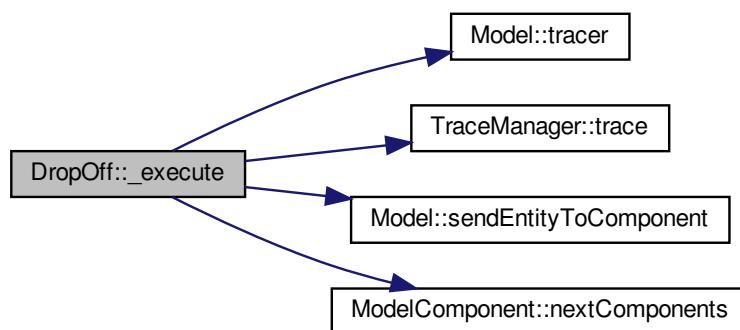
6.24.3.2 _execute()

```
void DropOff::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line [35](#) of file [DropOff.cpp](#).

Here is the call graph for this function:



6.24.3.3 `_initBetweenReplications()`

```
void DropOff::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [DropOff.cpp](#).

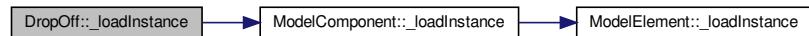
6.24.3.4 `_loadInstance()`

```
bool DropOff::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

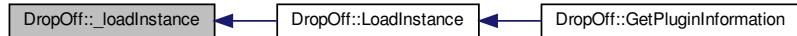
Reimplemented from [ModelComponent](#).

Definition at line 40 of file [DropOff.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



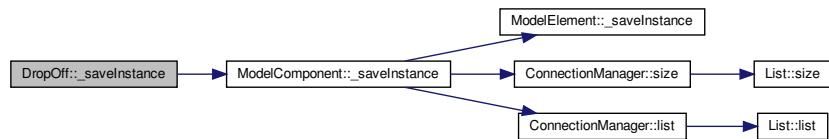
6.24.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * DropOff::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 51 of file [DropOff.cpp](#).

Here is the call graph for this function:

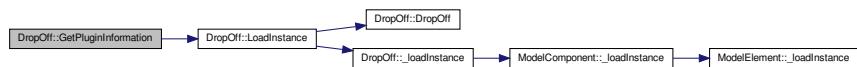


6.24.3.6 GetPluginInformation()

```
PluginInformation * DropOff::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [DropOff.cpp](#).

Here is the call graph for this function:

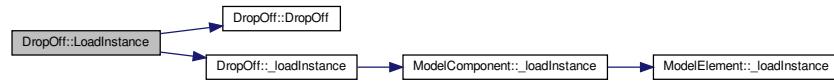


6.24.3.7 LoadInstance()

```
ModelComponent * DropOff::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 25 of file [DropOff.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



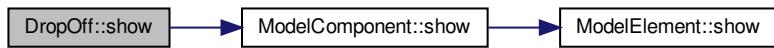
6.24.3.8 show()

```
std::string DropOff::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [DropOff.cpp](#).

Here is the call graph for this function:



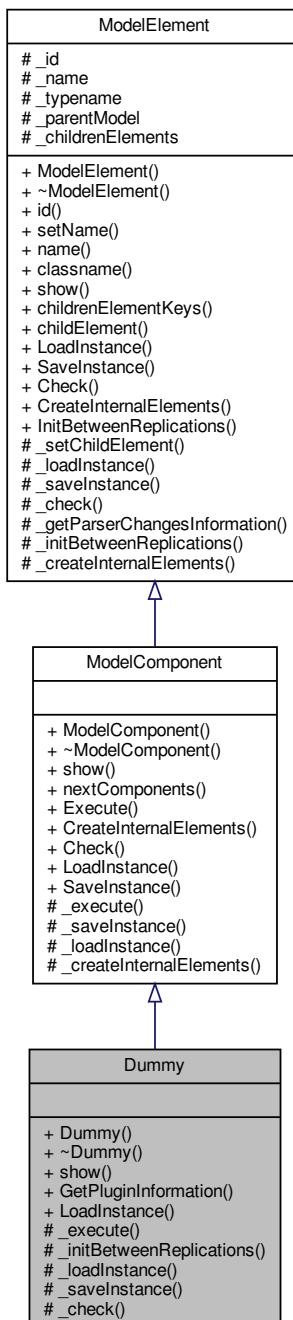
The documentation for this class was generated from the following files:

- [DropOff.h](#)
- [DropOff.cpp](#)

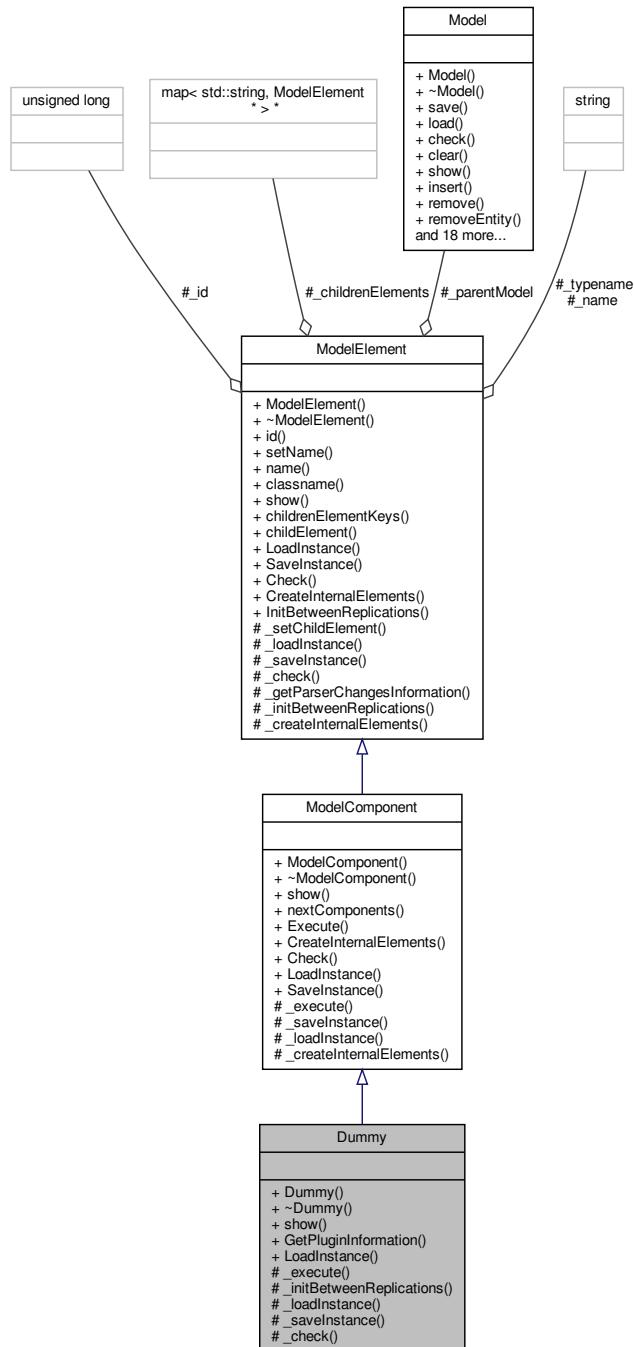
6.25 Dummy Class Reference

```
#include <Dummy.h>
```

Inheritance diagram for Dummy:



Collaboration diagram for Dummy:



Public Member Functions

- **Dummy** (`Model *model, std::string name="")`
 - virtual ~**Dummy** ()=default
 - virtual std::string **show** ()

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.25.1 Detailed Description

This component ...

Definition at line 22 of file [Dummy.h](#).

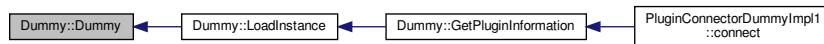
6.25.2 Constructor & Destructor Documentation

6.25.2.1 Dummy()

```
Dummy::Dummy (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Dummy.cpp](#).

Here is the caller graph for this function:



6.25.2.2 ~Dummy()

```
virtual Dummy::~Dummy ( ) [virtual], [default]
```

6.25.3 Member Function Documentation

6.25.3.1 `_check()`

```
bool Dummy::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Dummy.cpp](#).

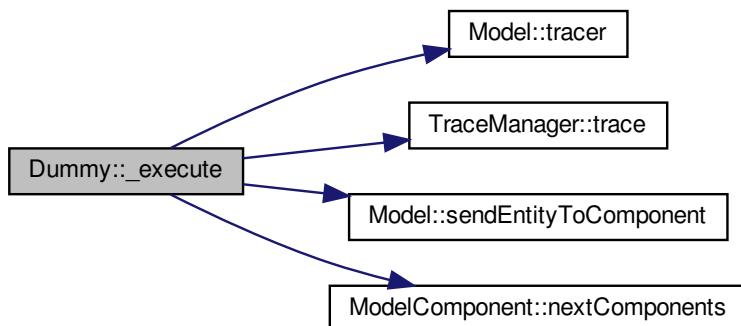
6.25.3.2 `_execute()`

```
void Dummy::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Dummy.cpp](#).

Here is the call graph for this function:



6.25.3.3 `_initBetweenReplications()`

```
void Dummy::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Dummy.cpp](#).

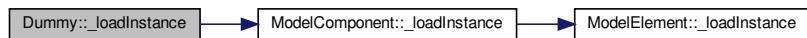
6.25.3.4 `_loadInstance()`

```
bool Dummy::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 40 of file [Dummy.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



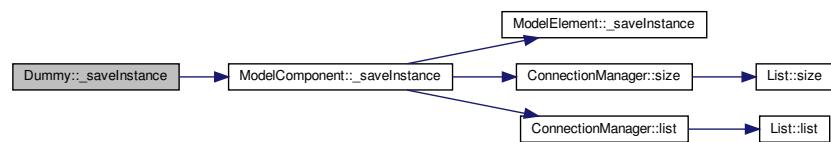
6.25.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Dummy::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Dummy.cpp](#).

Here is the call graph for this function:

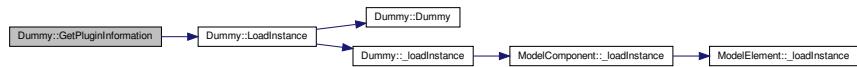


6.25.3.6 GetPluginInformation()

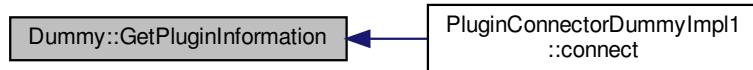
```
PluginInformation * Dummy::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [Dummy.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.3.7 LoadInstance()

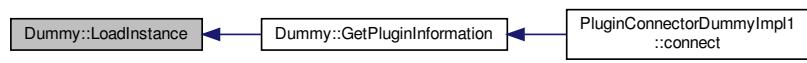
```
ModelComponent * Dummy::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Dummy.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



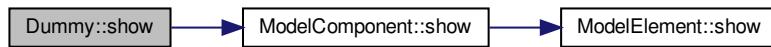
6.25.3.8 show()

```
std::string Dummy::show () [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Dummy.cpp](#).

Here is the call graph for this function:



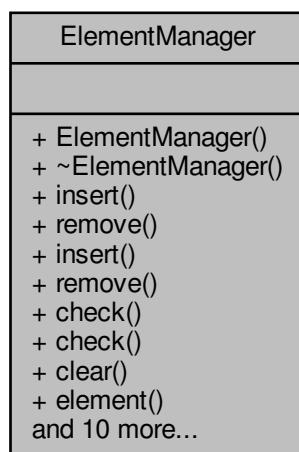
The documentation for this class was generated from the following files:

- [Dummy.h](#)
- [Dummy.cpp](#)

6.26 ElementManager Class Reference

```
#include <ElementManager.h>
```

Collaboration diagram for ElementManager:



Public Member Functions

- `ElementManager (Model *model)`
- `virtual ~ElementManager ()=default`
- `bool insert (ModelElement *infra)`
- `void remove (ModelElement *infra)`

Deprecated.
- `bool insert (std::string infraTypename, ModelElement *infra)`

Deprecated.
- `void remove (std::string infraTypename, ModelElement *infra)`
- `bool check (std::string infraTypename, ModelElement *infra, std::string expressionName, std::string *errorMessage)`
- `bool check (std::string infraTypename, std::string infraName, std::string expressionName, bool mandatory, std::string *errorMessage)`
- `void clear ()`
- `ModelElement * element (std::string infraTypename, Util::identification id)`
- `ModelElement * element (std::string infraTypename, std::string name)`
- `unsigned int numberOfElements (std::string infraTypename)`
- `unsigned int numberOfElements ()`
- `int rankOf (std::string infraTypename, std::string name)`

returns the position (1st position=0) of the element if found, or negative value if not found
- `std::list< std::string > * elementClassnames () const`
- `List< ModelElement * > * elementList (std::string infraTypename) const`
- `void show ()`
- `Model * parentModel () const`
- `bool hasChanged () const`
- `void setHasChanged (bool _hasChanged)`

6.26.1 Detailed Description

The `ElementManager` is responsible for inserting and removing elements (`ModelElement`) used by components, in a consistent way. TO FIX: No direct access for insertion or deletion should be allowed

Definition at line 29 of file `ElementManager.h`.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 ElementManager()

```
ElementManager::ElementManager (
    Model * model )
```

Elements are organized as a map from a string (key), the type of an element, and a list of elements of that type

Definition at line 17 of file `ElementManager.cpp`.

6.26.2.2 ~ElementManager()

```
virtual ElementManager::~ElementManager ( ) [virtual], [default]
```

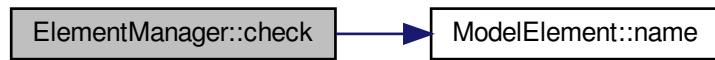
6.26.3 Member Function Documentation

6.26.3.1 check() [1/2]

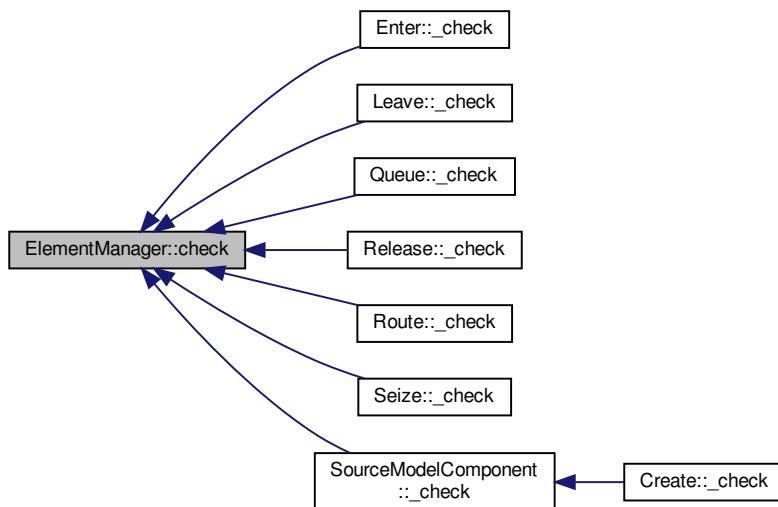
```
bool ElementManager::check (
    std::string infraTypename,
    ModelElement * infra,
    std::string expressionName,
    std::string * errorMessage )
```

Definition at line 72 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

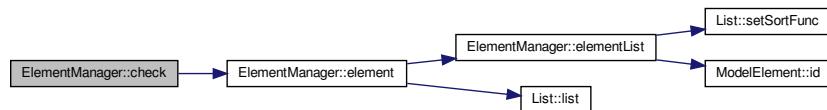


6.26.3.2 check() [2/2]

```
bool ElementManager::check (
    std::string infraTypename,
    std::string infraName,
    std::string expressionName,
    bool mandatory,
    std::string * errorMessage )
```

Definition at line 60 of file [ElementManager.cpp](#).

Here is the call graph for this function:

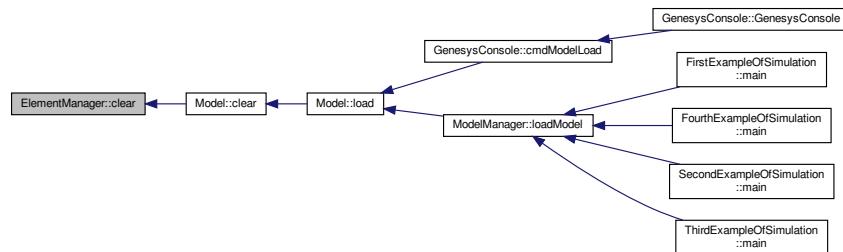


6.26.3.3 clear()

```
void ElementManager::clear ( )
```

Definition at line 83 of file [ElementManager.cpp](#).

Here is the caller graph for this function:

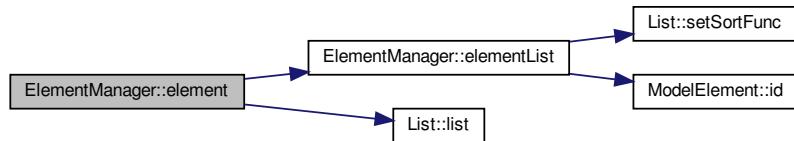


6.26.3.4 element() [1/2]

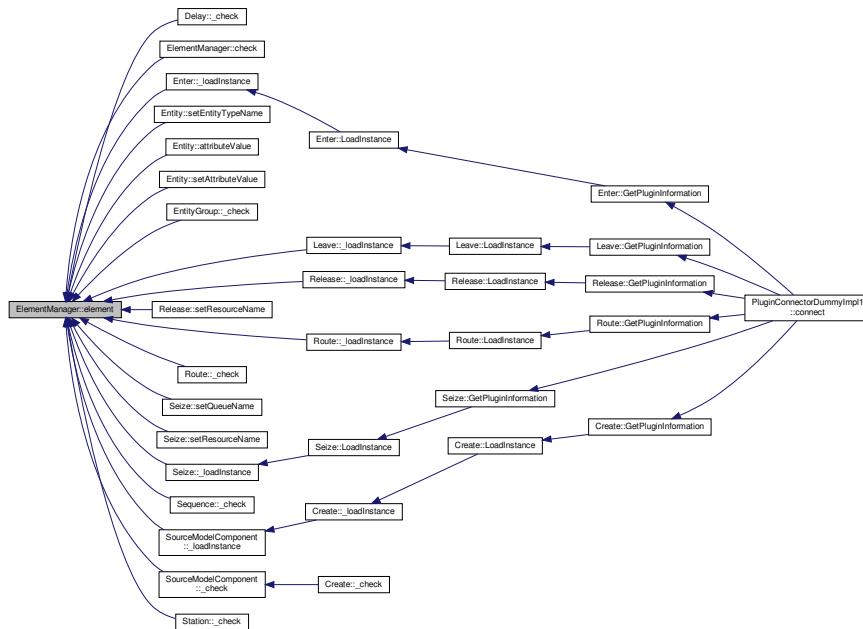
```
ModelElement * ElementManager::element (
    std::string infraTypename,
    Util::identification id )
```

Definition at line 150 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

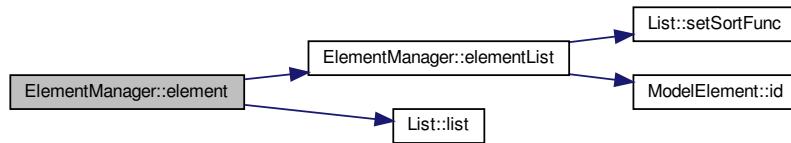


6.26.3.5 element() [2/2]

```
ModelElement * ElementManager::element (
    std::string infraTypename,
    std::string name )
```

Definition at line 181 of file [ElementManager.cpp](#).

Here is the call graph for this function:

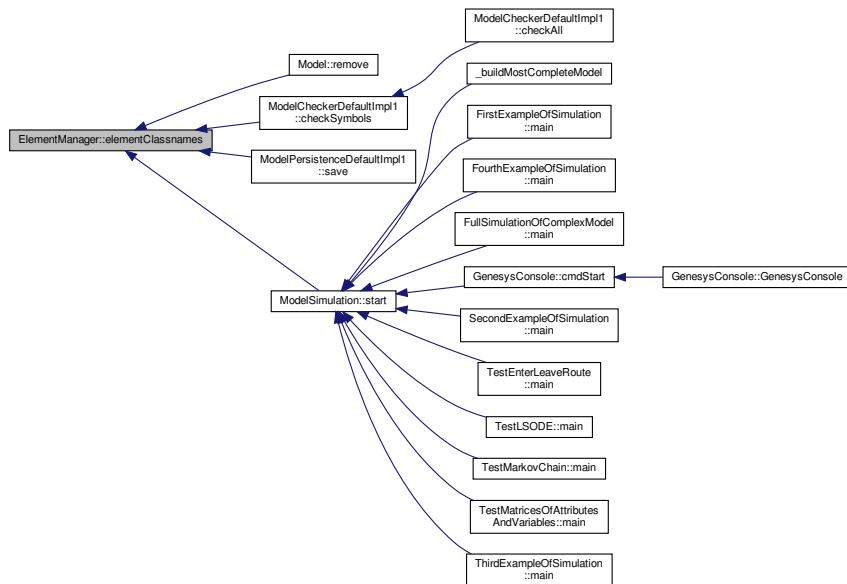


6.26.3.6 elementClassnames()

```
std::list< std::string > * ElementManager::elementClassnames ( ) const
```

Definition at line 173 of file [ElementManager.cpp](#).

Here is the caller graph for this function:

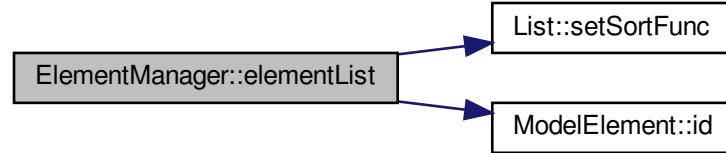


6.26.3.7 elementList()

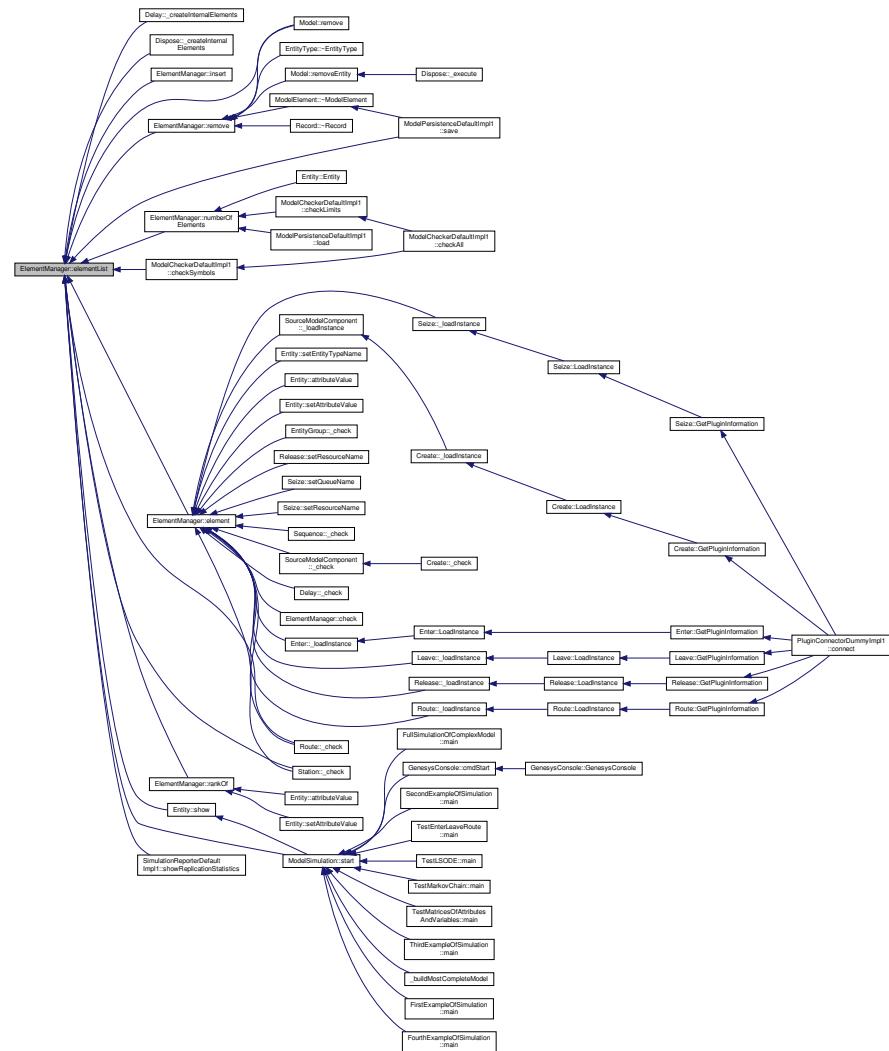
```
List< ModelElement * > * ElementManager::elementList (
    std::string infraTypename ) const
```

Definition at line 135 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3.8 hasChanged()

```
bool ElementManager::hasChanged ( ) const
```

Definition at line 127 of file [ElementManager.cpp](#).

Here is the caller graph for this function:

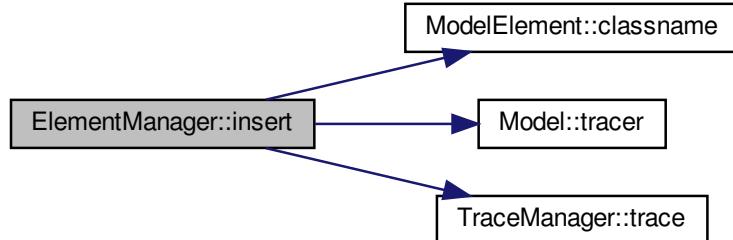


6.26.3.9 insert() [1/2]

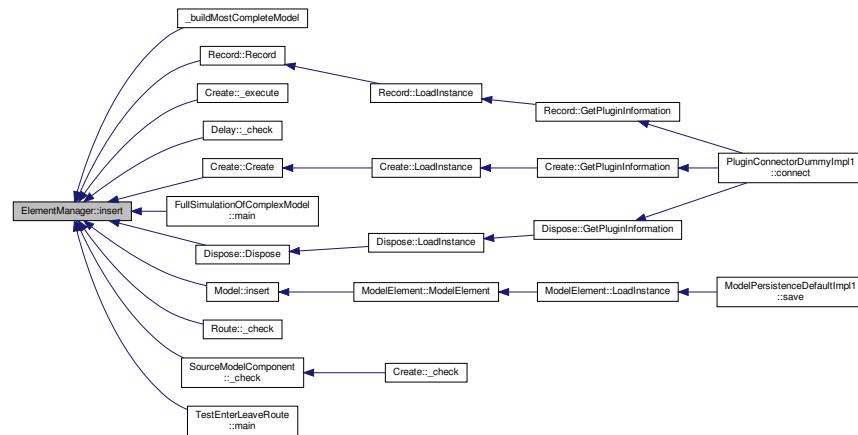
```
bool ElementManager::insert ( ModelElement * infra )
```

Definition at line 27 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



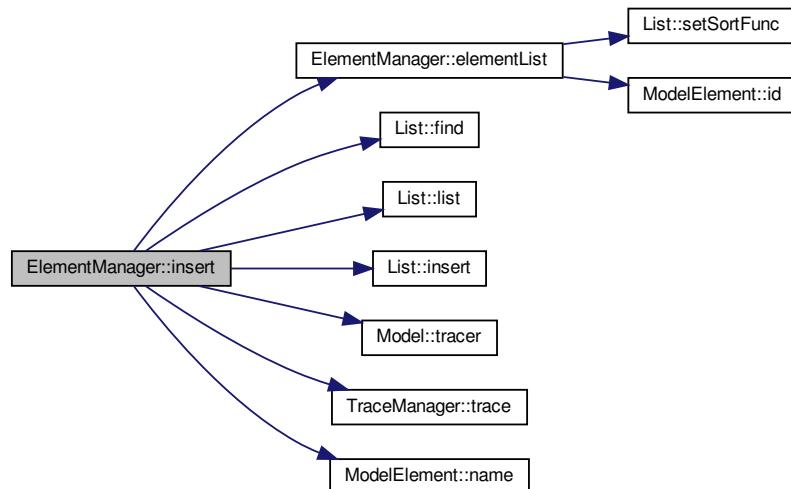
6.26.3.10 insert() [2/2]

```
bool ElementManager::insert (
    std::string infraTypename,
    ModelElement * infra )
```

Deprecated.

Definition at line 37 of file [ElementManager.cpp](#).

Here is the call graph for this function:

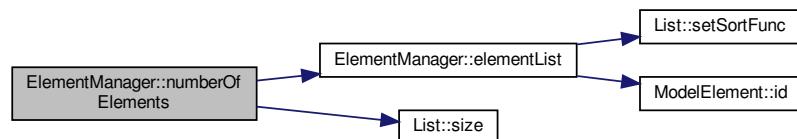


6.26.3.11 `numberOfElements()` [1/2]

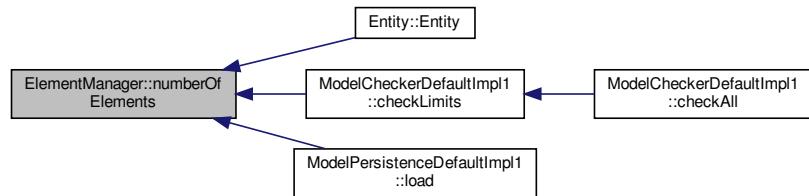
```
unsigned int ElementManager::numberOfElements (
    std::string infraTypename )
```

Definition at line 87 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

6.26.3.12 `numberOfElements()` [2/2]

```
unsigned int ElementManager::numberOfElements ( )
```

Definition at line 92 of file [ElementManager.cpp](#).

6.26.3.13 `parentModel()`

```
Model * ElementManager::parentModel ( ) const
```

Definition at line 123 of file [ElementManager.cpp](#).

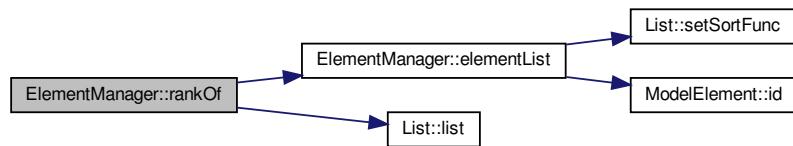
6.26.3.14 rankOf()

```
int ElementManager::rankOf (
    std::string infraTypename,
    std::string name )
```

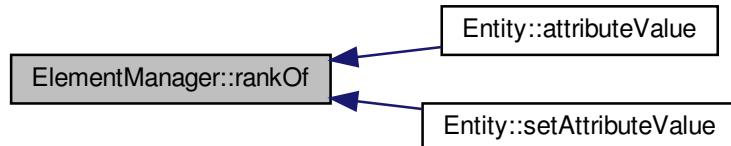
returns the position (1st position=0) of the element if found, or negative value if not found

Definition at line 160 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



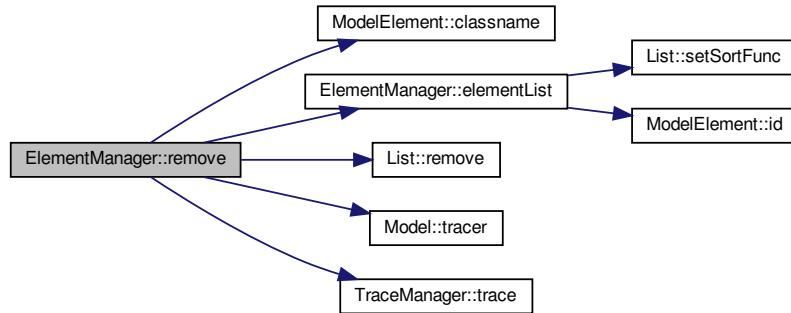
6.26.3.15 remove() [1/2]

```
void ElementManager::remove (
    ModelElement * infra )
```

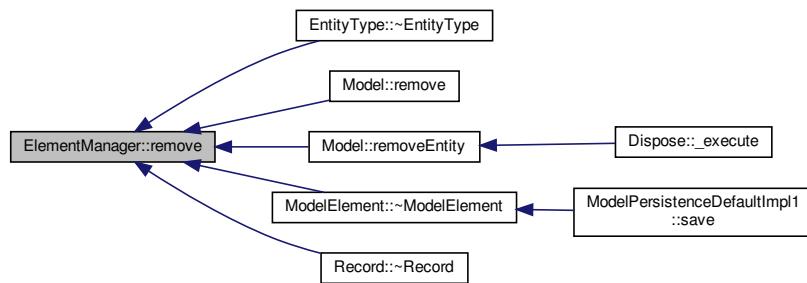
Deprecated.

Definition at line 47 of file [ElementManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

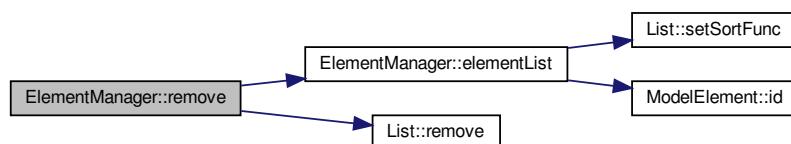


6.26.3.16 remove() [2/2]

```
void ElementManager::remove (
    std::string infraTypename,
    ModelElement * infra )
```

Definition at line 55 of file [ElementManager.cpp](#).

Here is the call graph for this function:



6.26.3.17 setHasChanged()

```
void ElementManager::setHasChanged ( bool _hasChanged )
```

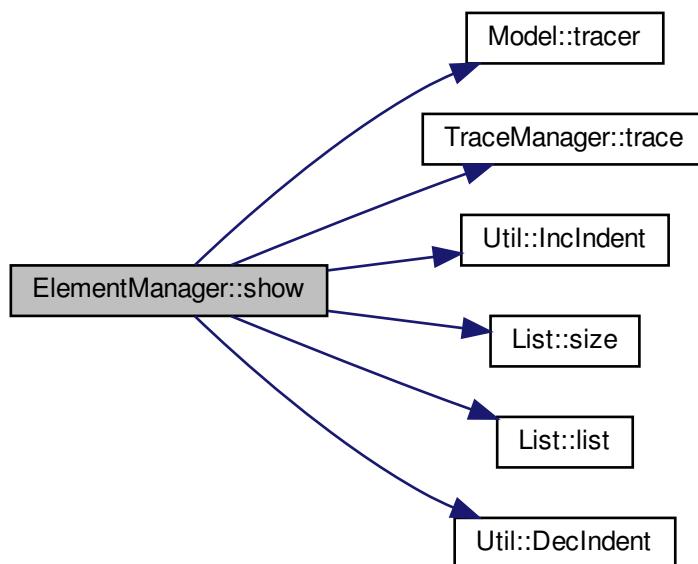
Definition at line 131 of file [ElementManager.cpp](#).

6.26.3.18 show()

```
void ElementManager::show ( )
```

Definition at line 100 of file [ElementManager.cpp](#).

Here is the call graph for this function:



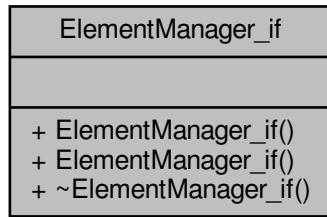
The documentation for this class was generated from the following files:

- [ElementManager.h](#)
- [ElementManager.cpp](#)

6.27 ElementManager_if Class Reference

```
#include <ElementManager_if.h>
```

Collaboration diagram for ElementManager_if:



Public Member Functions

- [ElementManager_if \(\)](#)
- [ElementManager_if \(const ElementManager_if &orig\)](#)
- [virtual ~ElementManager_if \(\)](#)

6.27.1 Detailed Description

Definition at line 17 of file [ElementManager_if.h](#).

6.27.2 Constructor & Destructor Documentation

6.27.2.1 ElementManager_if() [1/2]

```
ElementManager_if::ElementManager_if ( )
```

6.27.2.2 ElementManager_if() [2/2]

```
ElementManager_if::ElementManager_if (
    const ElementManager_if & orig )
```

6.27.2.3 ~ElementManager_if()

```
virtual ElementManager_if::~ElementManager_if ( ) [virtual]
```

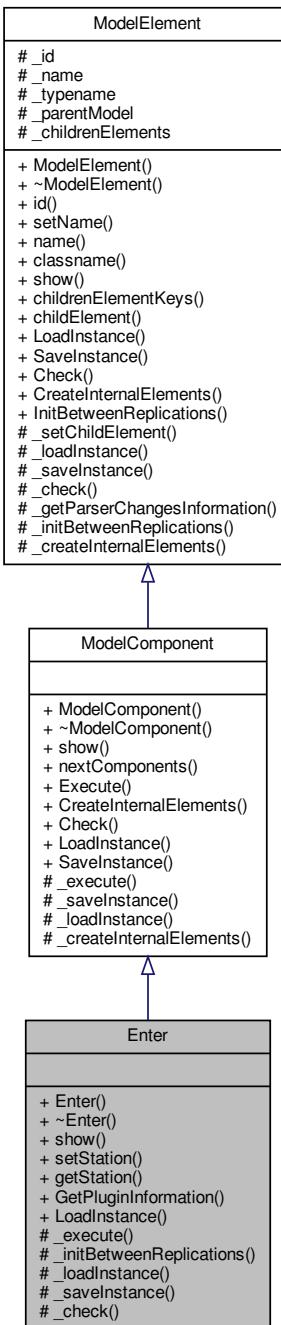
The documentation for this class was generated from the following file:

- [ElementManager_if.h](#)

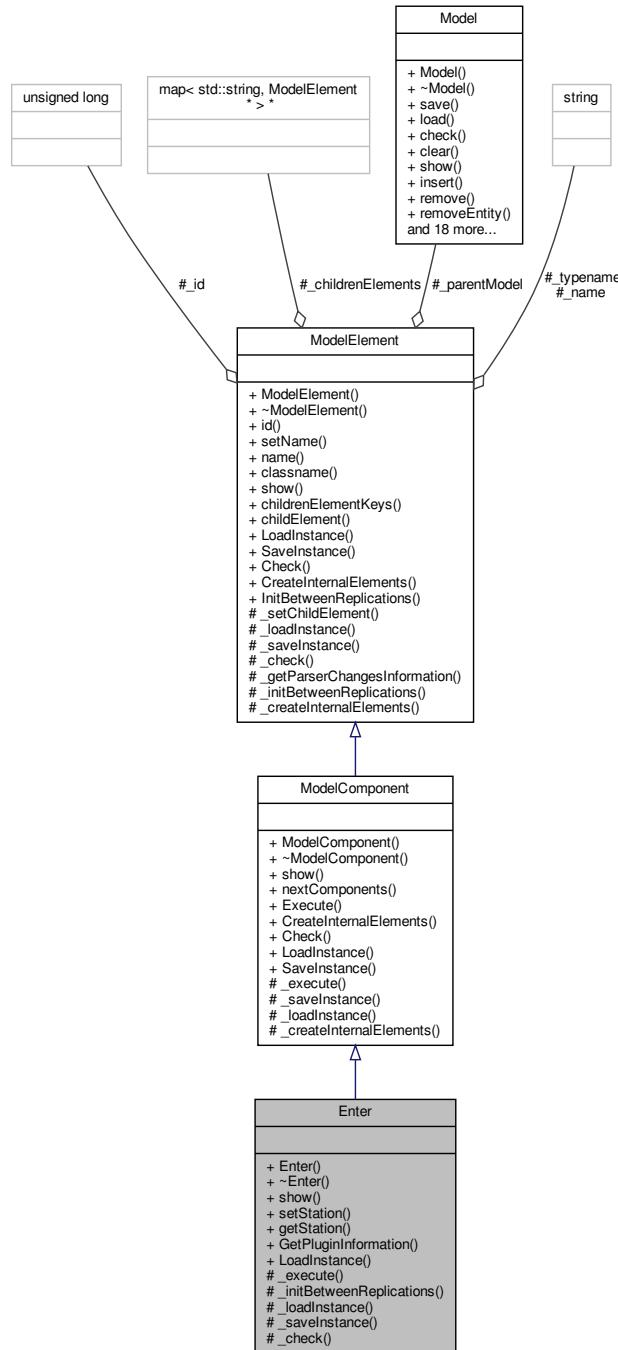
6.28 Enter Class Reference

```
#include <Enter.h>
```

Inheritance diagram for Enter:



Collaboration diagram for Enter:



Public Member Functions

- `Enter (Model *model, std::string name="")`
- `virtual ~Enter ()=default`
- `virtual std::string show ()`
- `void setStation (Station *_station)`
- `Station * getStation () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.28.1 Detailed Description

Enter module DESCRIPTION The `Enter` module defines a station (or a set of stations) corresponding to a physical or logical location where processing occurs. When an entity arrives at an `Enter` module, an unloading delay may occur and any transfer device used to transfer the entity to the `Enter` module's station may be released. The station (or each station within the defined set) has a matching Activity Area that is used to report all times and costs accrued by the entities in this station. This Activity Area's name is the same as the station. If a parent Activity Area is defined, then it also accrues any times and costs by the entities in this station. TYPICAL USES The start of a part's production in a series of parallel processes where the part's forklift needs to be released. The start of a document's processing after the document has been created where the mail clerk resource needs to be released. PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Station Type Type of station, either a single `Station` or station `Set`. Station Name Name of the individual station. A given station can only exist once within a model. Parent Activity Area Name of the Activity Area's parent. Associated Intersection Name of the intersection associated with this station in a guided transporter network. Report Statistics Specifies whether or not statistics will automatically be collected and stored in the report database for this station and its corresponding activity area. Set Name Name of the station set. A given station set can only exist once within a model. Save Attribute Specifies the attribute to be used to store the index into the station set for an entity entering this module. Set Members This repeat group permits you to define the individual stations that are to be members of the specified station set. A station set must have at least one member station. Active when Station Type is `Set`. Station Name This field indicates the name of a station that is to be a member of this station set. A given station can only exist within a model once. Therefore, an individual station can only be the member of one station set, and that individual station may not be the name of a station in another module. Parent Activity Area Name of the Activity Area's parent for the station set member. Associated Intersection Name of the intersection associated with this station set in a guided transporter network. Report Statistics Specifies whether or not statistics will automatically be collected and stored in the report database for this station set member and its corresponding activity area. Allocation Type of category to which the entity's incurred delay time and cost will be added. Delay This field defines the delay that will be experienced by entities immediately upon arrival at the station. Units Time units used for the delay time. Transfer In If a resource, transporter, or conveyor was used to transfer the entity to this station, this can be used to release, free, or exit the device. If `Release Resource` is selected, the specified resource is released. If `Free Transporter` is selected, the specified transporter is freed. If `Exit Conveyor` is selected, the specified conveyor is exited. Transporter Name Name of the transporter to be freed upon arrival to the station. Active when Transfer Name is `Free Transporter`. Unit Number Unit number of the transporter if the transporter is multicapacity. Conveyor Name Name of the conveyor to exit upon arrival to the station. Resource Type Type of allocation, either single `Resource` or resource `Set`. Resource Name Name of the resource to release. Active when Transfer Name is `Release Resource`. Set Name Name of the resource set from which the resource is to be released. Release Rule Determines which member of the set is to be released, either the Last Member Seized, First Member Seized, or Specific Member. Set Index Index into the set that determines which member of the set is to be released. Attribute Name Name of the attribute that determines the instance number of the resource to release. Expression Expression value that determines the instance number of the resource to release.

Definition at line 98 of file `Enter.h`.

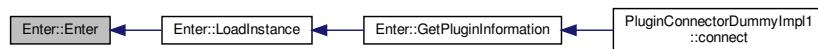
6.28.2 Constructor & Destructor Documentation

6.28.2.1 Enter()

```
Enter::Enter (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Enter.cpp](#).

Here is the caller graph for this function:



6.28.2.2 ~Enter()

```
virtual Enter::~Enter ( ) [virtual], [default]
```

6.28.3 Member Function Documentation

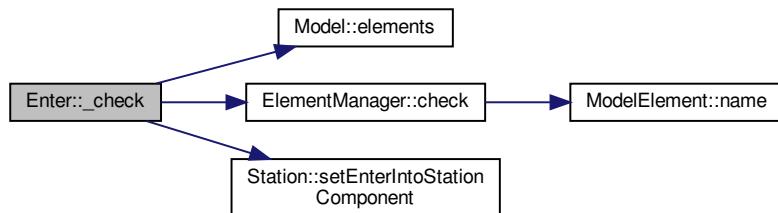
6.28.3.1 _check()

```
bool Enter::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 67 of file [Enter.cpp](#).

Here is the call graph for this function:



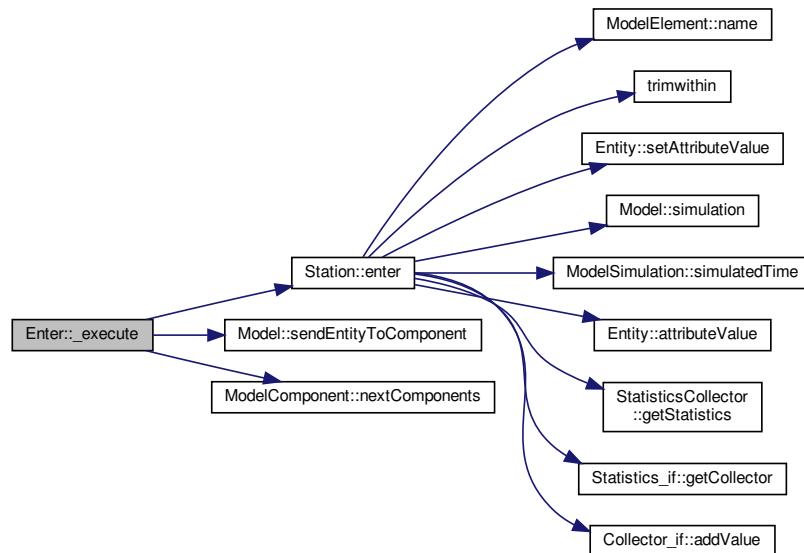
6.28.3.2 _execute()

```
void Enter::_execute (
    Entity * entity )  [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 43 of file [Enter.cpp](#).

Here is the call graph for this function:



6.28.3.3 _initBetweenReplications()

```
void Enter::_initBetweenReplications ( )  [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Enter.cpp](#).

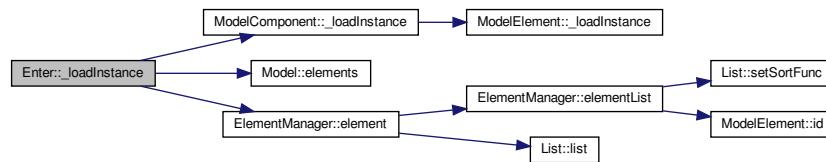
6.28.3.4 `_loadInstance()`

```
bool Enter::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 48 of file [Enter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



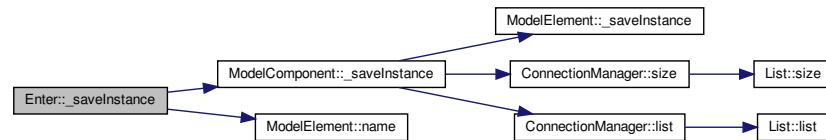
6.28.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Enter::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 61 of file [Enter.cpp](#).

Here is the call graph for this function:

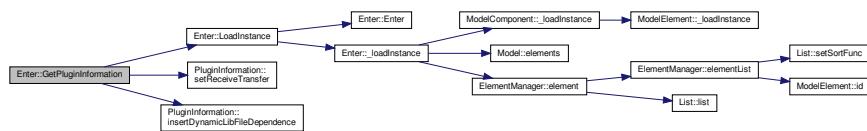


6.28.3.6 GetPluginInformation()

```
PluginInformation * Enter::GetPluginInformation ( ) [static]
```

Definition at line 76 of file [Enter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.28.3.7 getStation()

```
Station * Enter::getStation ( ) const
```

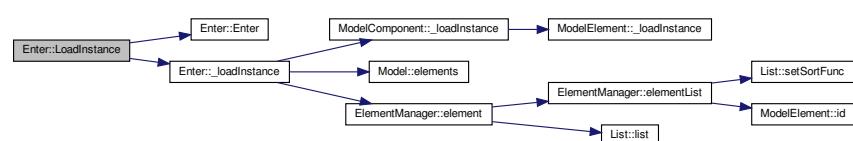
Definition at line 39 of file [Enter.cpp](#).

6.28.3.8 LoadInstance()

```
ModelComponent * Enter::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Enter.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

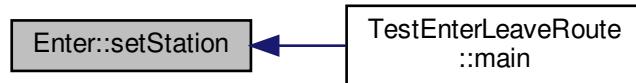


6.28.3.9 setStation()

```
void Enter::setStation (
    Station * _station )
```

Definition at line 35 of file [Enter.cpp](#).

Here is the caller graph for this function:



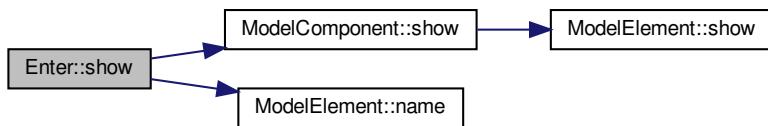
6.28.3.10 show()

```
std::string Enter::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Enter.cpp](#).

Here is the call graph for this function:



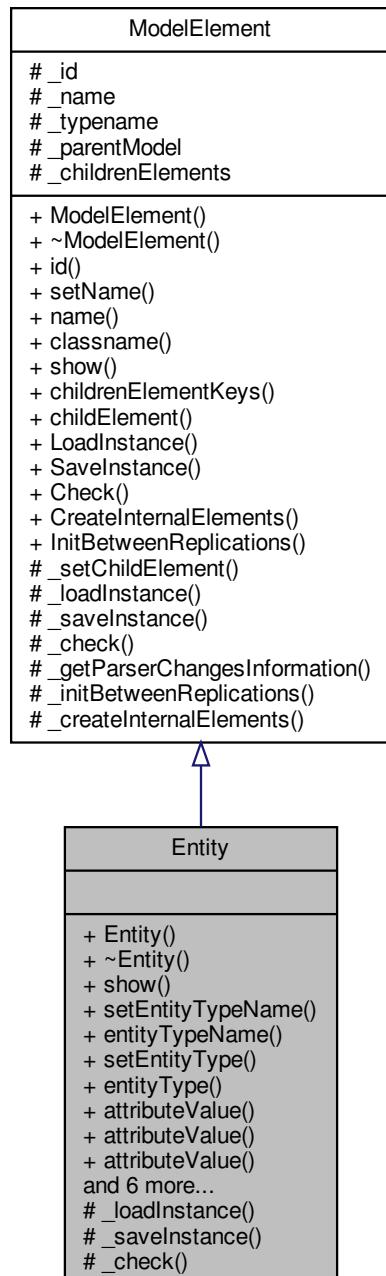
The documentation for this class was generated from the following files:

- [Enter.h](#)
- [Enter.cpp](#)

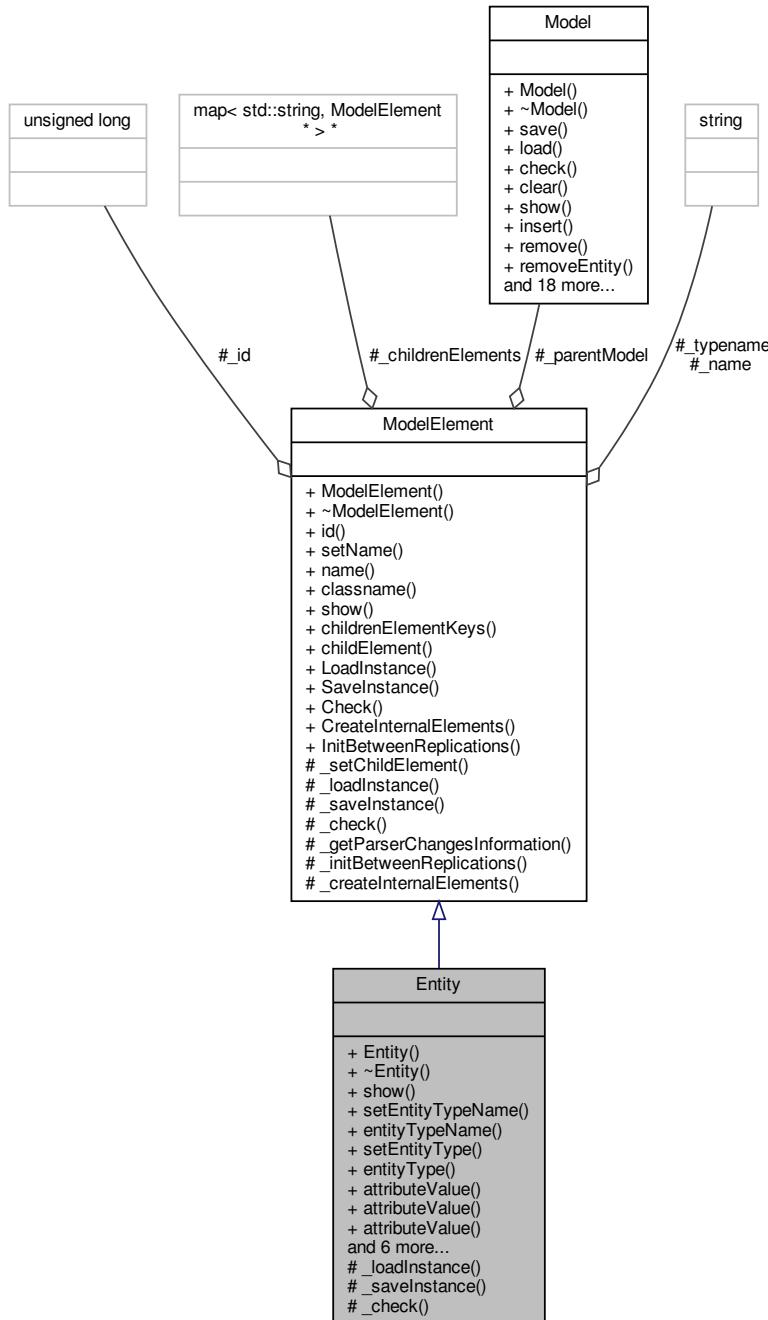
6.29 Entity Class Reference

```
#include <Entity.h>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



Public Member Functions

- `Entity (Model *model)`
- `virtual ~Entity ()=default`
- `virtual std::string show ()`
- `void setEntityTypeName (std::string entityTypeName) throw ()`
- `std::string entityTypeName () const`

- void `setEntityType (EntityType *entityType)`
- `EntityType * entityType () const`
- double `attributeValue (std::string attributeName)`
- double `attributeValue (std::string index, std::string attributeName)`
- double `attributeValue (Util::identification attributeID)`
- double `attributeValue (std::string index, Util::identification attributeID)`
- void `setAttributeValue (std::string attributeName, double value)`
- void `setAttributeValue (std::string index, std::string attributeName, double value)`
- void `setAttributeValue (Util::identification attributeID, double value)`
- void `setAttributeValue (std::string index, Util::identification attributeID, double value)`
- `Util::identification entityNumber () const`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.29.1 Detailed Description

Entity module DESCRIPTION This data module defines the various entity types and their initial picture values in a simulation. Initial costing information and holding costs are also defined for the entity. TYPICAL USES Items being produced or assembled (parts, pallets) Documents (forms, e-mails, faxes, reports) People moving through a process (customers, callers) PROMPTS Prompt Description Name The unique name of the attribute being defined. Rows Number of rows in a one- or two-dimensional attribute. Columns Number of columns in a two-dimensional attribute. Data Type The data type of the values stored in the attribute. Valid types are Real and String. The default type is Real. Initial Values Lists the initial value or values of the attribute. You can assign new values to the attribute by using the [Assign](#) module. Initial Value Entity attribute value when entity is created and enters the system. Prompt Description Entity Type The name of the entity type being defined. This name must be unique. Initial Picture Graphical representation of the entity at the start of the simulation. This value can be changed during the simulation using the [Assign](#) module. Holding Cost/Hour Hourly cost of processing the entity through the system. This cost is incurred when the entity is anywhere in the system. Initial VA Cost Initial cost value that will be assigned to the value-added cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a value-added activity. Initial NVA Cost Initial cost value that will be assigned to the non-value-added cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a non-value-added activity. Initial Waiting Cost Initial cost value that will be assigned to the waiting-cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a wait activity; for example, waiting to be batched or waiting for resource(s) at a Process module. Initial Transfer Cost Initial cost value that will be assigned to the transfer cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in a transfer activity. Initial Other Cost Initial cost value that will be assigned to the other cost attribute of the entity. This attribute accrues the costs incurred when an entity is spending time in another activity. Report Statistics Specifies whether or not statistics will be collected automatically and stored in the report database for this entity type.

Definition at line 75 of file [Entity.h](#).

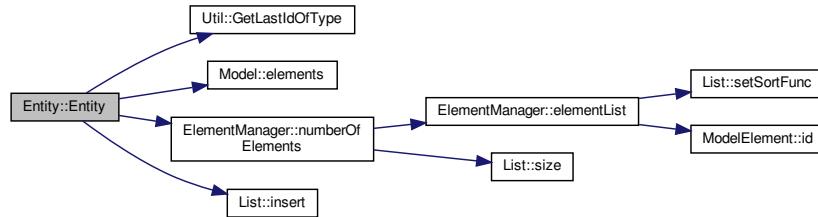
6.29.2 Constructor & Destructor Documentation

6.29.2.1 Entity()

```
Entity::Entity (
    Model * model )
```

Definition at line 19 of file [Entity.cpp](#).

Here is the call graph for this function:



6.29.2.2 ~Entity()

```
virtual Entity::~Entity () [virtual], [default]
```

6.29.3 Member Function Documentation

6.29.3.1 _check()

```
bool Entity::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 157 of file [Entity.cpp](#).

6.29.3.2 _loadInstance()

```
bool Entity::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 148 of file [Entity.cpp](#).

6.29.3.3 `_saveInstance()`

```
std::map< std::string, std::string > * Entity::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

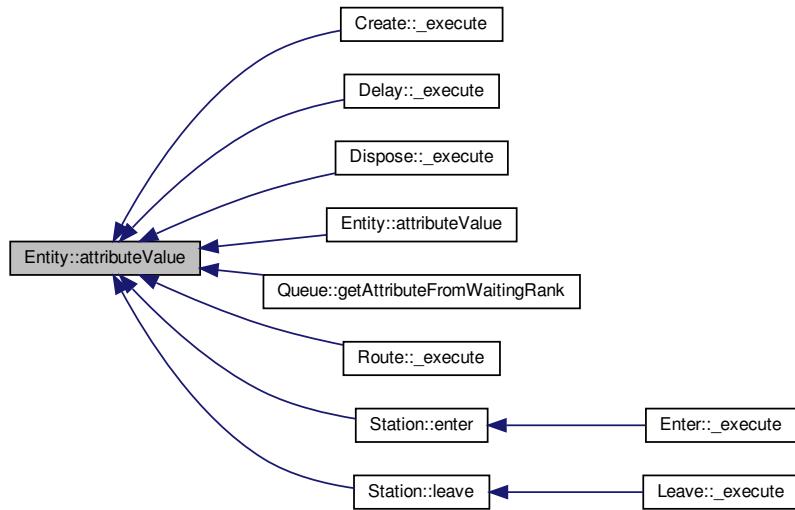
Definition at line [153](#) of file [Entity.cpp](#).

6.29.3.4 `attributeValue()` [1/4]

```
double Entity::attributeValue (
    std::string attributeName )
```

Definition at line [85](#) of file [Entity.cpp](#).

Here is the caller graph for this function:

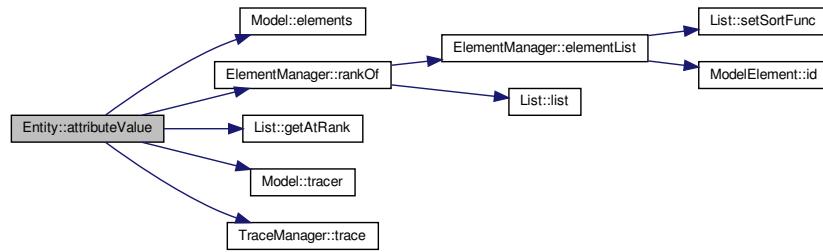


6.29.3.5 `attributeValue()` [2/4]

```
double Entity::attributeValue (
    std::string index,
    std::string attributeName )
```

Definition at line [89](#) of file [Entity.cpp](#).

Here is the call graph for this function:



6.29.3.6 `attributeValue()` [3/4]

```
double Entity::attributeValue (
    Util::identification attributeID )
```

Definition at line 104 of file [Entity.cpp](#).

Here is the call graph for this function:

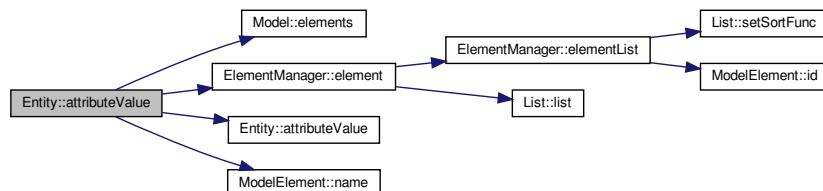


6.29.3.7 `attributeValue()` [4/4]

```
double Entity::attributeValue (
    std::string index,
    Util::identification attributeID )
```

Definition at line 108 of file [Entity.cpp](#).

Here is the call graph for this function:

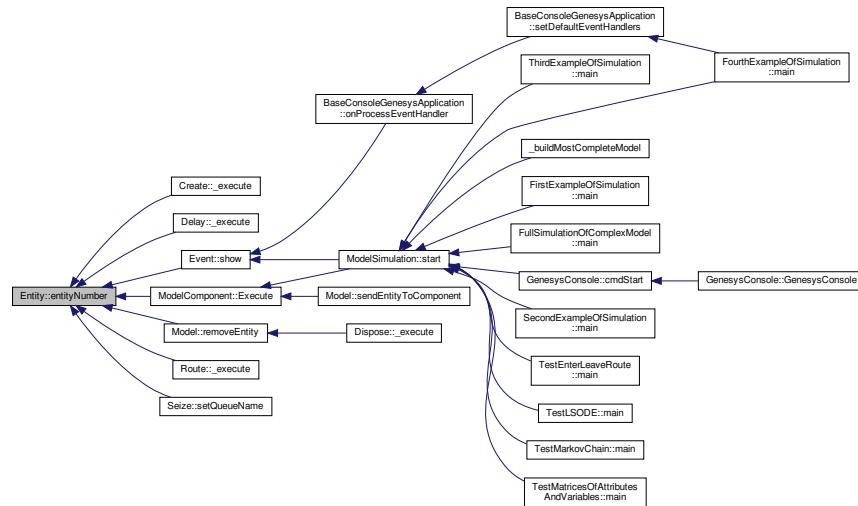


6.29.3.8 entityNumber()

```
Util::identification Entity::entityNumber() const
```

Definition at line 144 of file [Entity.cpp](#).

Here is the caller graph for this function:

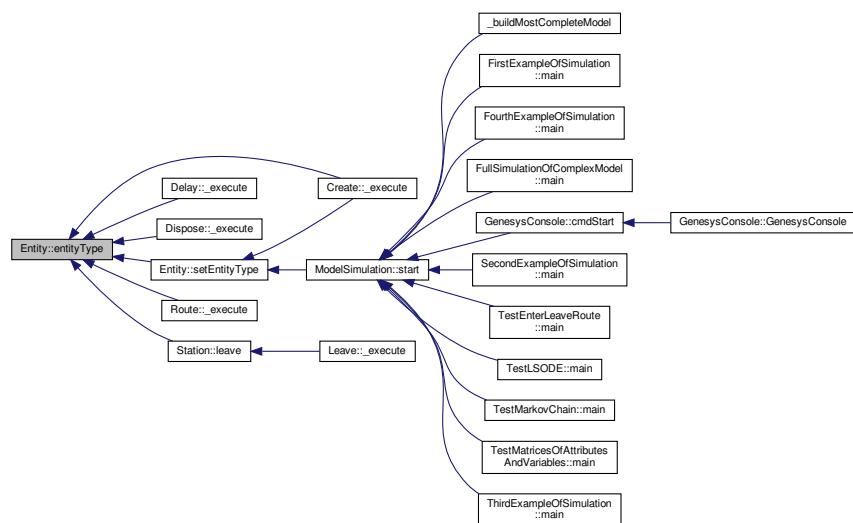


6.29.3.9 entityType()

```
EntityType * Entity::entityType() const
```

Definition at line 51 of file [Entity.cpp](#).

Here is the caller graph for this function:



6.29.3.10 entityTypeName()

```
std::string Entity::entityTypeName ( ) const
```

Definition at line 43 of file [Entity.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

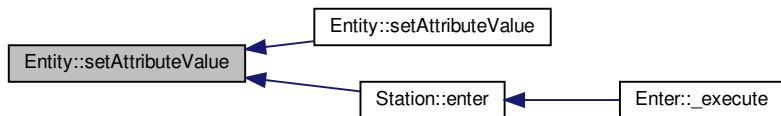


6.29.3.11 setAttributeValue() [1/4]

```
void Entity::setAttributeValue (
    std::string attributeName,
    double value )
```

Definition at line 116 of file [Entity.cpp](#).

Here is the caller graph for this function:

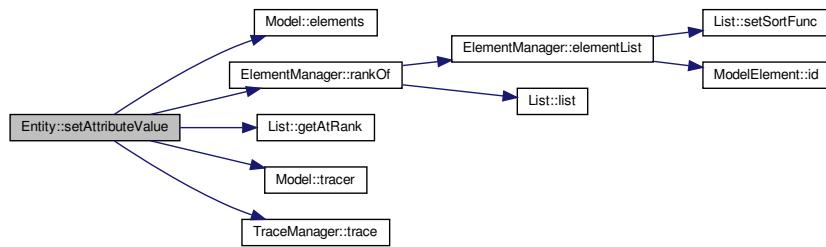


6.29.3.12 setAttributeValue() [2/4]

```
void Entity::setAttributeValue (
    std::string index,
    std::string attributeName,
    double value )
```

Definition at line 120 of file [Entity.cpp](#).

Here is the call graph for this function:



6.29.3.13 setAttributeValue() [3/4]

```
void Entity::setAttributeValue (
    Util::identification attributeID,
    double value )
```

Definition at line 135 of file [Entity.cpp](#).

Here is the call graph for this function:

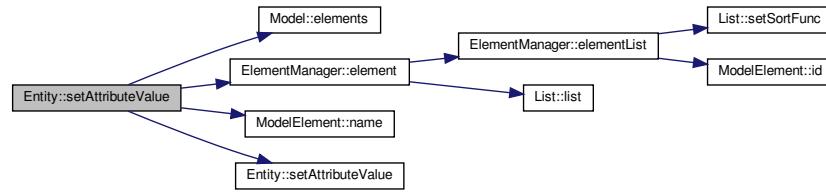


6.29.3.14 setAttributeValue() [4/4]

```
void Entity::setAttributeValue (
    std::string index,
    Util::identification attributeID,
    double value )
```

Definition at line 139 of file [Entity.cpp](#).

Here is the call graph for this function:



6.29.3.15 setEntityType()

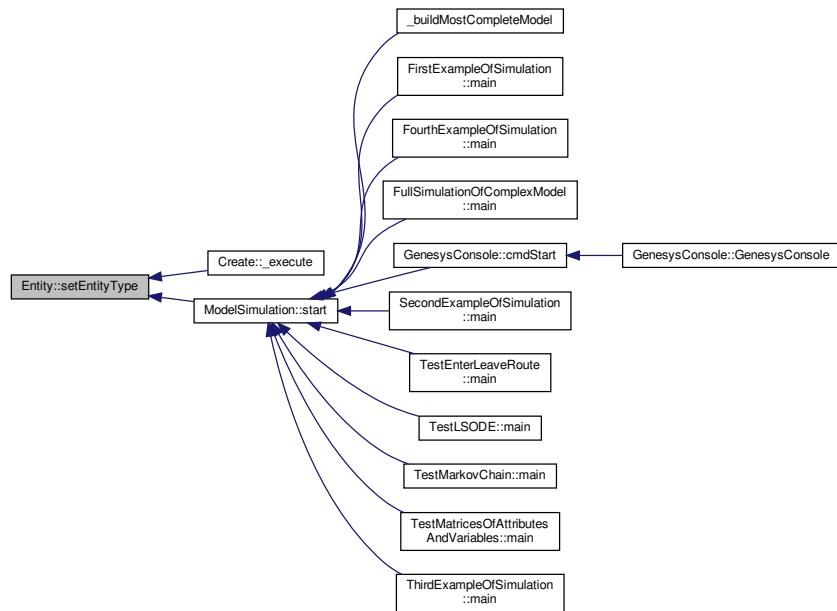
```
void Entity::setEntityType (
    EntityType * entityType )
```

Definition at line 47 of file [Entity.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

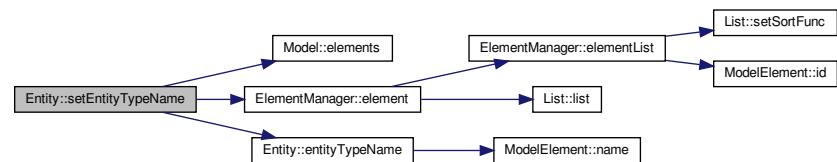


6.29.3.16 setEntityType()

```
void Entity::setEntityType( std::string entityTypeName ) throw()
```

Definition at line 34 of file [Entity.cpp](#).

Here is the call graph for this function:



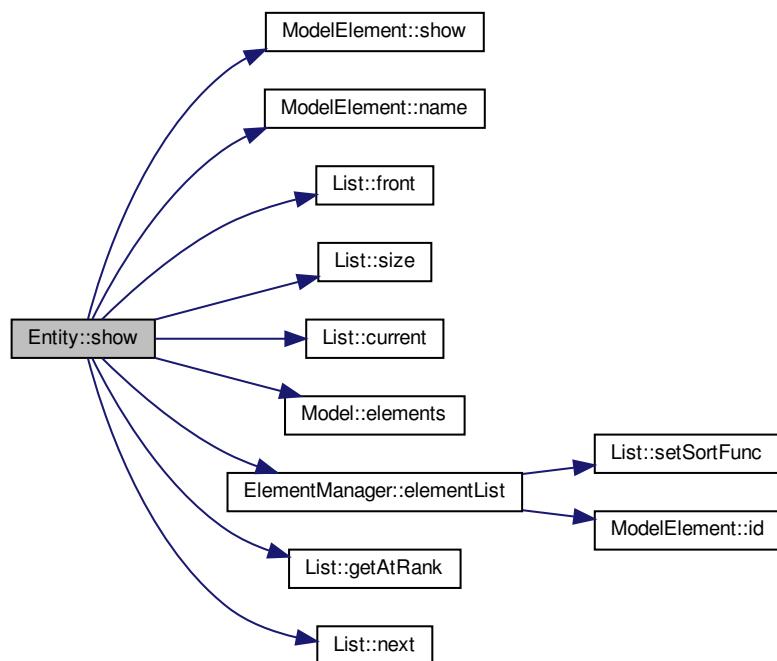
6.29.3.17 show()

```
std::string Entity::show ( ) [virtual]
```

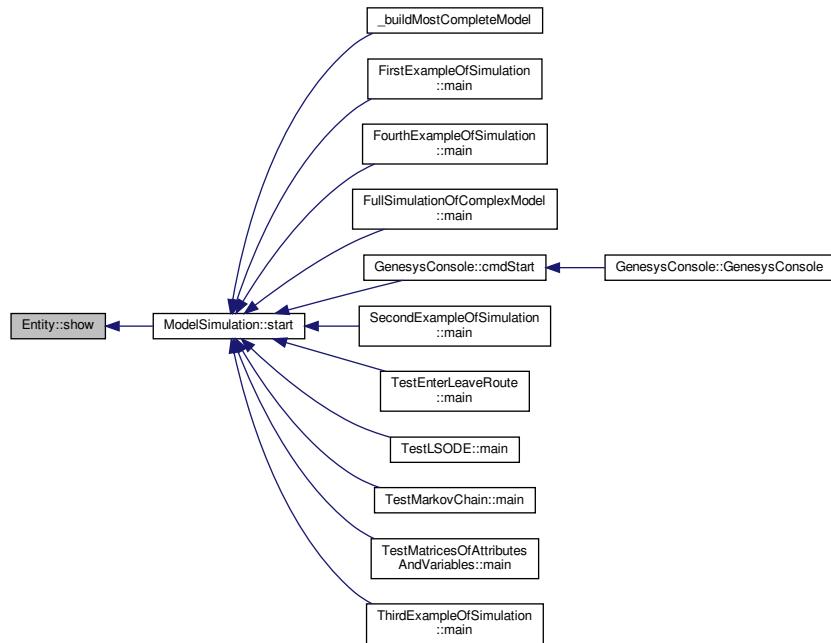
Reimplemented from [ModelElement](#).

Definition at line 55 of file [Entity.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



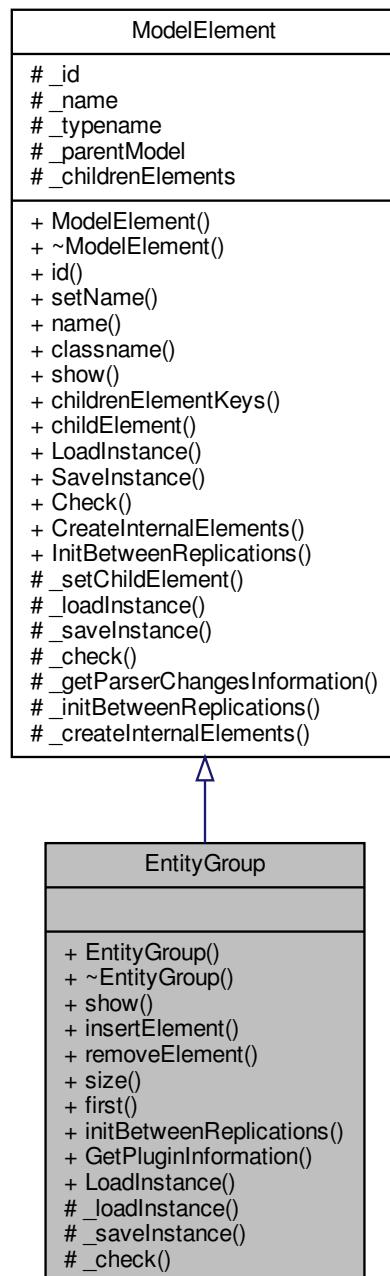
The documentation for this class was generated from the following files:

- [Entity.h](#)
- [Entity.cpp](#)

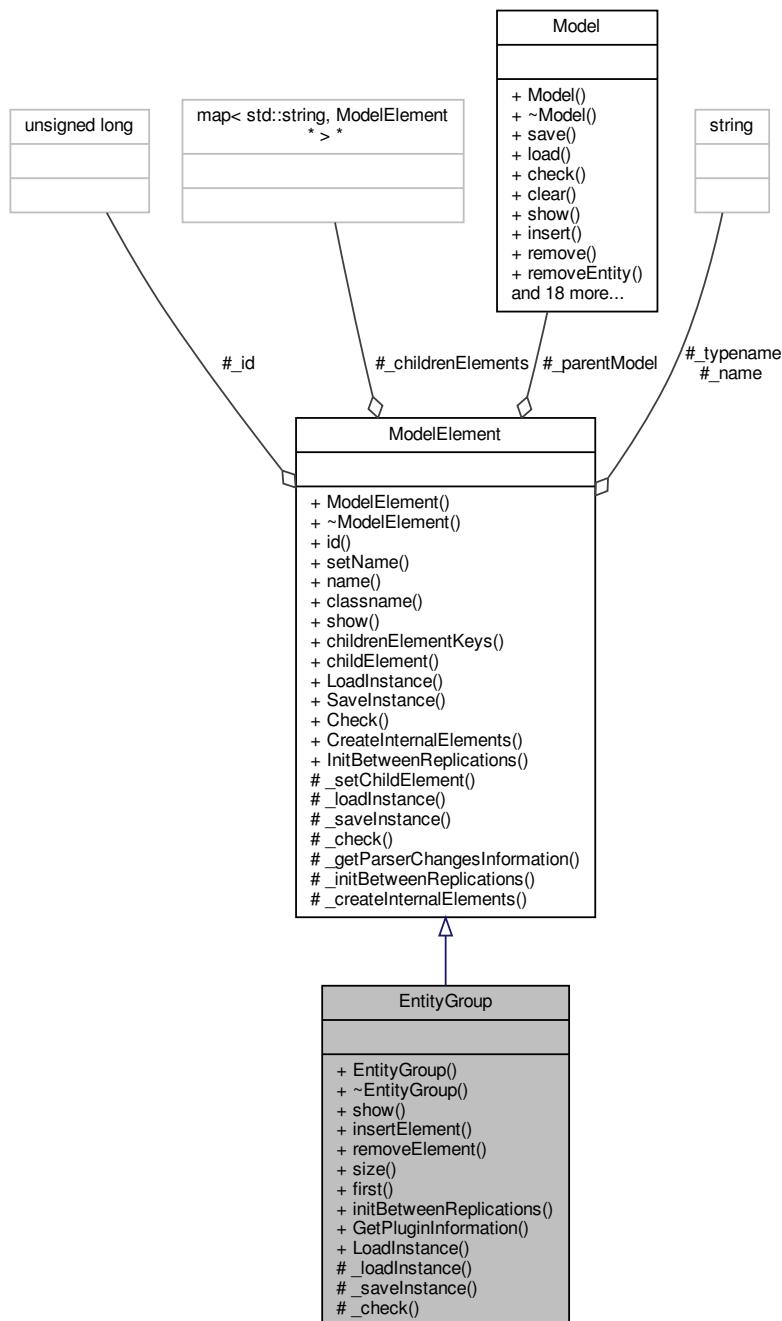
6.30 EntityGroup Class Reference

```
#include <EntityGroup.h>
```

Inheritance diagram for EntityGroup:



Collaboration diagram for EntityGroup:



Public Member Functions

- `EntityGroup (Model *model, std::string name="")`
- `virtual ~EntityGroup ()`
- `virtual std::string show ()`
- `void insertElement (Entity *element)`
- `void removeElement (Entity *element)`

- `unsigned int size ()`
- `Entity * first ()`
- `void initBetweenReplications ()`

Static Public Member Functions

- `static PluginInformation * GetPluginInformation ()`
- `static ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- `virtual bool _loadInstance (std::map< std::string, std::string > *fields)`
- `virtual std::map< std::string, std::string > * _saveInstance ()`
- `virtual bool _check (std::string *errorMessage)`

Additional Inherited Members

6.30.1 Detailed Description

Definition at line 25 of file [EntityGroup.h](#).

6.30.2 Constructor & Destructor Documentation

6.30.2.1 EntityGroup()

```
EntityGroup::EntityGroup (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [EntityGroup.cpp](#).

Here is the caller graph for this function:



6.30.2.2 ~EntityGroup()

```
EntityGroup::~EntityGroup ( ) [virtual]
```

Definition at line 27 of file [EntityGroup.cpp](#).

6.30.3 Member Function Documentation

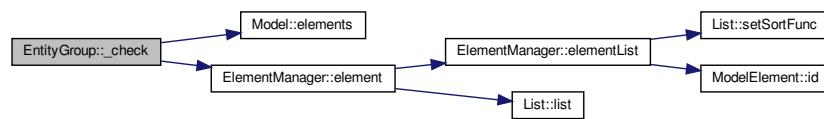
6.30.3.1 `_check()`

```
bool EntityGroup::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 93 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



6.30.3.2 `_loadInstance()`

```
bool EntityGroup::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 78 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



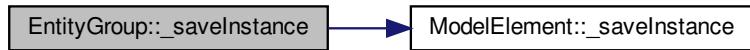
6.30.3.3 `_saveInstance()`

```
std::map< std::string, std::string > * EntityGroup::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 88 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



6.30.3.4 `first()`

```
Entity * EntityGroup::first ( )
```

Definition at line 55 of file [EntityGroup.cpp](#).

Here is the call graph for this function:

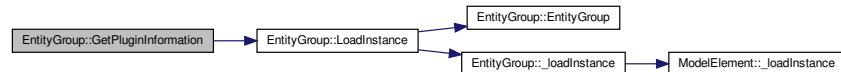


6.30.3.5 `GetPluginInformation()`

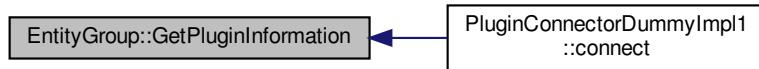
```
PluginInformation * EntityGroup::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

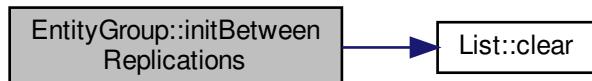


6.30.3.6 initBetweenReplications()

```
void EntityGroup::initBetweenReplications ( )
```

Definition at line 47 of file [EntityGroup.cpp](#).

Here is the call graph for this function:

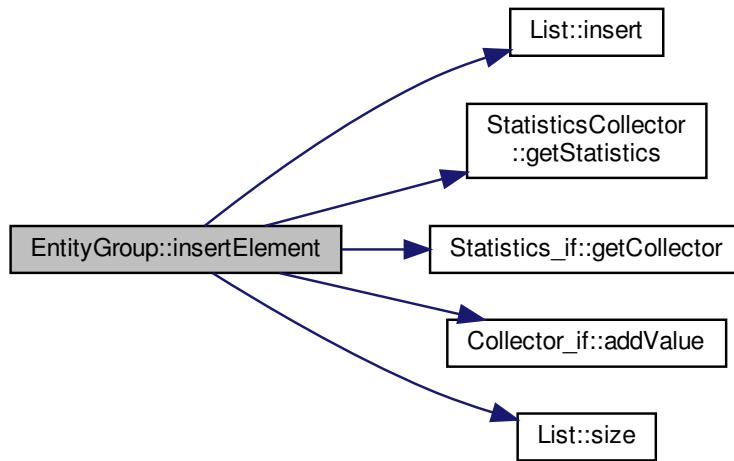


6.30.3.7 insertElement()

```
void EntityGroup::insertElement ( Entity * element )
```

Definition at line 36 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



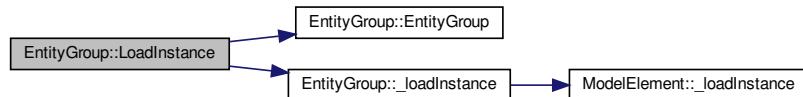
6.30.3.8 LoadInstance()

```

ModelElement * EntityGroup::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
  
```

Definition at line 68 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

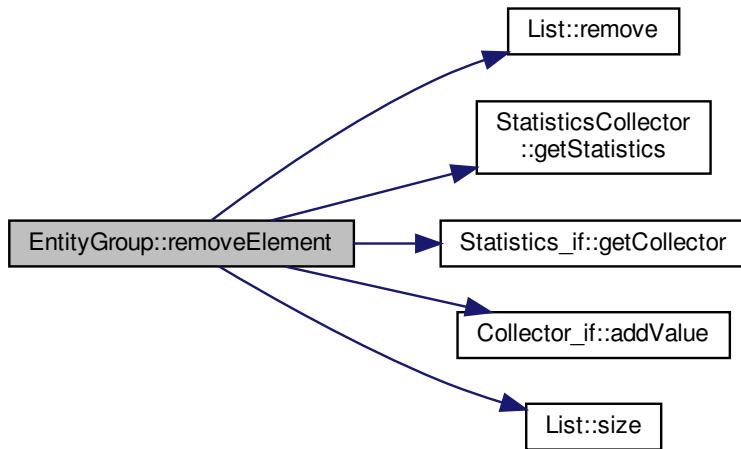


6.30.3.9 removeElement()

```
void EntityGroup::removeElement ( Entity * element )
```

Definition at line 41 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



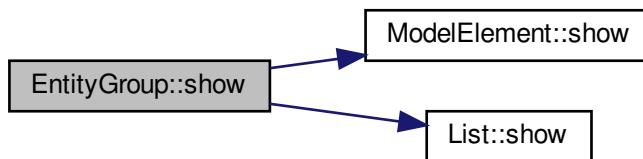
6.30.3.10 show()

```
std::string EntityGroup::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 31 of file [EntityGroup.cpp](#).

Here is the call graph for this function:

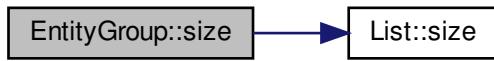


6.30.3.11 size()

```
unsigned int EntityGroup::size ( )
```

Definition at line 51 of file [EntityGroup.cpp](#).

Here is the call graph for this function:



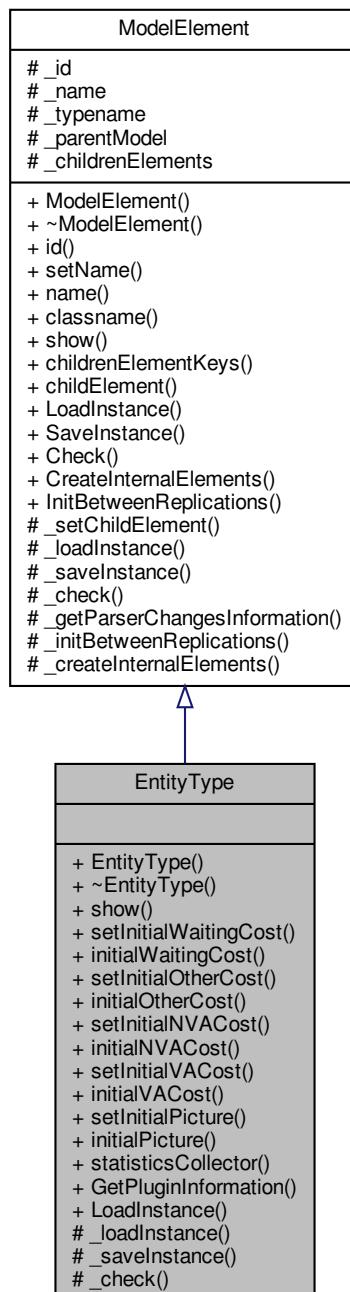
The documentation for this class was generated from the following files:

- [EntityGroup.h](#)
- [EntityGroup.cpp](#)

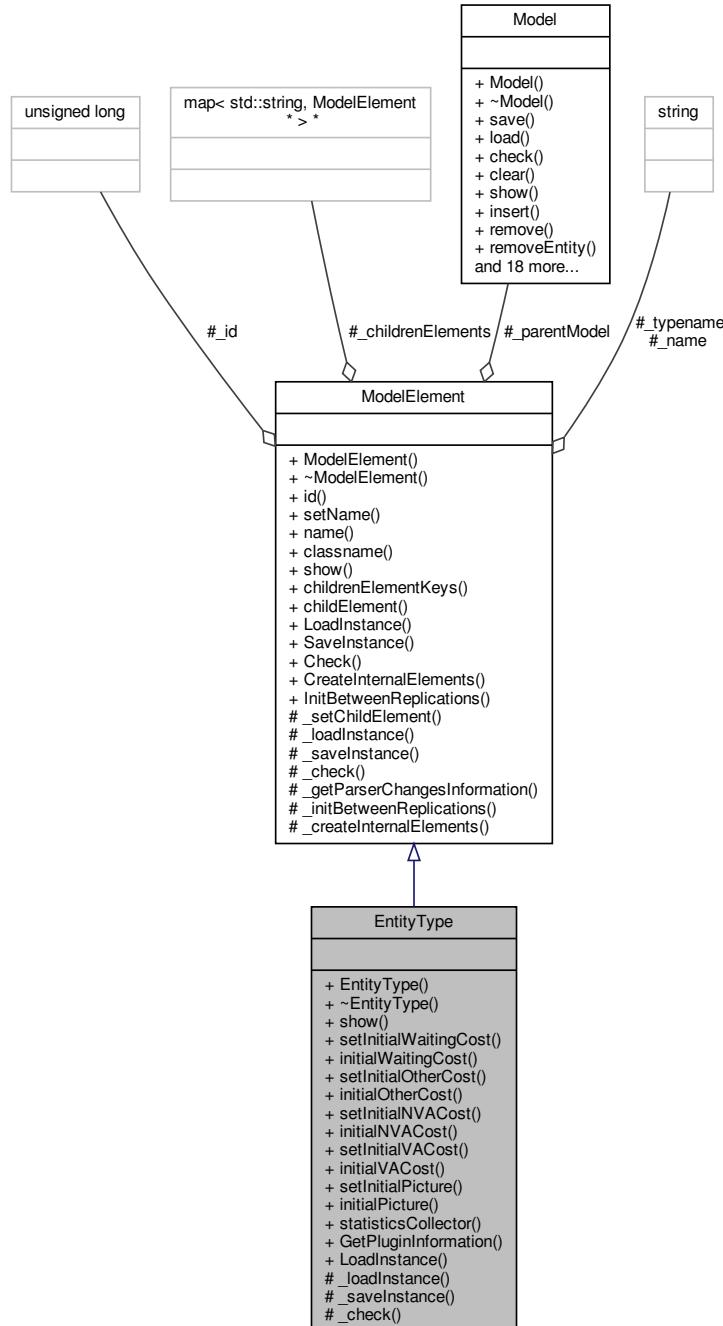
6.31 EntityType Class Reference

```
#include <EntityType.h>
```

Inheritance diagram for EntityType:



Collaboration diagram for EntityType:



Public Member Functions

- `EntityType (Model *model, std::string name="")`
- `virtual ~EntityType ()`
- `virtual std::string show ()`
- `void setInitialWaitingCost (double _initialWaitingCost)`
- `double initialWaitingCost () const`

- void [setInitialOtherCost](#) (double _initialOtherCost)
- double [initialOtherCost](#) () const
- void [setInitialNVACost](#) (double _initialNVACost)
- double [initialNVACost](#) () const
- void [setInitialVACost](#) (double _initialVACost)
- double [initialVACost](#) () const
- void [setInitialPicture](#) (std::string _initialPicture)
- std::string [initialPicture](#) () const
- [StatisticsCollector * statisticsCollector](#) (std::string name)

Static Public Member Functions

- static [PluginInformation * GetPluginInformation](#) ()
- static [ModelElement * LoadInstance](#) ([Model](#) *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)

Additional Inherited Members

6.31.1 Detailed Description

Definition at line 25 of file [EntityType.h](#).

6.31.2 Constructor & Destructor Documentation

6.31.2.1 EntityType()

```
EntityType::EntityType (
    Model * model,
    std::string name = "" )
```

Definition at line 23 of file [EntityType.cpp](#).

Here is the caller graph for this function:

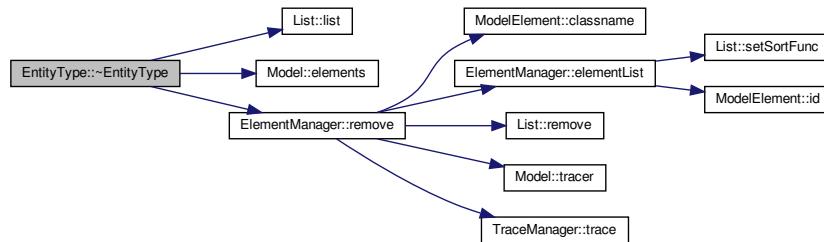


6.31.2.2 ~EntityType()

```
EntityType::~EntityType ( ) [virtual]
```

Definition at line 51 of file [EntityType.cpp](#).

Here is the call graph for this function:



6.31.3 Member Function Documentation

6.31.3.1 _check()

```
bool EntityType::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 184 of file [EntityType.cpp](#).

6.31.3.2 _loadInstance()

```
bool EntityType::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

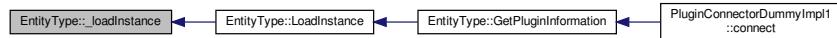
Reimplemented from [ModelElement](#).

Definition at line 162 of file [EntityType.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.3.3 _saveInstance()

`std::map< std::string, std::string > * EntityType::_saveInstance () [protected], [virtual]`

Reimplemented from [ModelElement](#).

Definition at line 174 of file [EntityType.cpp](#).

Here is the call graph for this function:

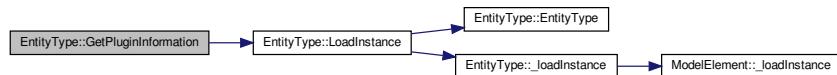


6.31.3.4 GetPluginInformation()

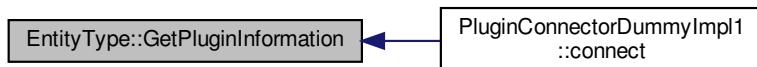
`PluginInformation * EntityType::GetPluginInformation () [static]`

Definition at line 148 of file [EntityType.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.3.5 initialNVACost()

```
double EntityType::initialNVACost () const
```

Definition at line 89 of file [EntityType.cpp](#).

6.31.3.6 initialOtherCost()

```
double EntityType::initialOtherCost () const
```

Definition at line 81 of file [EntityType.cpp](#).

6.31.3.7 initialPicture()

```
std::string EntityType::initialPicture () const
```

Definition at line 105 of file [EntityType.cpp](#).

6.31.3.8 initialVACost()

```
double EntityType::initialVACost () const
```

Definition at line 97 of file [EntityType.cpp](#).

6.31.3.9 initialWaitingCost()

```
double EntityType::initialWaitingCost () const
```

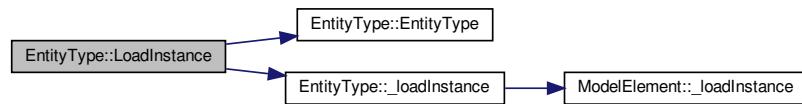
Definition at line 73 of file [EntityType.cpp](#).

6.31.3.10 LoadInstance()

```
ModelElement * EntityType::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 152 of file [EntityType.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.3.11 setInitialNVACost()

```
void EntityType::setInitialNVACost (
    double _initialNVACost )
```

Definition at line 85 of file [EntityType.cpp](#).

6.31.3.12 setInitialOtherCost()

```
void EntityType::setInitialOtherCost (
    double _initialOtherCost )
```

Definition at line 77 of file [EntityType.cpp](#).

6.31.3.13 setInitialPicture()

```
void EntityType::setInitialPicture (
    std::string _initialPicture )
```

Definition at line 101 of file [EntityType.cpp](#).

6.31.3.14 setInitialVACost()

```
void EntityType::setInitialVACost (  
    double _initialVACost )
```

Definition at line 93 of file [EntityType.cpp](#).

6.31.3.15 setInitialWaitingCost()

```
void EntityType::setInitialWaitingCost (  
    double _initialWaitingCost )
```

Definition at line 69 of file [EntityType.cpp](#).

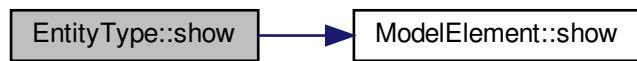
6.31.3.16 show()

```
std::string EntityType::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 64 of file [EntityType.cpp](#).

Here is the call graph for this function:

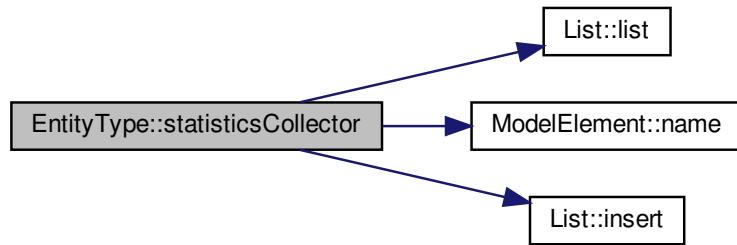


6.31.3.17 statisticsCollector()

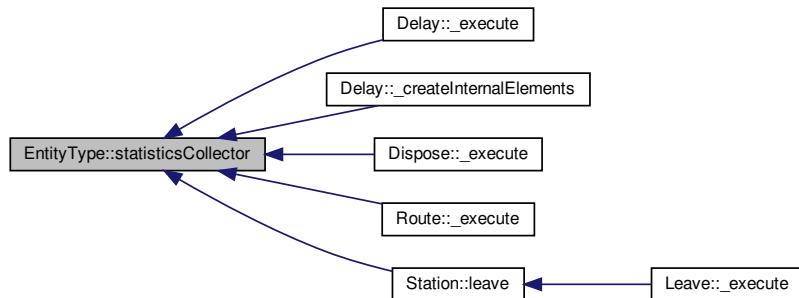
```
StatisticsCollector * EntityType::statisticsCollector (  
    std::string name )
```

Definition at line 133 of file [EntityType.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



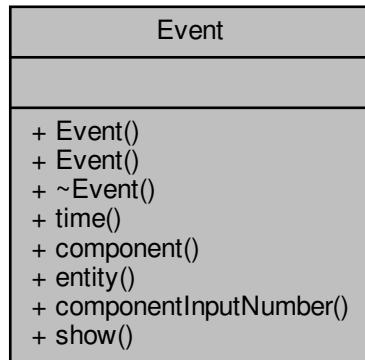
The documentation for this class was generated from the following files:

- [EntityType.h](#)
- [EntityType.cpp](#)

6.32 Event Class Reference

```
#include <Event.h>
```

Collaboration diagram for Event:



Public Member Functions

- `Event (double time, Entity *entity, ModelComponent *component, unsigned int componentInputNumber=0)`
- `Event (double time, Entity *entity, Connection *connection)`
- `virtual ~Event ()=default`
- `double time () const`
- `ModelComponent * component () const`
- `Entity * entity () const`
- `unsigned int componentInputNumber () const`
- `std::string show ()`

6.32.1 Detailed Description

An an instantaneous event, triggered at a certain moment by an entity upon reaching a component. The simulated time advances in discrete points in time and that are the instants that an event is triggered.

Definition at line 26 of file [Event.h](#).

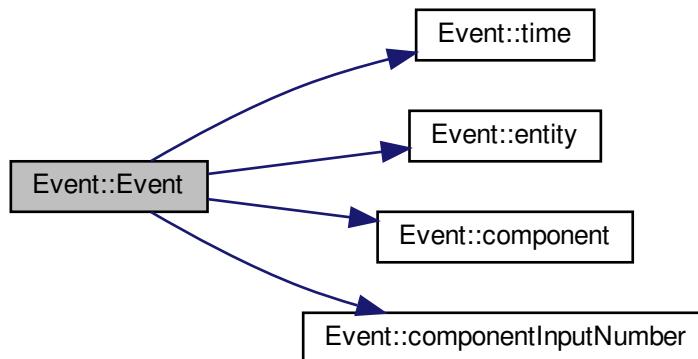
6.32.2 Constructor & Destructor Documentation

6.32.2.1 Event() [1/2]

```
Event::Event (
    double time,
    Entity * entity,
    ModelComponent * component,
    unsigned int componentInputNumber = 0 )
```

Definition at line 16 of file [Event.cpp](#).

Here is the call graph for this function:

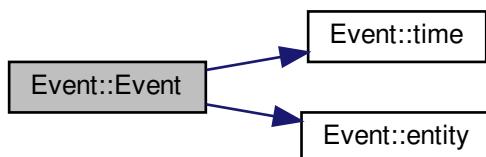


6.32.2.2 Event() [2/2]

```
Event::Event (
    double time,
    Entity * entity,
    Connection * connection )
```

Definition at line 23 of file [Event.cpp](#).

Here is the call graph for this function:



6.32.2.3 ~Event()

```
virtual Event::~Event ( ) [virtual], [default]
```

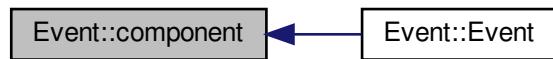
6.32.3 Member Function Documentation

6.32.3.1 component()

```
ModelComponent * Event::component ( ) const
```

Definition at line 45 of file [Event.cpp](#).

Here is the caller graph for this function:

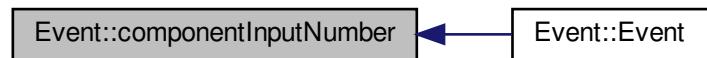


6.32.3.2 componentInputNumber()

```
unsigned int Event::componentInputNumber ( ) const
```

Definition at line 37 of file [Event.cpp](#).

Here is the caller graph for this function:

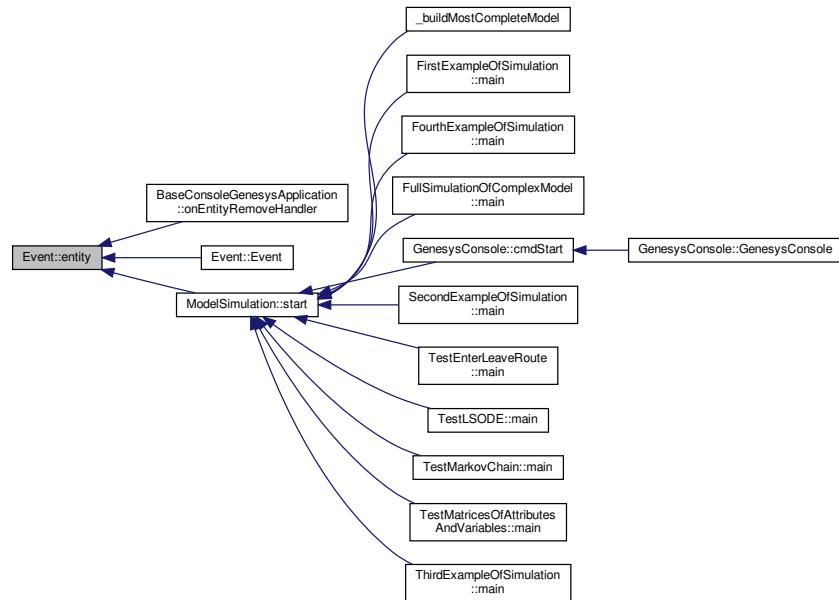


6.32.3.3 entity()

```
Entity * Event::entity ( ) const
```

Definition at line 49 of file [Event.cpp](#).

Here is the caller graph for this function:

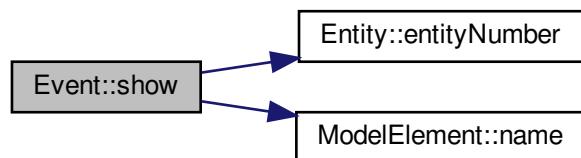


6.32.3.4 show()

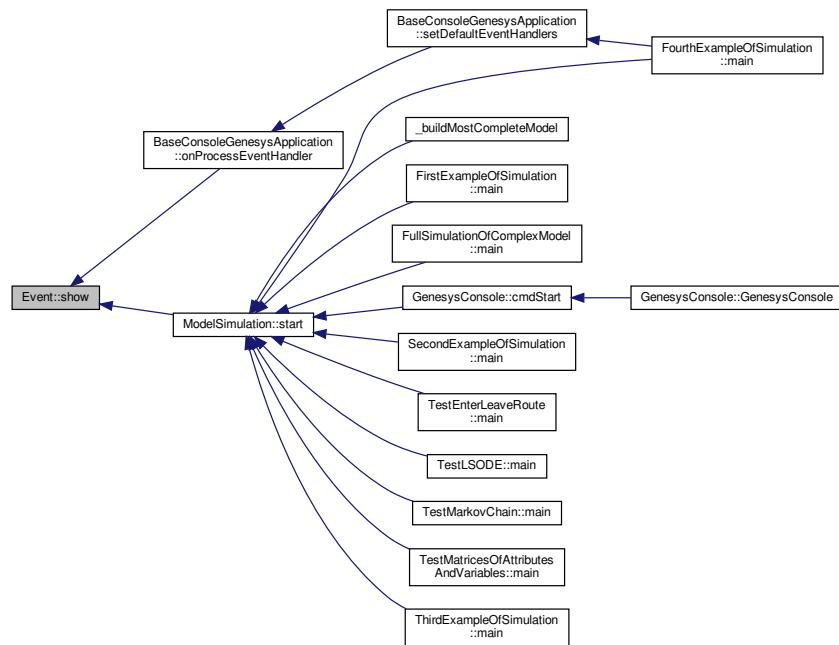
```
std::string Event::show ( )
```

Definition at line 31 of file [Event.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

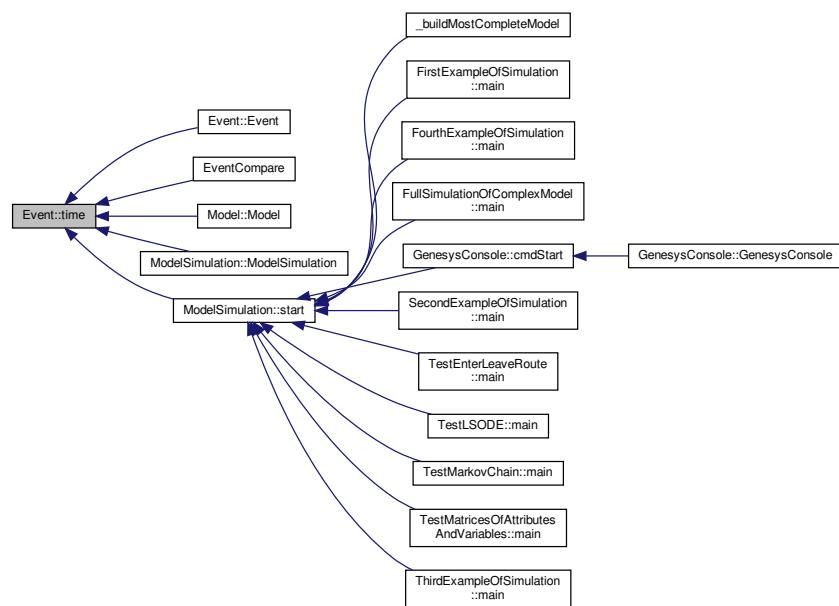


6.32.3.5 time()

```
double Event::time ( ) const
```

Definition at line 41 of file [Event.cpp](#).

Here is the caller graph for this function:



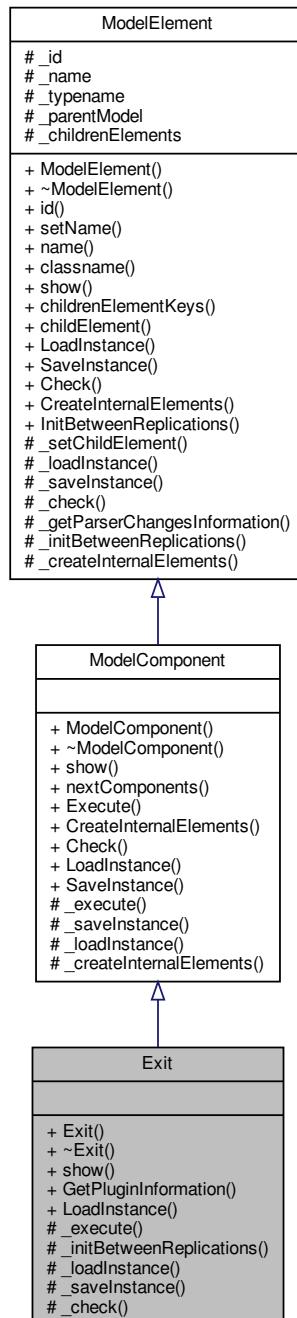
The documentation for this class was generated from the following files:

- [Event.h](#)
- [Event.cpp](#)

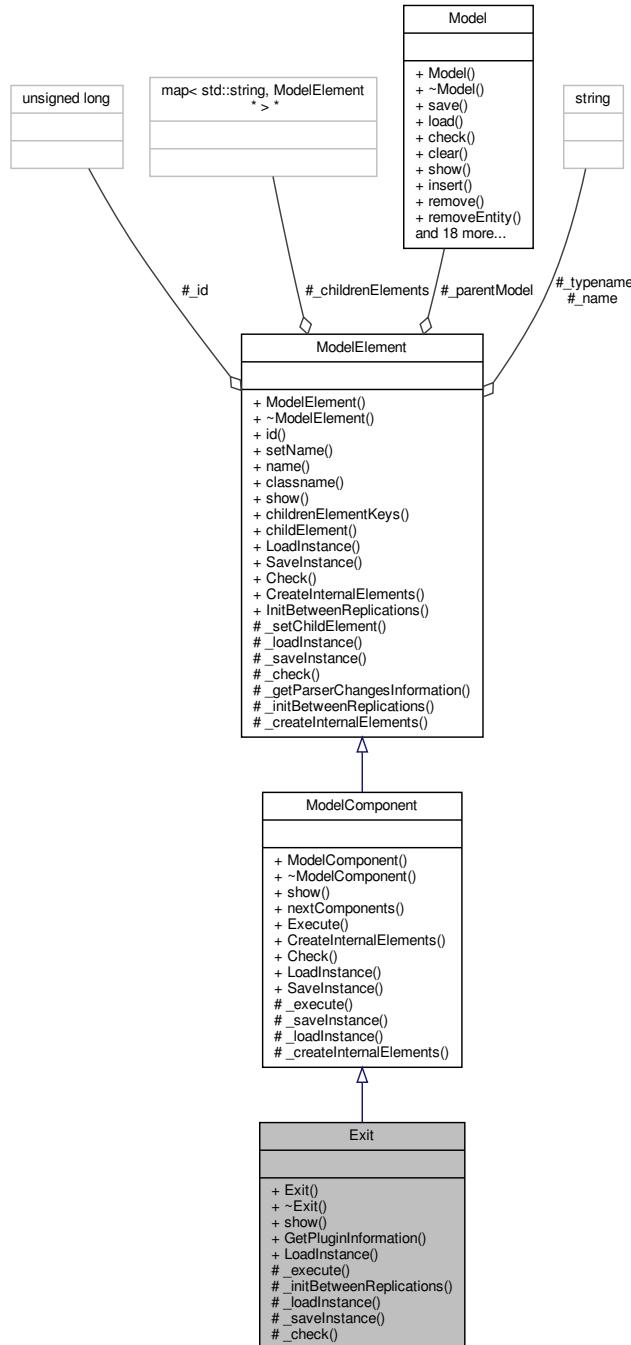
6.33 Exit Class Reference

```
#include <Exit.h>
```

Inheritance diagram for Exit:



Collaboration diagram for Exit:



Public Member Functions

- `Exit (Model *model, std::string name="")`
- virtual `~Exit ()=default`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.33.1 Detailed Description

Exit module DESCRIPTION The `Exit` module releases the entity's cells on the specified conveyor. If another entity is waiting in queue for the conveyor at the same station when the cells are released, it will then access the conveyor.
TYPICAL USES Cases exit a conveyor for packing Bad parts are removed from the conveyor and disposed Passengers remove luggage from the baggage claim conveyor
PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor on which the entity will exit. If left blank, the previously accessed conveyor is assumed.

of Cells Number of contiguous conveyor cells the entity will relinquish.

Definition at line 37 of file `Exit.h`.

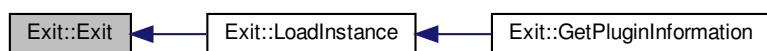
6.33.2 Constructor & Destructor Documentation

6.33.2.1 Exit()

```
Exit::Exit (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file `Exit.cpp`.

Here is the caller graph for this function:



6.33.2.2 ~Exit()

```
virtual Exit::~Exit ( ) [virtual], [default]
```

6.33.3 Member Function Documentation

6.33.3.1 _check()

```
bool Exit::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Exit.cpp](#).

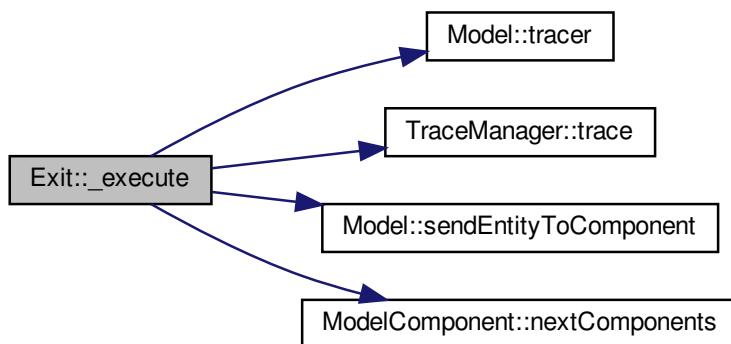
6.33.3.2 _execute()

```
void Exit::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Exit.cpp](#).

Here is the call graph for this function:



6.33.3.3 `_initBetweenReplications()`

```
void Exit::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Exit.cpp](#).

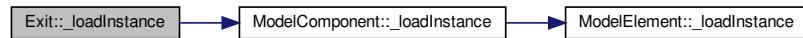
6.33.3.4 `_loadInstance()`

```
bool Exit::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Exit.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



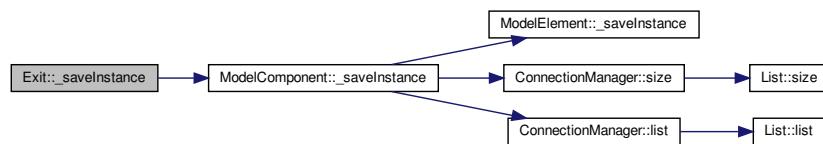
6.33.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Exit::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Exit.cpp](#).

Here is the call graph for this function:



6.33.3.6 GetPluginInformation()

```
PluginInformation * Exit::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Exit.cpp](#).

Here is the call graph for this function:

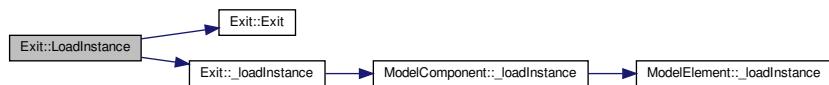


6.33.3.7 LoadInstance()

```
ModelComponent * Exit::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Exit.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



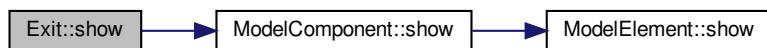
6.33.3.8 show()

```
std::string Exit::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Exit.cpp](#).

Here is the call graph for this function:



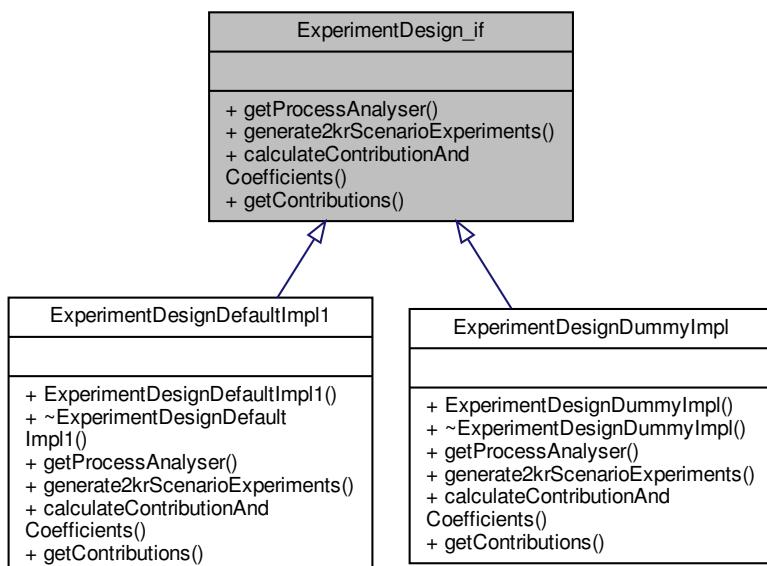
The documentation for this class was generated from the following files:

- [Exit.h](#)
- [Exit.cpp](#)

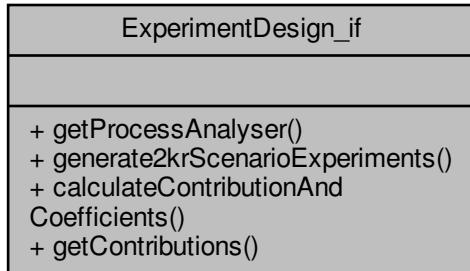
6.34 ExperimentDesign_if Class Reference

```
#include <ExperimentDesign_if.h>
```

Inheritance diagram for ExperimentDesign_if:



Collaboration diagram for ExperimentDesign_if:



Public Member Functions

- virtual `ProcessAnalyser_if * getProcessAnalyser () const =0`
- virtual `bool generate2krScenarioExperiments ()=0`
- virtual `bool calculateContributionAndCoefficients ()=0`
- virtual `std::list< FactorOrInteractionContribution * > * getContributions () const =0`

6.34.1 Detailed Description

It designs a set of experiments ([SimulationScenario](#)) where que level of factors ([SimulationControl](#)) are set automatically to create a $2^k.r$ experiment design, and where the contributions of the factors and their interactions (just a set of [SimulationControl](#)) can be obtained.

Definition at line 23 of file [ExperimentDesign_if.h](#).

6.34.2 Member Function Documentation

6.34.2.1 calculateContributionAndCoefficients()

```
virtual bool ExperimentDesign_if::calculateContributionAndCoefficients ( ) [pure virtual]
```

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

6.34.2.2 generate2krScenarioExperiments()

```
virtual bool ExperimentDesign_if::generate2krScenarioExperiments ( ) [pure virtual]
```

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

6.34.2.3 getContributions()

```
virtual std::list<FactorOrInteractionContribution*>* ExperimentDesign_if::getContributions ( ) const [pure virtual]
```

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

6.34.2.4 getProcessAnalyser()

```
virtual ProcessAnalyser_if* ExperimentDesign_if::getProcessAnalyser ( ) const [pure virtual]
```

Implemented in [ExperimentDesignDummyImpl](#), and [ExperimentDesignDefaultImpl1](#).

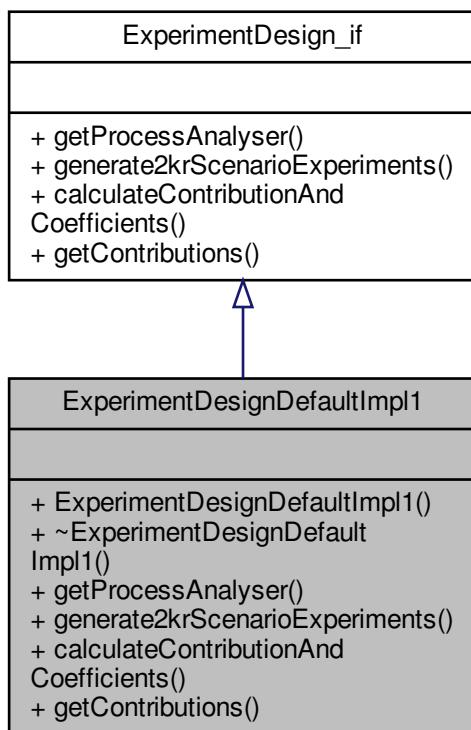
The documentation for this class was generated from the following file:

- [ExperimentDesign_if.h](#)

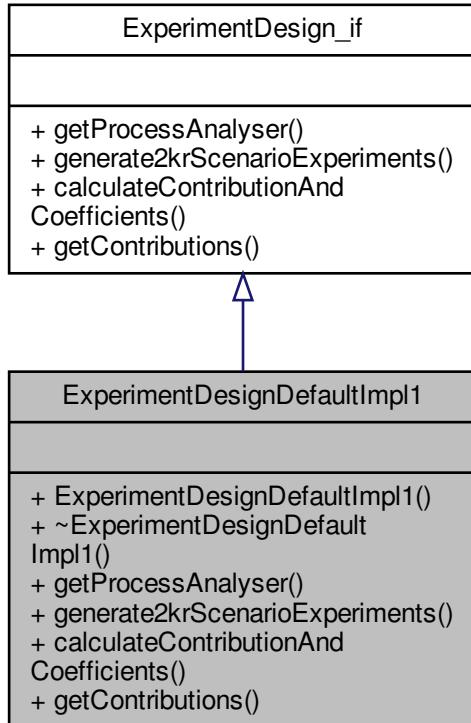
6.35 ExperimentDesignDefaultImpl1 Class Reference

```
#include <ExperimentDesignDefaultImpl1.h>
```

Inheritance diagram for ExperimentDesignDefaultImpl1:



Collaboration diagram for ExperimentDesignDefaultImpl1:



Public Member Functions

- `ExperimentDesignDefaultImpl1 ()`
- virtual `~ExperimentDesignDefaultImpl1 ()=default`
- virtual `ProcessAnalyser_if * getProcessAnalyser () const`
- virtual `bool generate2krScenarioExperiments ()`
- virtual `bool calculateContributionAndCoefficients ()`
- virtual `std::list< FactorOrInteractionContribution * > * getContributions () const`

6.35.1 Detailed Description

Definition at line 19 of file [ExperimentDesignDefaultImpl1.h](#).

6.35.2 Constructor & Destructor Documentation

6.35.2.1 ExperimentDesignDefaultImpl1()

```
ExperimentDesignDefaultImpl1::ExperimentDesignDefaultImpl1 ( )
```

Definition at line 16 of file [ExperimentDesignDefaultImpl1.cpp](#).

6.35.2.2 ~ExperimentDesignDefaultImpl1()

```
virtual ExperimentDesignDefaultImpl1::~ExperimentDesignDefaultImpl1 ( ) [virtual], [default]
```

6.35.3 Member Function Documentation

6.35.3.1 calculateContributionAndCoefficients()

```
bool ExperimentDesignDefaultImpl1::calculateContributionAndCoefficients ( ) [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 28 of file [ExperimentDesignDefaultImpl1.cpp](#).

6.35.3.2 generate2krScenarioExperiments()

```
bool ExperimentDesignDefaultImpl1::generate2krScenarioExperiments ( ) [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 24 of file [ExperimentDesignDefaultImpl1.cpp](#).

6.35.3.3 getContributions()

```
std::list< FactorOrInteractionContribution * > * ExperimentDesignDefaultImpl1::getContributions  
( ) const [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 20 of file [ExperimentDesignDefaultImpl1.cpp](#).

6.35.3.4 getProcessAnalyser()

```
ProcessAnalyser_if * ExperimentDesignDefaultImpl1::getProcessAnalyser () const [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 32 of file [ExperimentDesignDefaultImpl1.cpp](#).

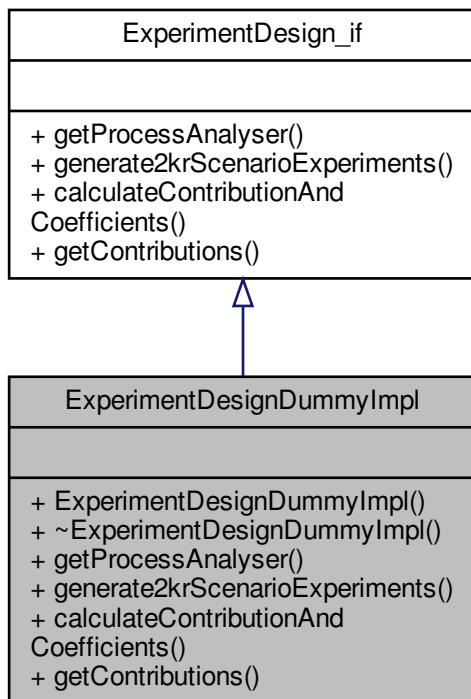
The documentation for this class was generated from the following files:

- [ExperimentDesignDefaultImpl1.h](#)
- [ExperimentDesignDefaultImpl1.cpp](#)

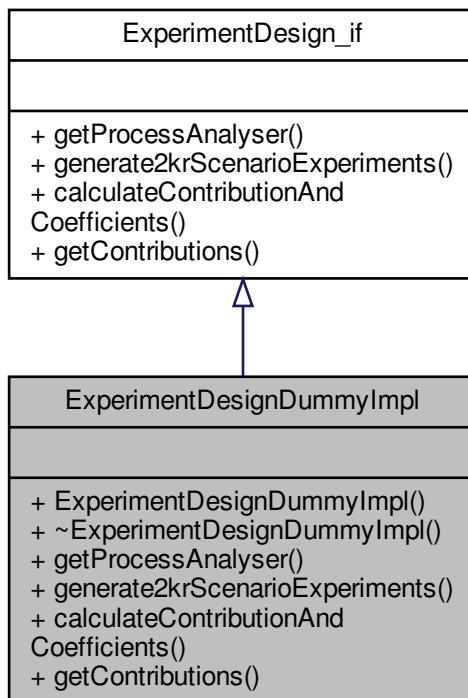
6.36 ExperimentDesignDummyImpl Class Reference

```
#include <ExperimentDesignDummyImpl.h>
```

Inheritance diagram for ExperimentDesignDummyImpl:



Collaboration diagram for ExperimentDesignDummyImpl:



Public Member Functions

- [ExperimentDesignDummyImpl \(\)](#)
- virtual [~ExperimentDesignDummyImpl \(\)=default](#)
- virtual [ProcessAnalyser_if * getProcessAnalyser \(\) const](#)
- virtual bool [generate2krScenarioExperiments \(\)](#)
- virtual bool [calculateContributionAndCoefficients \(\)](#)
- virtual std::list< [FactorOrInteractionContribution](#) * > * [getContributions \(\) const](#)

6.36.1 Detailed Description

Definition at line 22 of file [ExperimentDesignDummyImpl.h](#).

6.36.2 Constructor & Destructor Documentation

6.36.2.1 ExperimentDesignDummyImpl()

`ExperimentDesignDummyImpl::ExperimentDesignDummyImpl ()`

Definition at line 17 of file [ExperimentDesignDummyImpl.cpp](#).

6.36.2.2 ~ExperimentDesignDummyImpl()

```
virtual ExperimentDesignDummyImpl::~ExperimentDesignDummyImpl ( ) [virtual], [default]
```

6.36.3 Member Function Documentation

6.36.3.1 calculateContributionAndCoefficients()

```
bool ExperimentDesignDummyImpl::calculateContributionAndCoefficients ( ) [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 29 of file [ExperimentDesignDummyImpl.cpp](#).

6.36.3.2 generate2krScenarioExperiments()

```
bool ExperimentDesignDummyImpl::generate2krScenarioExperiments ( ) [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 25 of file [ExperimentDesignDummyImpl.cpp](#).

6.36.3.3 getContributions()

```
std::list< FactorOrInteractionContribution * > * ExperimentDesignDummyImpl::getContributions ( ) const [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 20 of file [ExperimentDesignDummyImpl.cpp](#).

6.36.3.4 getProcessAnalyser()

```
ProcessAnalyser_if * ExperimentDesignDummyImpl::getProcessAnalyser ( ) const [virtual]
```

Implements [ExperimentDesign_if](#).

Definition at line 33 of file [ExperimentDesignDummyImpl.cpp](#).

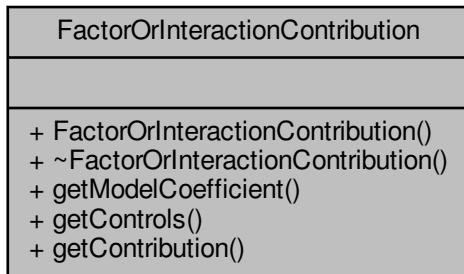
The documentation for this class was generated from the following files:

- [ExperimentDesignDummyImpl.h](#)
- [ExperimentDesignDummyImpl.cpp](#)

6.37 FactorOrInteractionContribution Class Reference

```
#include <FactorOrInteractionContribution.h>
```

Collaboration diagram for FactorOrInteractionContribution:



Public Member Functions

- [FactorOrInteractionContribution](#) (double contribution, double modelCoefficient, std::list< [SimulationControl](#) *> *controls)
- [~FactorOrInteractionContribution](#) ()
- double [getModelCoefficient](#) () const
- std::list< [SimulationControl](#) *> * [getControls](#) () const
- double [getContribution](#) () const

6.37.1 Detailed Description

This simple class corresponds to a factor when it refers to just one [SimulationControl](#), or to the interaction between two or more factors when it refers to more [SimulationControl](#). It also encapsulates the contribution of the factor or interaction and its coefficient in the full model that estimates one specific [SimulationResponse](#).

Definition at line 23 of file [FactorOrInteractionContribution.h](#).

6.37.2 Constructor & Destructor Documentation

6.37.2.1 FactorOrInteractionContribution()

```
FactorOrInteractionContribution::FactorOrInteractionContribution (
    double contribution,
    double modelCoefficient,
    std::list< SimulationControl *> * controls )
```

Definition at line 16 of file [FactorOrInteractionContribution.cpp](#).

6.37.2.2 ~FactorOrInteractionContribution()

```
FactorOrInteractionContribution::~FactorOrInteractionContribution ( )
```

6.37.3 Member Function Documentation

6.37.3.1 getContribution()

```
double FactorOrInteractionContribution::getContribution ( ) const
```

Definition at line 31 of file [FactorOrInteractionContribution.cpp](#).

6.37.3.2 getControls()

```
std::list< SimulationControl * > * FactorOrInteractionContribution::getControls ( ) const
```

Definition at line 27 of file [FactorOrInteractionContribution.cpp](#).

6.37.3.3 getModelCoefficient()

```
double FactorOrInteractionContribution::getModelCoefficient ( ) const
```

Definition at line 23 of file [FactorOrInteractionContribution.cpp](#).

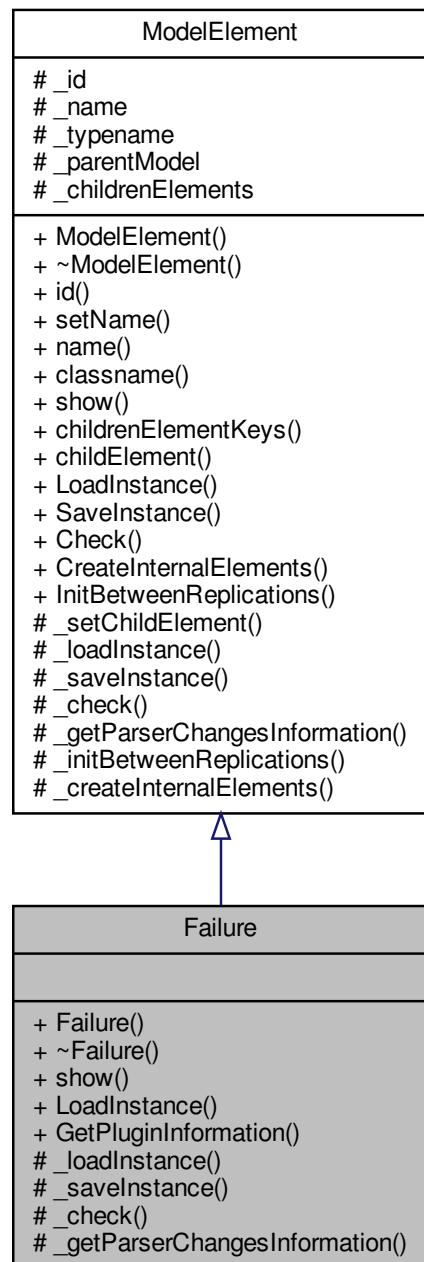
The documentation for this class was generated from the following files:

- [FactorOrInteractionContribution.h](#)
- [FactorOrInteractionContribution.cpp](#)

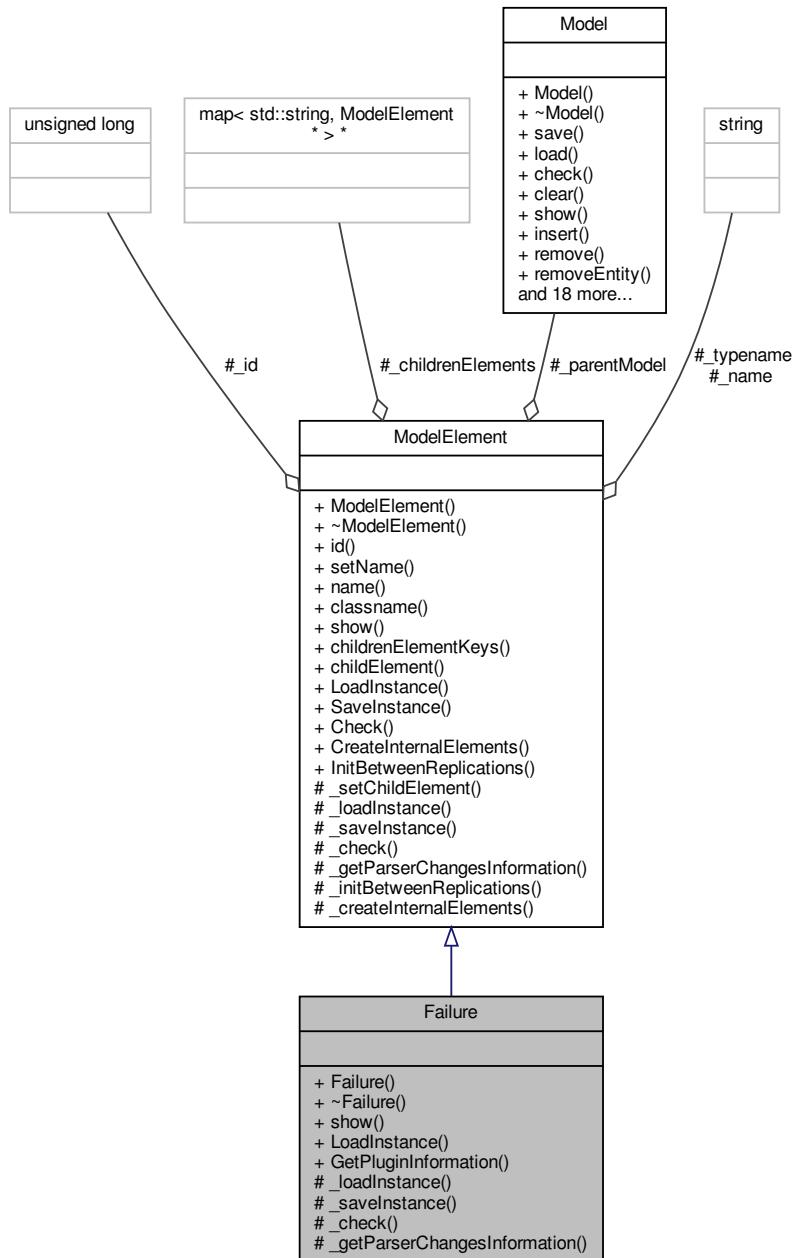
6.38 Failure Class Reference

```
#include <Failure.h>
```

Inheritance diagram for Failure:



Collaboration diagram for Failure:



Public Member Functions

- **Failure** (`Model *model, std::string name=""`)
- virtual **~Failure** ()=default
- virtual std::string **show** ()

Static Public Member Functions

- static **ModelElement *** **LoadInstance** (`Model *model, std::map< std::string, std::string *> *fields`)
- static **PluginInformation *** **GetPluginInformation** ()

Protected Member Functions

- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)
- virtual ParserChangesInformation * `_getParserChangesInformation` ()

Additional Inherited Members

6.38.1 Detailed Description

Failure module DESCRIPTION The **Failure** module is designed for use with resources. When a failure occurs, the entire resource (regardless of its capacity) is failed. Failures are designed to be used with single-capacity resources or with multiple-capacity resources whose individual resource units all fail at the same time. TYPICAL USES Breakdown information for a machine Cash register tape refill every “x” customers Random computer shutdowns or restarts PROMPTS Recordset Name of the recordset in the specified file from which to read values. This field is available only if you specify a **File** Name with a file access type, path, and recordset. Arena uses the Rows and Columns properties to determine the amount of data to read from the recordset. A recordset is required for all file types except .xml. The recordset size must be equal to or greater than the number of rows and columns specified for the expression. Expression Values Lists the value or values of the expression. This property is not available if you specify a **File** Name from which to read expression values. Expression Value Expression value associated with the expression name. Prompt Description Name The name of the failure associated with one or more resources. Type Determines if the failure is time-based or count-based. Count Defines the number of resource releases for count-based failures. Valid when the Type is Count. Up Time Defines the time between failures for time-based failures. Valid when the Type is Time. Up Time Units Time units for the time between failures (Up Time) for timebased failures. Down Time Defines the duration of the failure. Down Time Units Time units for the duration of the failure (Down Time). Uptime in this State only Defines the state that should be considered for the time between failures (only for time-based failures). If state is not specified, then all states are considered (that is, the time between failures does not depend on the time spent in a specific state, but rather on the total simulation time). For example, you might want to define a failure to be based only on the state Busy, and therefore, the time between downtimes would be based on the amount of time that a resource is busy, not simulated clock time.

Definition at line 67 of file [Failure.h](#).

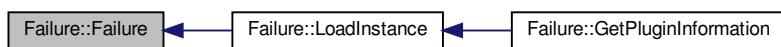
6.38.2 Constructor & Destructor Documentation

6.38.2.1 Failure()

```
Failure::Failure (
    Model * model,
    std::string name = "" )
```

Definition at line 16 of file [Failure.cpp](#).

Here is the caller graph for this function:



6.38.2.2 ~Failure()

```
virtual Failure::~Failure ( ) [virtual], [default]
```

6.38.3 Member Function Documentation

6.38.3.1 _check()

```
bool Failure::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Failure.cpp](#).

6.38.3.2 _getParserChangesInformation()

```
ParserChangesInformation * Failure::_getParserChangesInformation ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 64 of file [Failure.cpp](#).

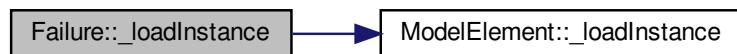
6.38.3.3 _loadInstance()

```
bool Failure::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

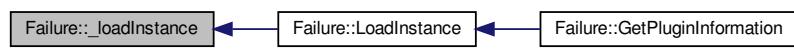
Reimplemented from [ModelElement](#).

Definition at line 39 of file [Failure.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



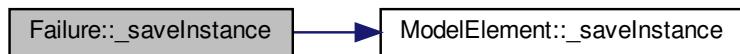
6.38.3.4 `_saveInstance()`

```
std::map< std::string, std::string > * Failure::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 51 of file [Failure.cpp](#).

Here is the call graph for this function:

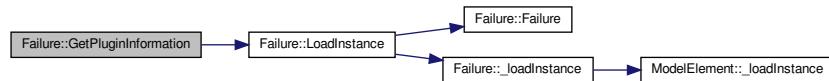


6.38.3.5 `GetPluginInformation()`

```
PluginInformation * Failure::GetPluginInformation ( ) [static]
```

Definition at line 24 of file [Failure.cpp](#).

Here is the call graph for this function:

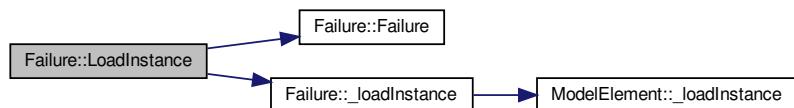


6.38.3.6 `LoadInstance()`

```
ModelElement * Failure::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 29 of file [Failure.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



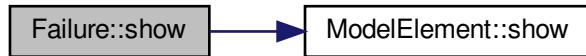
6.38.3.7 show()

```
std::string Failure::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 19 of file [Failure.cpp](#).

Here is the call graph for this function:



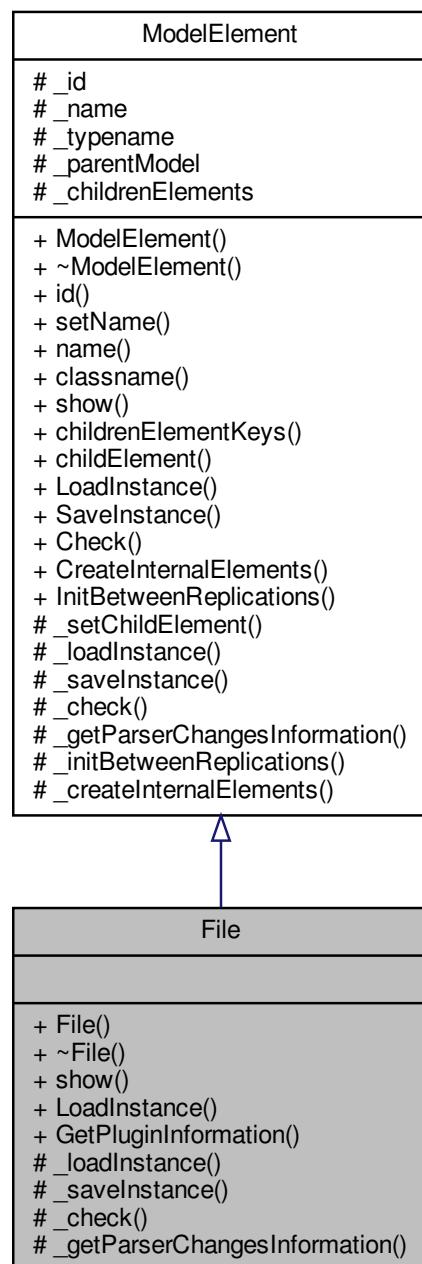
The documentation for this class was generated from the following files:

- [Failure.h](#)
- [Failure.cpp](#)

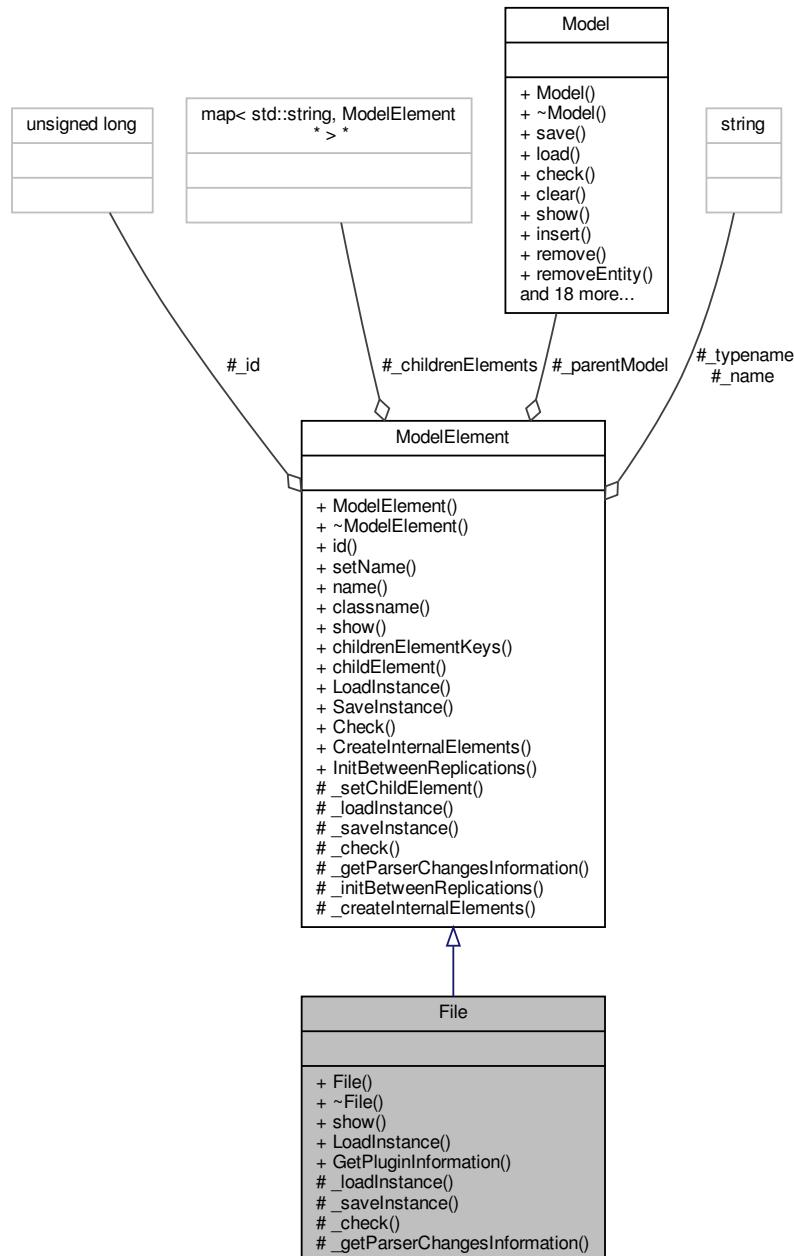
6.39 File Class Reference

```
#include <File.h>
```

Inheritance diagram for File:



Collaboration diagram for File:



Public Member Functions

- **File (Model *model, std::string name="")**
- virtual **~File ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static **ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)**
- static **PluginInformation * GetPluginInformation ()**

Protected Member Functions

- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)
- virtual `ParserChangesInformation` * `_getParserChangesInformation` ()

Additional Inherited Members

6.39.1 Detailed Description

File module DESCRIPTION Use the `File` module to access external files for the `ReadWrite` module, `Variable` module, and `Expression` module. The `File` module identifies the system file name and defines the access type and operational characteristics of the file. TYPICAL USES `File` containing predefined airline flight data `File` specifying customer order times and relevant information `File` to write user model configuration data from menu input PR← OMPTS Prompt Description Name The name of the file whose characteristics are being defined. This name must be unique. `Access` Type The file type. Operating System `File` Name Name of the actual file that is being read from or to which it is being written. Connecting String Connection string used to open ADO connection to the data source. Structure `File` structure, which can be unformatted, free format, or a specific C or FORTRAN format. End of `File` Action Type of action to occur if an end of file condition is reached. Initialize Option Action to be taken on file at beginning of each simulation replication. Comment Character Character indicating comment record. Recordset Name Name used to identify the recordset in the `Expression`, `ReadWrite`, and `Variable` modules. This name must be unique within the file. This field is available for Microsoft Excel, Microsoft Excel 2007, Microsoft `Access`, Microsoft `Access` 2007, and ActiveX Data Objects files. CommandText Text of the command that will be used to open the recordset (for example, SQL statement, procedure name, table name.) This field is available for ActiveX Data Object files only. CommandType Type of command entered in the `CommandText`. Named Range The named range in the Excel workbook to which the recordset refers. Table Name The name of the table in the `Access` database to which the recordset refers.

Definition at line 64 of file `File.h`.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 `File()`

```
File::File (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file `File.cpp`.

Here is the caller graph for this function:



6.39.2.2 ~File()

```
virtual File::~File ( ) [virtual], [default]
```

6.39.3 Member Function Documentation

6.39.3.1 _check()

```
bool File::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [60](#) of file [File.cpp](#).

6.39.3.2 _getParserChangesInformation()

```
ParserChangesInformation * File::_getParserChangesInformation ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [66](#) of file [File.cpp](#).

6.39.3.3 _loadInstance()

```
bool File::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [41](#) of file [File.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.39.3.4 `_saveInstance()`

```
std::map< std::string, std::string > * File::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 53 of file [File.cpp](#).

Here is the call graph for this function:

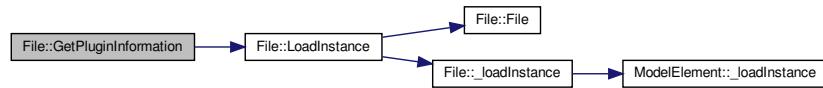


6.39.3.5 `GetPluginInformation()`

```
PluginInformation * File::GetPluginInformation ( ) [static]
```

Definition at line 26 of file [File.cpp](#).

Here is the call graph for this function:

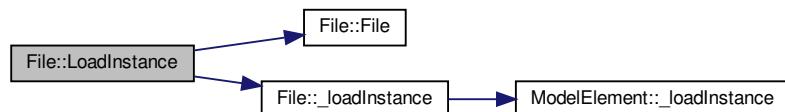


6.39.3.6 `LoadInstance()`

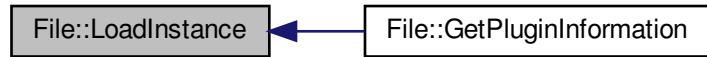
```
ModelElement * File::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 31 of file [File.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.39.3.7 show()

```
std::string File::show () [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [21](#) of file [File.cpp](#).

Here is the call graph for this function:



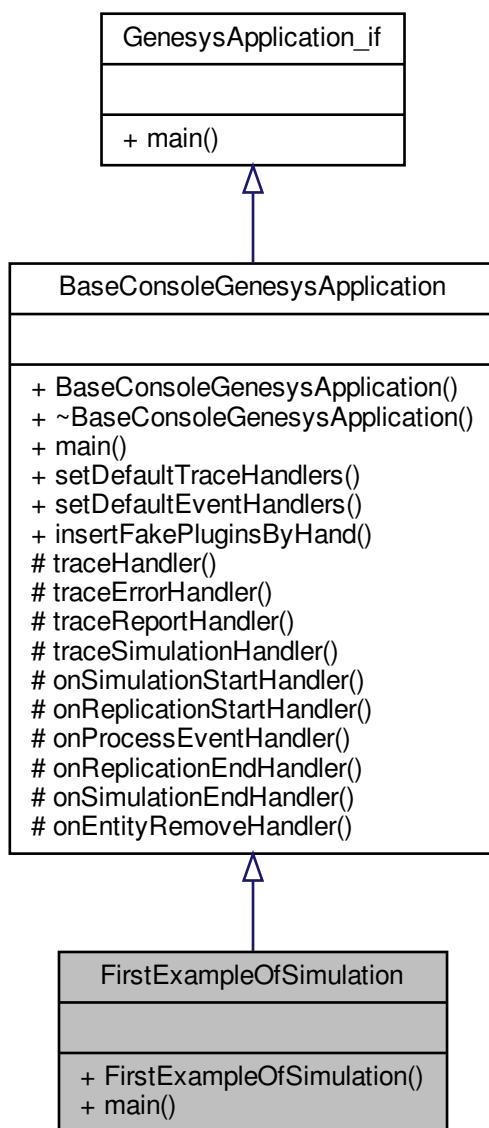
The documentation for this class was generated from the following files:

- [File.h](#)
- [File.cpp](#)

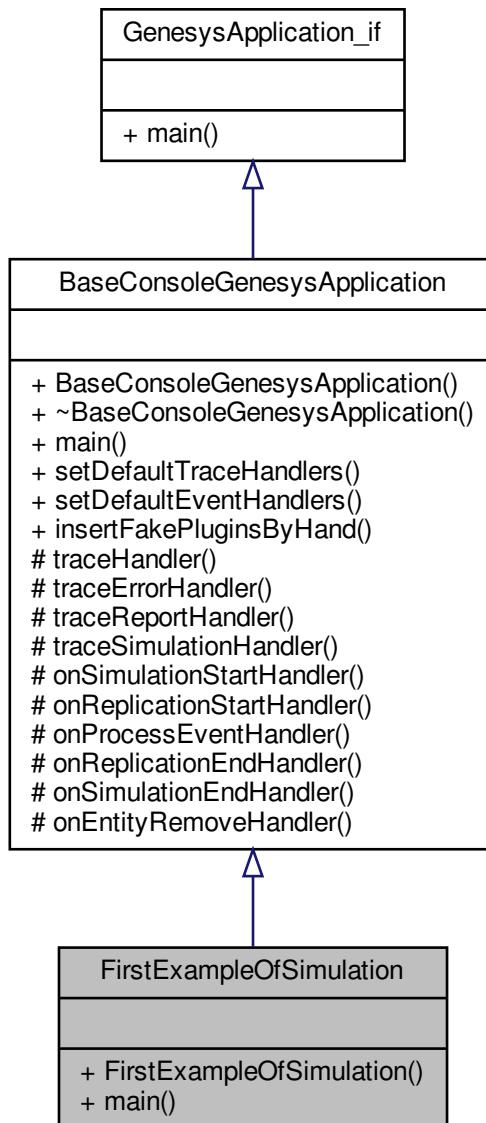
6.40 FirstExampleOfSimulation Class Reference

```
#include <FirstExampleOfSimulation.h>
```

Inheritance diagram for FirstExampleOfSimulation:



Collaboration diagram for FirstExampleOfSimulation:



Public Member Functions

- [FirstExampleOfSimulation \(\)](#)
- virtual int [main \(int argc, char **argv\)](#)

Additional Inherited Members

6.40.1 Detailed Description

Definition at line 19 of file [FirstExampleOfSimulation.h](#).

6.40.2 Constructor & Destructor Documentation

6.40.2.1 FirstExampleOfSimulation()

```
FirstExampleOfSimulation::FirstExampleOfSimulation ( )
```

Definition at line 29 of file [FirstExampleOfSimulation.cpp](#).

6.40.3 Member Function Documentation

6.40.3.1 main()

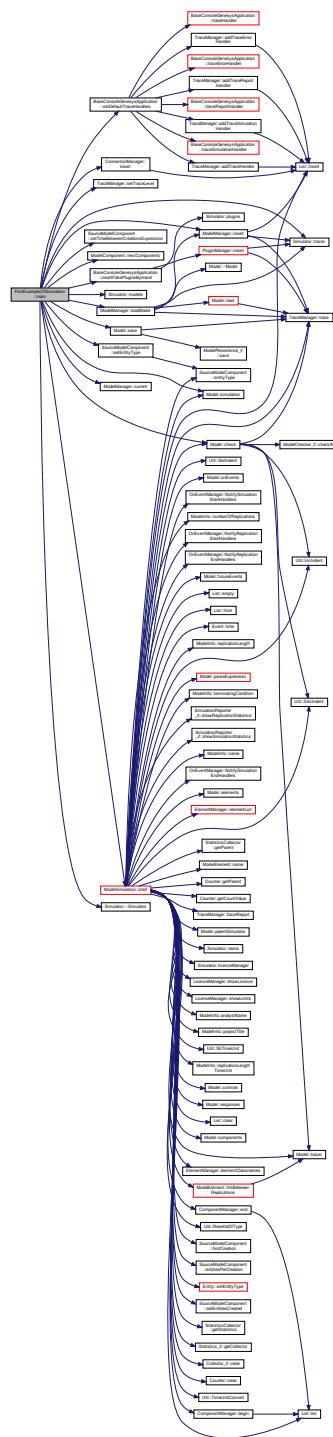
```
int FirstExampleOfSimulation::main (
    int argc,
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

Definition at line 36 of file [FirstExampleOfSimulation.cpp](#).

Here is the call graph for this function:



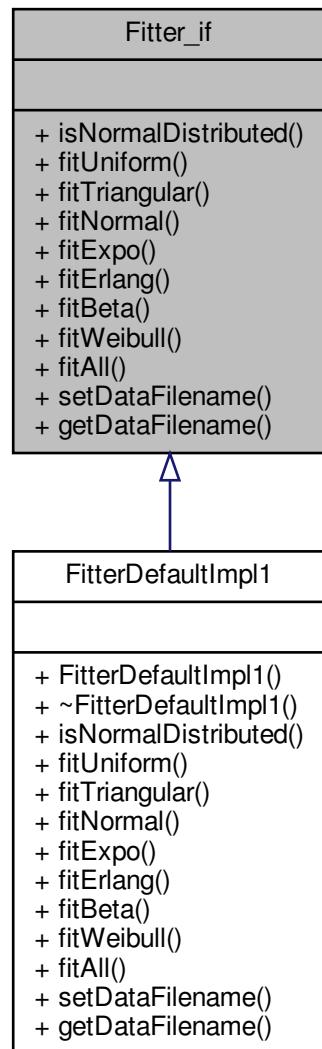
The documentation for this class was generated from the following files:

- [FirstExampleOfSimulation.h](#)
- [FirstExampleOfSimulation.cpp](#)

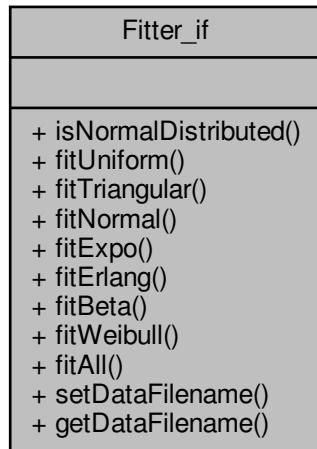
6.41 Fitter_if Class Reference

```
#include <Fitter_if.h>
```

Inheritance diagram for Fitter_if:



Collaboration diagram for Fitter_if:



Public Member Functions

- virtual bool `isNormalDistributed` (double confidencelevel)=0
- virtual void `fitUniform` (double *sqrerror, double *min, double *max)=0
- virtual void `fitTriangular` (double *sqrerror, double *min, double *mo, double *max)=0
- virtual void `fitNormal` (double *sqrerror, double *avg, double *stddev)=0
- virtual void `fitExpo` (double *sqrerror, double *avg1)=0
- virtual void `fitErlang` (double *sqrerror, double *avg, double *m)=0
- virtual void `fitBeta` (double *sqrerror, double *alpha, double *beta, double *infLimit, double *supLimit)=0
- virtual void `fitWeibull` (double *sqrerror, double *alpha, double *scale)=0
- virtual void `fitAll` (double *sqrerror, std::string *name)=0
- virtual void `setDataFilename` (std::string dataFilename)=0
- virtual std::string `getDataFilename` ()=0

6.41.1 Detailed Description

Definition at line 19 of file [Fitter_if.h](#).

6.41.2 Member Function Documentation

6.41.2.1 fitAll()

```
virtual void Fitter_if::fitAll (
    double * sgrerror,
    std::string * name ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

Here is the caller graph for this function:



6.41.2.2 fitBeta()

```
virtual void Fitter_if::fitBeta (
    double * sgrerror,
    double * alpha,
    double * beta,
    double * infLimit,
    double * supLimit ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

6.41.2.3 fitErlang()

```
virtual void Fitter_if::fitErlang (
    double * sgrerror,
    double * avg,
    double * m ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

6.41.2.4 fitExpo()

```
virtual void Fitter_if::fitExpo (
    double * sgrerror,
    double * avg1 ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

6.41.2.5 fitNormal()

```
virtual void Fitter_if::fitNormal (
    double * sgrerror,
    double * avg,
    double * stddev ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

Here is the caller graph for this function:

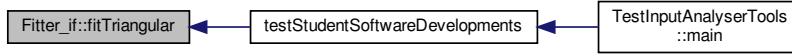


6.41.2.6 fitTriangular()

```
virtual void Fitter_if::fitTriangular (
    double * sgrerror,
    double * min,
    double * mo,
    double * max ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

Here is the caller graph for this function:

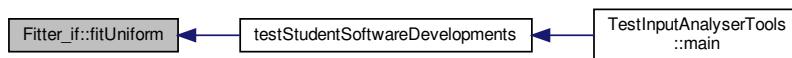


6.41.2.7 fitUniform()

```
virtual void Fitter_if::fitUniform (
    double * sgrerror,
    double * min,
    double * max ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

Here is the caller graph for this function:



6.41.2.8 fitWeibull()

```
virtual void Fitter_if::fitWeibull (
    double * sqrerror,
    double * alpha,
    double * scale ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

6.41.2.9 getDataFilename()

```
virtual std::string Fitter_if::getDataFilename () [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

6.41.2.10 isNormalDistributed()

```
virtual bool Fitter_if::isNormalDistributed (
    double confidencelevel ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

Here is the caller graph for this function:



6.41.2.11 setDataFilename()

```
virtual void Fitter_if::setDataFilename (
    std::string datafilename ) [pure virtual]
```

Implemented in [FitterDefaultImpl1](#).

Here is the caller graph for this function:



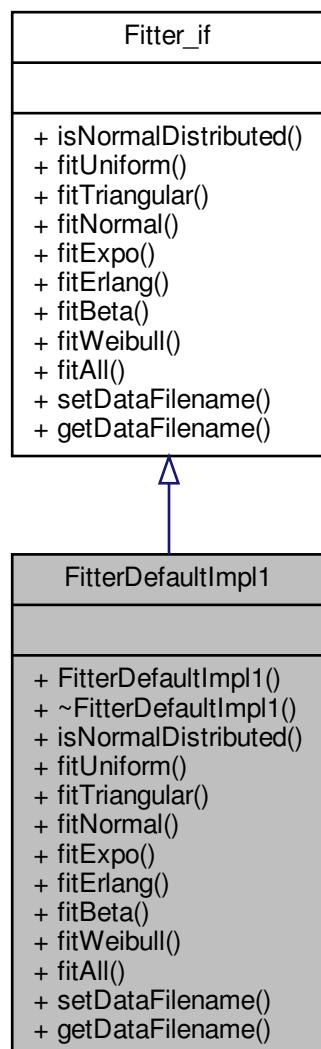
The documentation for this class was generated from the following file:

- [Fitter_if.h](#)

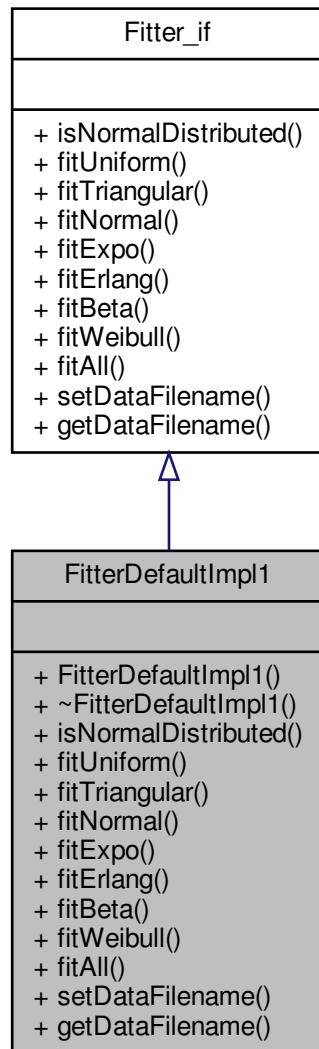
6.42 FitterDefaultImpl1 Class Reference

```
#include <FitterDefaultImpl1.h>
```

Inheritance diagram for FitterDefaultImpl1:



Collaboration diagram for FitterDefaultImpl1:



Public Member Functions

- `FitterDefaultImpl1 \(\)`
- `virtual ~FitterDefaultImpl1 \(\) =default`
- `bool isNormalDistributed (double confidencelevel)`
- `void fitUniform (double *sqrerror, double *min, double *max)`
- `void fitTriangular (double *sqrerror, double *min, double *mo, double *max)`
- `void fitNormal (double *sqrerror, double *avg, double *stddev)`
- `void fitExpo (double *sqrerror, double *avg1)`
- `void fitErlang (double *sqrerror, double *avg, double *m)`
- `void fitBeta (double *sqrerror, double *alpha, double *beta, double *infLimit, double *supLimit)`
- `void fitWeibull (double *sqrerror, double *alpha, double *scale)`
- `void fitAll (double *sqrerror, std::string *name)`
- `void setDataFilename (std::string dataFilename)`
- `std::string getDataFilename \(\)`

6.42.1 Detailed Description

Definition at line 19 of file [FitterDefaultImpl1.h](#).

6.42.2 Constructor & Destructor Documentation

6.42.2.1 FitterDefaultImpl1()

```
FitterDefaultImpl1::FitterDefaultImpl1 ( )
```

Definition at line 16 of file [FitterDefaultImpl1.cpp](#).

6.42.2.2 ~FitterDefaultImpl1()

```
virtual FitterDefaultImpl1::~FitterDefaultImpl1 ( ) [virtual], [default]
```

6.42.3 Member Function Documentation

6.42.3.1 fitAll()

```
void FitterDefaultImpl1::fitAll (
    double * sgrerror,
    std::string * name ) [virtual]
```

Implements [Fitter_if](#).

Definition at line 51 of file [FitterDefaultImpl1.cpp](#).

6.42.3.2 fitBeta()

```
void FitterDefaultImpl1::fitBeta (
    double * sgrerror,
    double * alpha,
    double * beta,
    double * infLimit,
    double * supLimit ) [virtual]
```

Implements [Fitter_if](#).

Definition at line 43 of file [FitterDefaultImpl1.cpp](#).

6.42.3.3 fitErlang()

```
void FitterDefaultImpl1::fitErlang (
    double * sqrerror,
    double * avg,
    double * m ) [virtual]
```

Implements [Fitter_if](#).

Definition at line 39 of file [FitterDefaultImpl1.cpp](#).

6.42.3.4 fitExpo()

```
void FitterDefaultImpl1::fitExpo (
    double * sqrerror,
    double * avg1 ) [virtual]
```

Implements [Fitter_if](#).

Definition at line 36 of file [FitterDefaultImpl1.cpp](#).

6.42.3.5 fitNormal()

```
void FitterDefaultImpl1::fitNormal (
    double * sqrerror,
    double * avg,
    double * stddev ) [virtual]
```

Implements [Fitter_if](#).

Definition at line 32 of file [FitterDefaultImpl1.cpp](#).

6.42.3.6 fitTriangular()

```
void FitterDefaultImpl1::fitTriangular (
    double * sqrerror,
    double * min,
    double * mo,
    double * max ) [virtual]
```

Implements [Fitter_if](#).

Definition at line 28 of file [FitterDefaultImpl1.cpp](#).

6.42.3.7 fitUniform()

```
void FitterDefaultImpl1::fitUniform (
    double * sqrerror,
    double * min,
    double * max ) [virtual]
```

Implements [Fitter_if](#).

Definition at line [24](#) of file [FitterDefaultImpl1.cpp](#).

6.42.3.8 fitWeibull()

```
void FitterDefaultImpl1::fitWeibull (
    double * sqrerror,
    double * alpha,
    double * scale ) [virtual]
```

Implements [Fitter_if](#).

Definition at line [47](#) of file [FitterDefaultImpl1.cpp](#).

6.42.3.9 getDataFilename()

```
std::string FitterDefaultImpl1::getDataFilename () [virtual]
```

Implements [Fitter_if](#).

Definition at line [59](#) of file [FitterDefaultImpl1.cpp](#).

6.42.3.10 isNormalDistributed()

```
bool FitterDefaultImpl1::isNormalDistributed (
    double confidencelevel ) [virtual]
```

Implements [Fitter_if](#).

Definition at line [20](#) of file [FitterDefaultImpl1.cpp](#).

6.42.3.11 setDataFilename()

```
void FitterDefaultImpl1::setDataFilename (
    std::string datafilename ) [virtual]
```

Implements [Fitter_if](#).

Definition at line [55](#) of file [FitterDefaultImpl1.cpp](#).

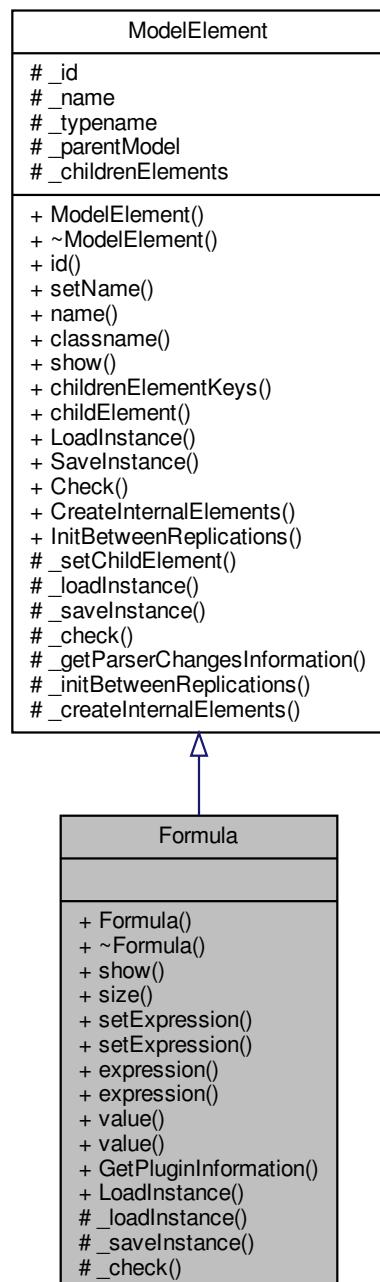
The documentation for this class was generated from the following files:

- [FitterDefaultImpl1.h](#)
- [FitterDefaultImpl1.cpp](#)

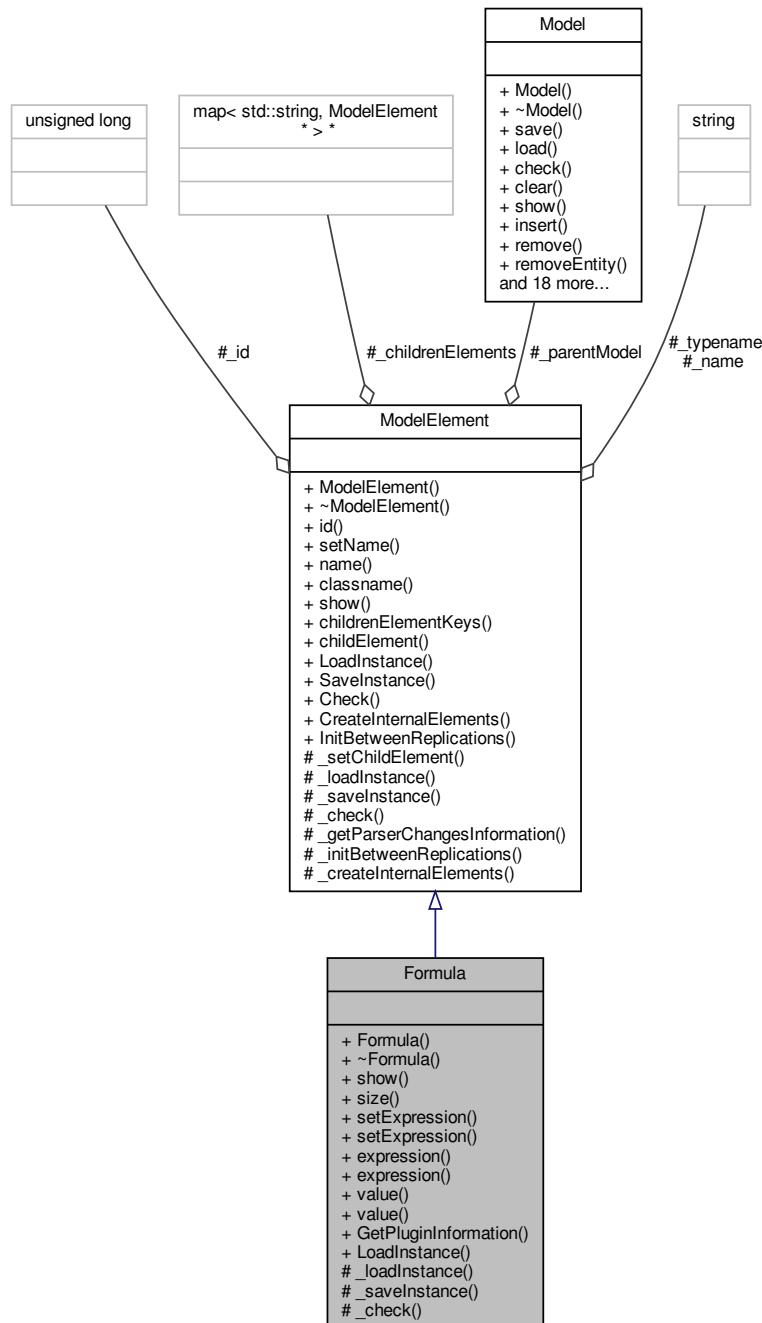
6.43 Formula Class Reference

```
#include <Formula.h>
```

Inheritance diagram for Formula:



Collaboration diagram for Formula:



Public Member Functions

- **Formula (Model *model, std::string name="")**
- virtual **~Formula ()=default**
- virtual std::string **show ()**
- unsigned int **size ()**
- void **setExpression (std::string index, std::string formulaExpression)**

- void `setExpression` (std::string formulaExpression)
- std::string `expression` (std::string index)
- std::string `expression` ()
- double `value` ()
- double `value` (std::string index)

Static Public Member Functions

- static `PluginInformation * GetPluginInformation` ()
- static `ModelElement * LoadInstance` (`Model` *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)

Additional Inherited Members

6.43.1 Detailed Description

Definition at line 22 of file [Formula.h](#).

6.43.2 Constructor & Destructor Documentation

6.43.2.1 `Formula()`

```
Formula::Formula (
    Model * model,
    std::string name = "")
```

Definition at line 20 of file [Formula.cpp](#).

Here is the caller graph for this function:



6.43.2.2 `~Formula()`

```
virtual Formula::~Formula ( ) [virtual], [default]
```

6.43.3 Member Function Documentation

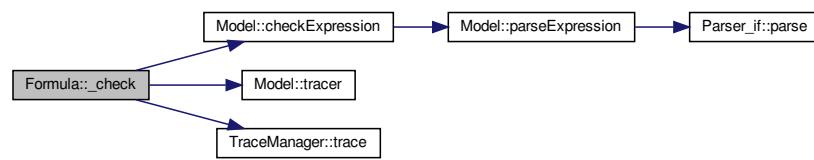
6.43.3.1 `_check()`

```
bool Formula::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 103 of file [Formula.cpp](#).

Here is the call graph for this function:



6.43.3.2 `_loadInstance()`

```
bool Formula::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 93 of file [Formula.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



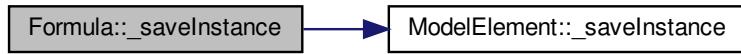
6.43.3.3 `_saveInstance()`

```
std::map< std::string, std::string > * Formula::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 97 of file [Formula.cpp](#).

Here is the call graph for this function:



6.43.3.4 `expression()` [1/2]

```
std::string Formula::expression ( std::string index )
```

Definition at line 46 of file [Formula.cpp](#).

Here is the caller graph for this function:

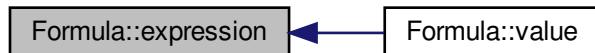


6.43.3.5 `expression()` [2/2]

```
std::string Formula::expression ( )
```

Definition at line 59 of file [Formula.cpp](#).

Here is the caller graph for this function:

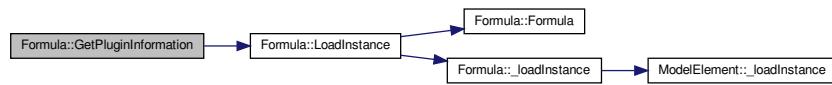


6.43.3.6 GetPluginInformation()

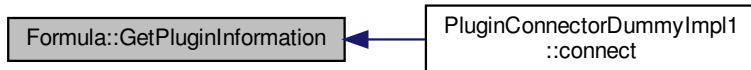
```
PluginInformation * Formula::GetPluginInformation ( ) [static]
```

Definition at line 78 of file [Formula.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

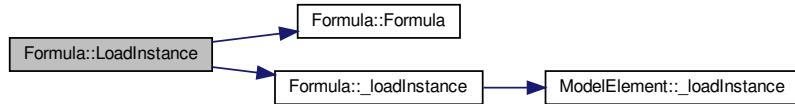


6.43.3.7 LoadInstance()

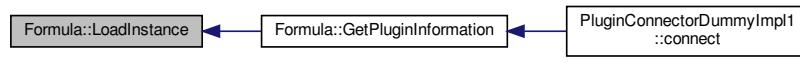
```
ModelElement * Formula::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 83 of file [Formula.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

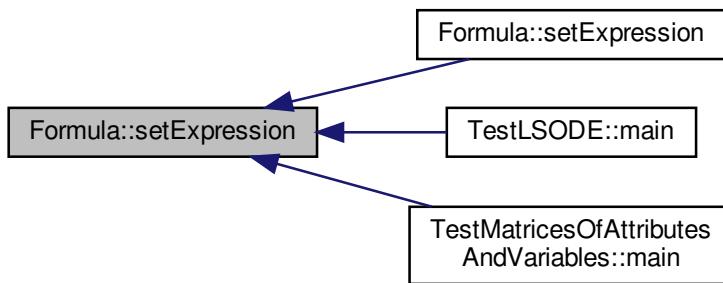


6.43.3.8 setExpression() [1/2]

```
void Formula::setExpression (
    std::string index,
    std::string formulaExpression )
```

Definition at line 37 of file [Formula.cpp](#).

Here is the caller graph for this function:

**6.43.3.9 setExpression() [2/2]**

```
void Formula::setExpression (
    std::string formulaExpression )
```

Definition at line 55 of file [Formula.cpp](#).

Here is the call graph for this function:



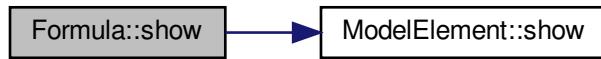
6.43.3.10 show()

```
std::string Formula::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [24](#) of file [Formula.cpp](#).

Here is the call graph for this function:



6.43.3.11 size()

```
unsigned int Formula::size ( )
```

Definition at line [33](#) of file [Formula.cpp](#).

Here is the caller graph for this function:



6.43.3.12 value() [1/2]

```
double Formula::value ( )
```

Definition at line [74](#) of file [Formula.cpp](#).

Here is the caller graph for this function:

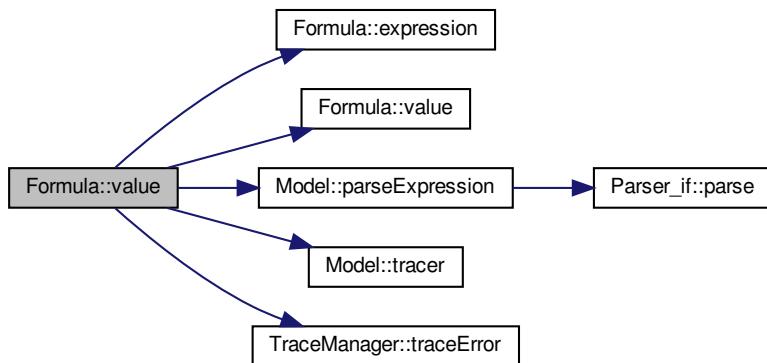


6.43.3.13 value() [2/2]

```
double Formula::value (
    std::string index )
```

Definition at line 63 of file [Formula.cpp](#).

Here is the call graph for this function:



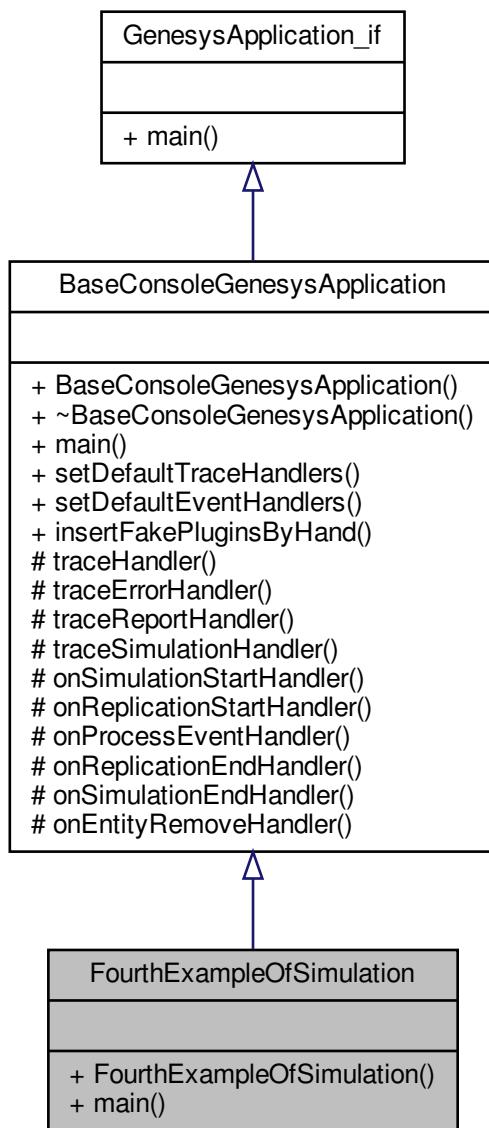
The documentation for this class was generated from the following files:

- [Formula.h](#)
- [Formula.cpp](#)

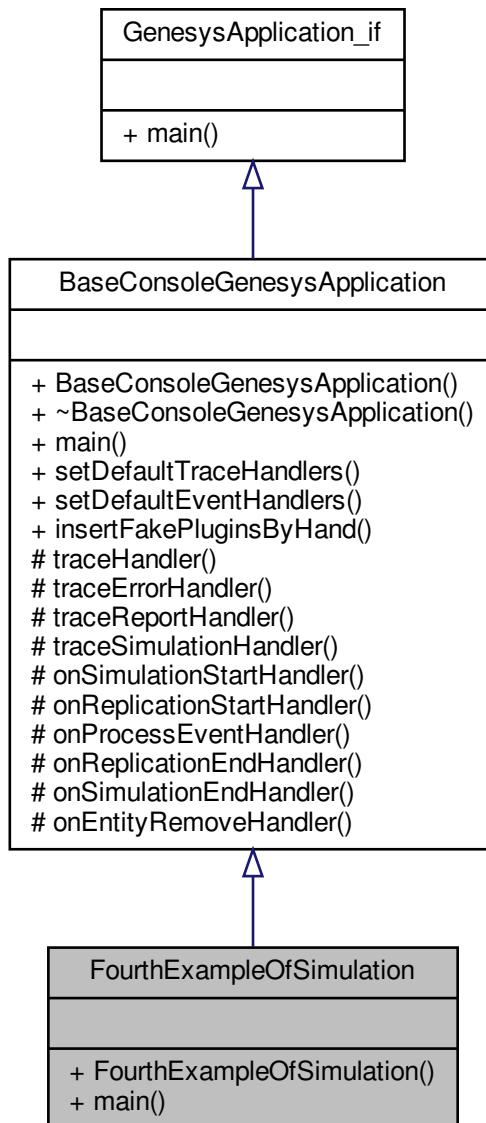
6.44 FourthExampleOfSimulation Class Reference

```
#include <FourthExampleOfSimulation.h>
```

Inheritance diagram for FourthExampleOfSimulation:



Collaboration diagram for FourthExampleOfSimulation:



Public Member Functions

- [FourthExampleOfSimulation \(\)](#)
- virtual int [main \(int argc, char **argv\)](#)

Additional Inherited Members

6.44.1 Detailed Description

Definition at line 19 of file [FourthExampleOfSimulation.h](#).

6.44.2 Constructor & Destructor Documentation

6.44.2.1 FourthExampleOfSimulation()

```
FourthExampleOfSimulation::FourthExampleOfSimulation ( )
```

Definition at line 39 of file [FourthExampleOfSimulation.cpp](#).

6.44.3 Member Function Documentation

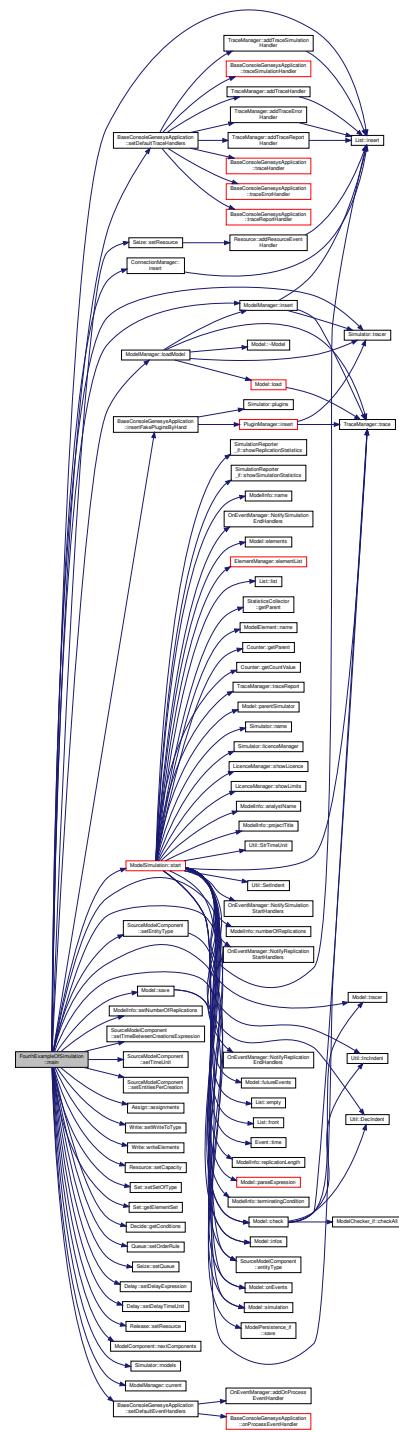
6.44.3.1 main()

```
int FourthExampleOfSimulation::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 42 of file [FourthExampleOfSimulation.cpp](#).

Here is the call graph for this function:



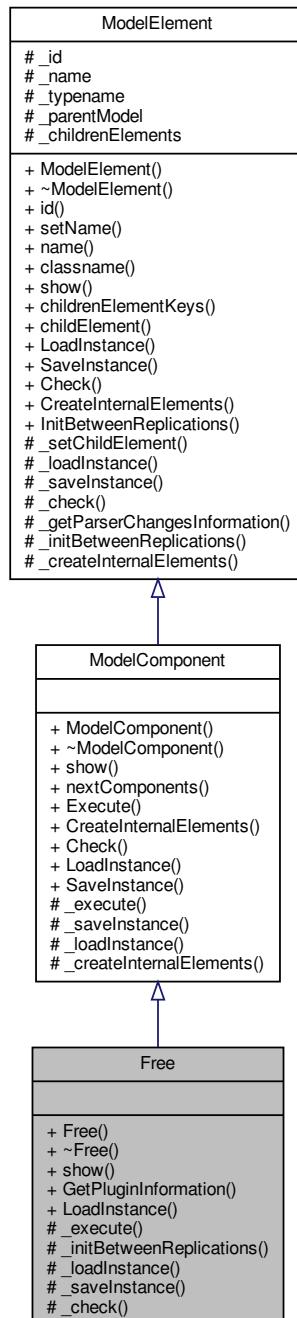
The documentation for this class was generated from the following files:

- [FourthExampleOfSimulation.h](#)
- [FourthExampleOfSimulation.cpp](#)

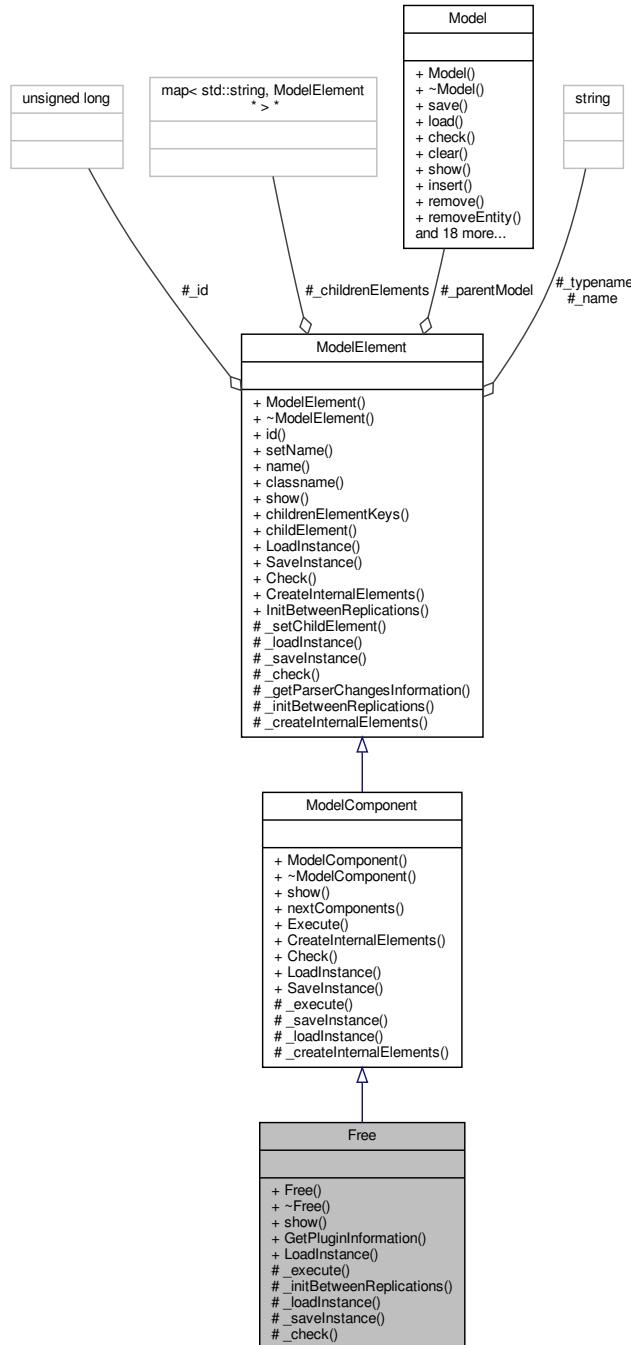
6.45 Free Class Reference

```
#include <Free.h>
```

Inheritance diagram for Free:



Collaboration diagram for Free:



Public Member Functions

- `Free (Model *model, std::string name="")`
- virtual `~Free ()=default`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.45.1 Detailed Description

Free module DESCRIPTION The `Free` module releases the entity's most recently allocated transporter unit. If another entity is waiting in a queue to request or allocate the transporter, the transporter will be given to that entity. If there are no waiting entities at the time the transporter unit is freed, the transporter will wait idle at the freeing entity's station location, unless otherwise specified in the Transporter module. TYPICAL USES A part awaiting a shipping truck frees its forklift An airport transfer cart completes its trip PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Transporter Name Name of the transporter to free. A blank value assumes the most recently allocated or requested transporter. Unit Number Determines which of the transporter units in the transporter set to free.

Definition at line 39 of file `Free.h`.

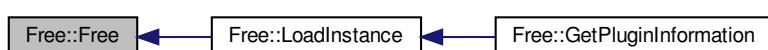
6.45.2 Constructor & Destructor Documentation

6.45.2.1 Free()

```
Free::Free (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file `Free.cpp`.

Here is the caller graph for this function:



6.45.2.2 ~Free()

```
virtual Free::~Free ( ) [virtual], [default]
```

6.45.3 Member Function Documentation

6.45.3.1 _check()

```
bool Free::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Free.cpp](#).

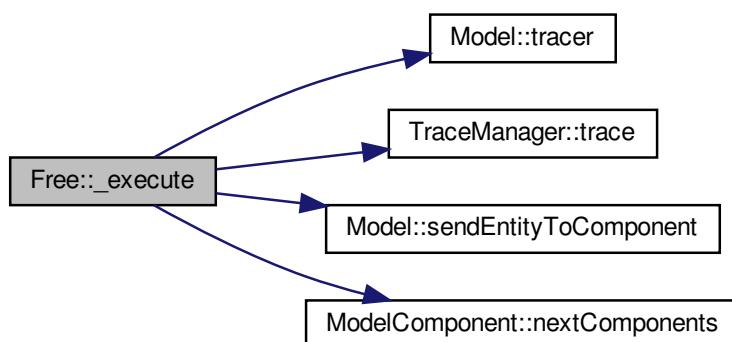
6.45.3.2 _execute()

```
void Free::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Free.cpp](#).

Here is the call graph for this function:



6.45.3.3 `_initBetweenReplications()`

```
void Free::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Free.cpp](#).

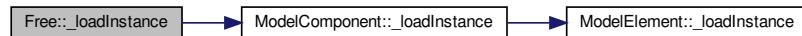
6.45.3.4 `_loadInstance()`

```
bool Free::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

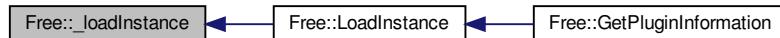
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Free.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



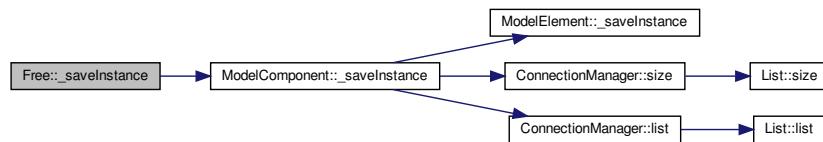
6.45.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Free::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Free.cpp](#).

Here is the call graph for this function:



6.45.3.6 GetPluginInformation()

```
PluginInformation * Free::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Free.cpp](#).

Here is the call graph for this function:

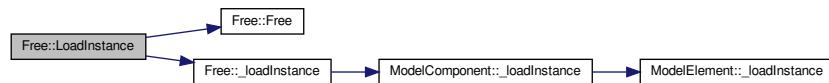


6.45.3.7 LoadInstance()

```
ModelComponent * Free::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Free.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.45.3.8 show()

```
std::string Free::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [Free.cpp](#).

Here is the call graph for this function:



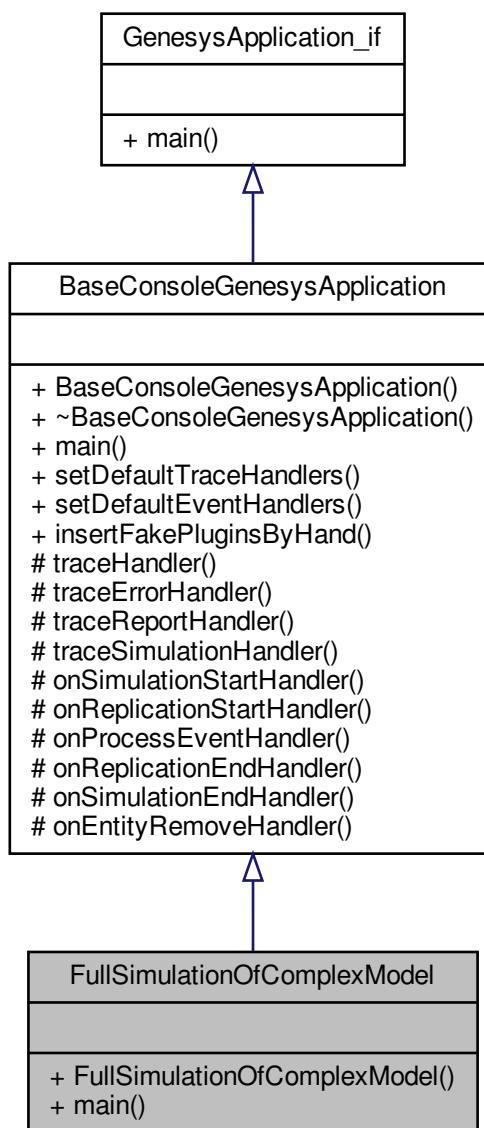
The documentation for this class was generated from the following files:

- [Free.h](#)
- [Free.cpp](#)

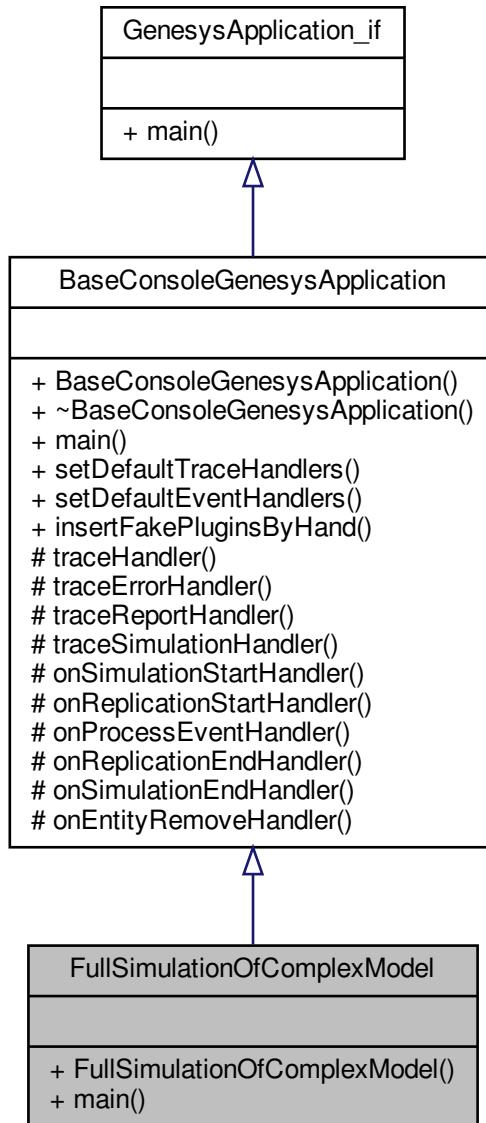
6.46 FullSimulationOfComplexModel Class Reference

```
#include <FullSimulationOfComplexModel.h>
```

Inheritance diagram for FullSimulationOfComplexModel:



Collaboration diagram for FullSimulationOfComplexModel:



Public Member Functions

- `FullSimulationOfComplexModel ()`
- `virtual int main (int argc, char **argv)`

Additional Inherited Members

6.46.1 Detailed Description

Definition at line 20 of file [FullSimulationOfComplexModel.h](#).

6.46.2 Constructor & Destructor Documentation

6.46.2.1 FullSimulationOfComplexModel()

```
FullSimulationOfComplexModel::FullSimulationOfComplexModel ( )
```

Definition at line 32 of file [FullSimulationOfComplexModel.cpp](#).

6.46.3 Member Function Documentation

6.46.3.1 main()

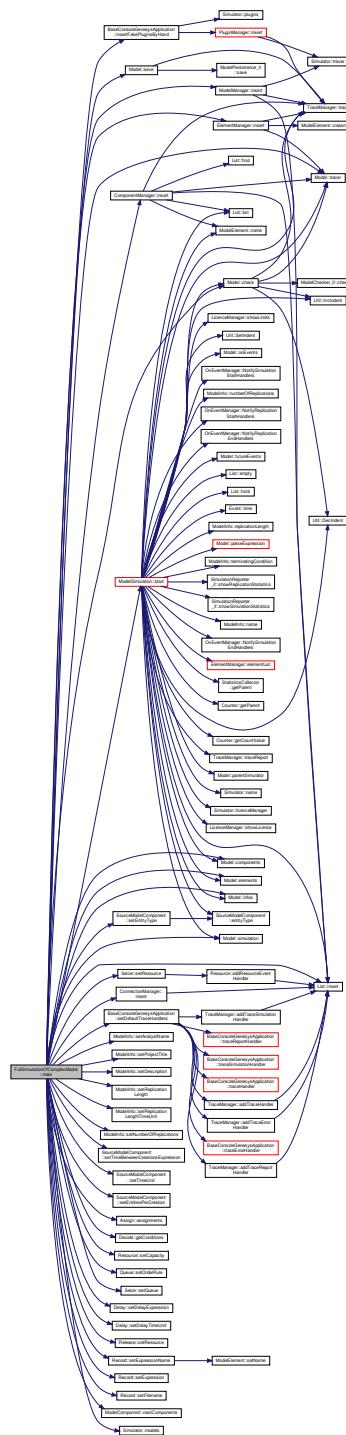
```
int FullSimulationOfComplexModel::main (
    int argc,
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

Definition at line 40 of file [FullSimulationOfComplexModel.cpp](#).

Here is the call graph for this function:



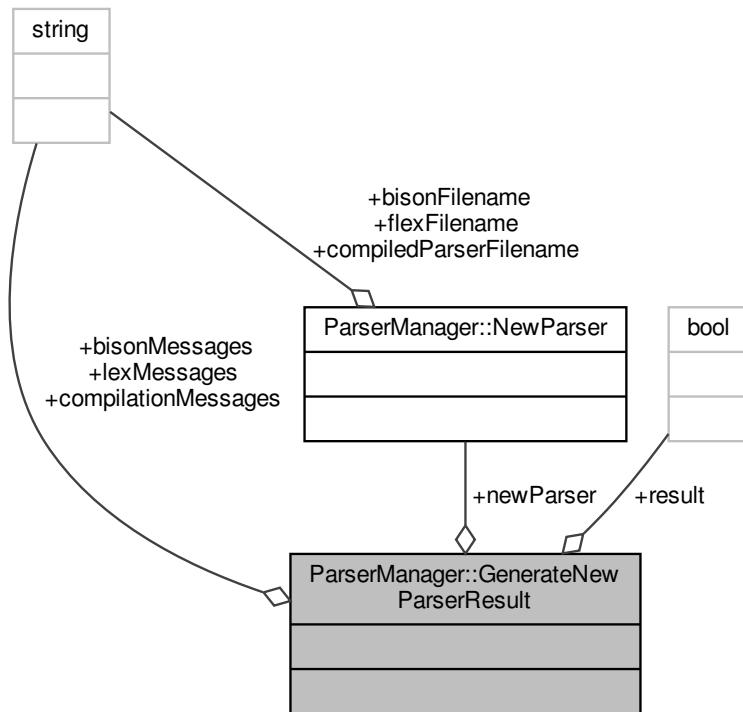
The documentation for this class was generated from the following files:

- [FullSimulationOfComplexModel.h](#)
- [FullSimulationOfComplexModel.cpp](#)

6.47 ParserManager::GenerateNewParserResult Struct Reference

```
#include <ParserManager.h>
```

Collaboration diagram for ParserManager::GenerateNewParserResult:



Public Attributes

- `bool result`
- `std::string bisonMessages`
- `std::string lexMessages`
- `std::string compilationMessages`
- `NewParser newParser`

6.47.1 Detailed Description

Definition at line 27 of file [ParserManager.h](#).

6.47.2 Member Data Documentation

6.47.2.1 bisonMessages

```
std::string ParserManager::GenerateNewParserResult::bisonMessages
```

Definition at line 29 of file [ParserManager.h](#).

6.47.2.2 compilationMessages

```
std::string ParserManager::GenerateNewParserResult::compilationMessages
```

Definition at line 31 of file [ParserManager.h](#).

6.47.2.3 lexMessages

```
std::string ParserManager::GenerateNewParserResult::lexMessages
```

Definition at line 30 of file [ParserManager.h](#).

6.47.2.4 newParser

```
NewParser ParserManager::GenerateNewParserResult::newParser
```

Definition at line 32 of file [ParserManager.h](#).

6.47.2.5 result

```
bool ParserManager::GenerateNewParserResult::result
```

Definition at line 28 of file [ParserManager.h](#).

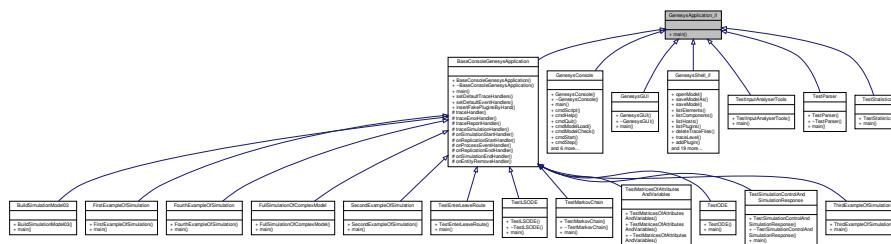
The documentation for this struct was generated from the following file:

- [ParserManager.h](#)

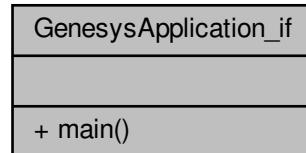
6.48 GenesysApplication_if Class Reference

```
#include <GenesysApplication_if.h>
```

Inheritance diagram for GenesysApplication_if:



Collaboration diagram for GenesysApplication_if:



Public Member Functions

- virtual int main (int argc, char **argv)=0

6.48.1 Detailed Description

Definition at line 17 of file [GenesysApplication_if.h](#).

6.48.2 Member Function Documentation

6.48.2.1 main()

```
virtual int GenesysApplication_if::main (
    int argc,
    char ** argv ) [pure virtual]
```

Implemented in [GenesysConsole](#), [BaseConsoleGenesysApplication](#), [TestMatricesOfAttributesAndVariables](#), [BuildSimulationModel03](#), [TestInputAnalyserTools](#), [TestLSODE](#), [FullSimulationOfComplexModel](#), [GenesysG↔UI](#), [TestMarkovChain](#), [TestParser](#), [TestSimulationControlAndSimulationResponse](#), [FirstExampleOfSimulation](#), [FourthExampleOfSimulation](#), [SecondExampleOfSimulation](#), [TestEnterLeaveRoute](#), [TestODE](#), [TestStatistics](#), and [ThirdExampleOfSimulation](#).

Here is the caller graph for this function:



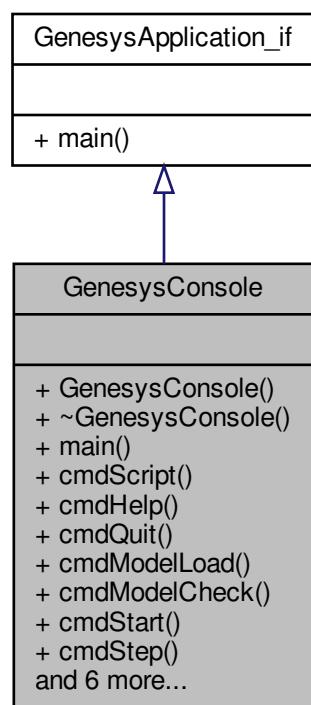
The documentation for this class was generated from the following file:

- [GenesysApplication_if.h](#)

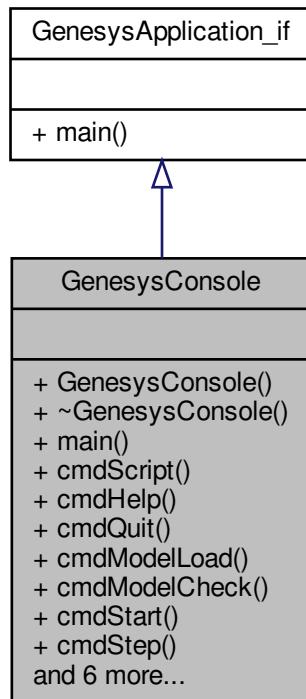
6.49 GenesysConsole Class Reference

```
#include <GenesysConsole.h>
```

Inheritance diagram for GenesysConsole:



Collaboration diagram for GenesysConsole:



Public Member Functions

- `GenesysConsole ()`
- `virtual ~GenesysConsole ()=default`
- `virtual int main (int argc, char **argv)`
- `void cmdScript ()`
- `void cmdHelp ()`
- `void cmdQuit ()`
- `void cmdModelLoad ()`
- `void cmdModelCheck ()`
- `void cmdStart ()`
- `void cmdStep ()`
- `void cmdStop ()`
- `void cmdShowReport ()`
- `void cmdModelSave ()`
- `void cmdModelShow ()`
- `void cmdVersion ()`
- `void cmdTraceLevel ()`

6.49.1 Detailed Description

Definition at line 21 of file [GenesysConsole.h](#).

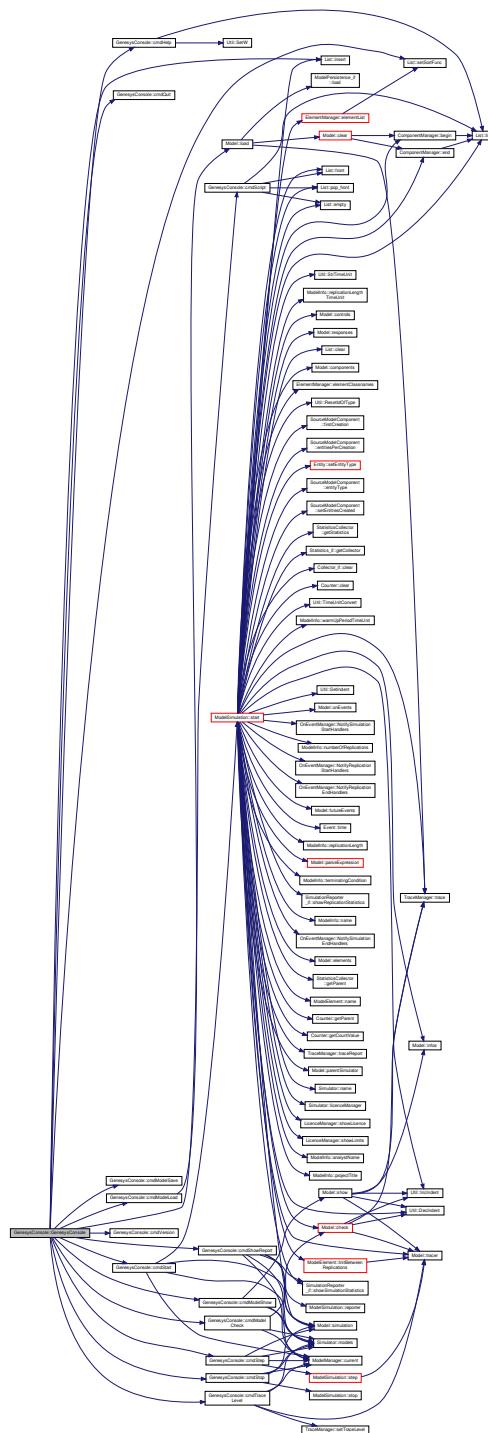
6.49.2 Constructor & Destructor Documentation

6.49.2.1 GenesysConsole()

```
GenesysConsole::GenesysConsole ( )
```

Definition at line 23 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



6.49.2.2 ~GenesysConsole()

```
virtual GenesysConsole::~GenesysConsole ( ) [virtual], [default]
```

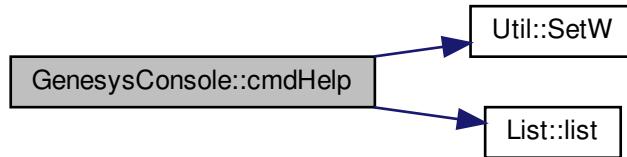
6.49.3 Member Function Documentation

6.49.3.1 cmdHelp()

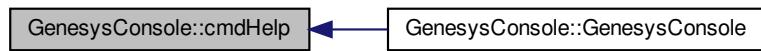
```
void GenesysConsole::cmdHelp ( )
```

Definition at line 100 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

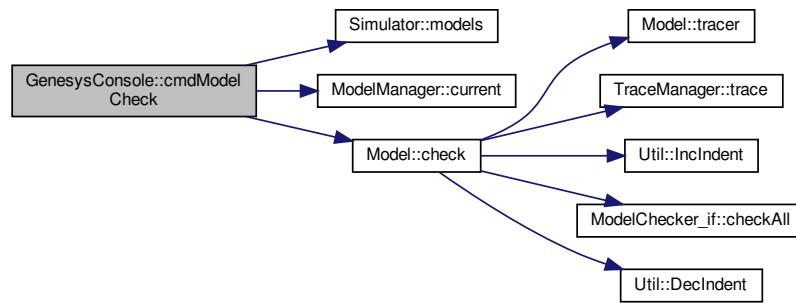


6.49.3.2 cmdModelCheck()

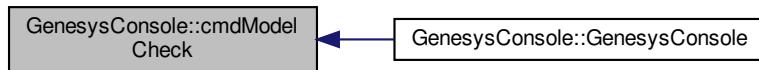
```
void GenesysConsole::cmdModelCheck( )
```

Definition at line 55 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

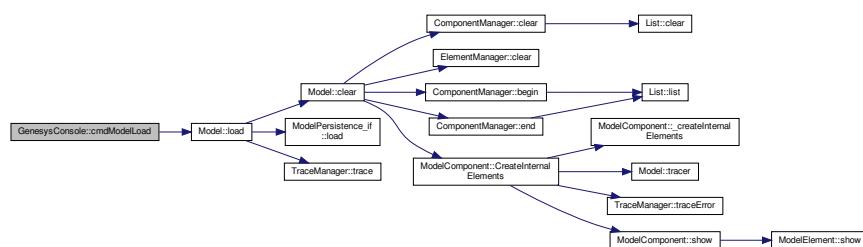


6.49.3.3 cmdModelLoad()

```
void GenesysConsole::cmdModelLoad( )
```

Definition at line 120 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.49.3.4 cmdModelSave()

```
void GenesysConsole::cmdModelSave ( )
```

Definition at line 142 of file [GenesysConsole.cpp](#).

Here is the caller graph for this function:

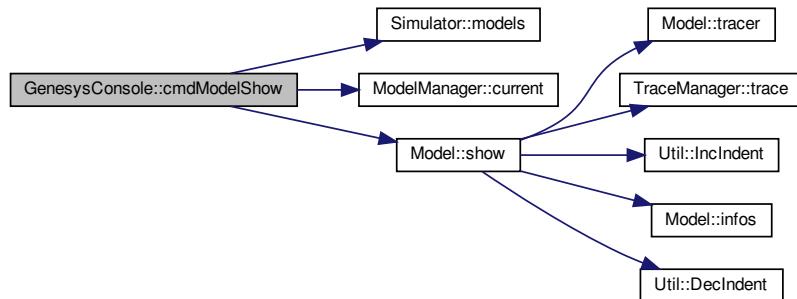


6.49.3.5 cmdModelShow()

```
void GenesysConsole::cmdModelShow ( )
```

Definition at line 132 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.49.3.6 cmdQuit()

```
void GenesysConsole::cmdQuit ( )
```

Definition at line 111 of file [GenesysConsole.cpp](#).

Here is the caller graph for this function:

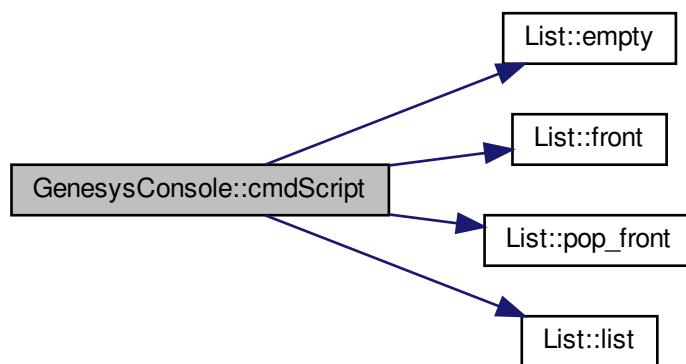


6.49.3.7 cmdScript()

```
void GenesysConsole::cmdScript ( )
```

Definition at line 146 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

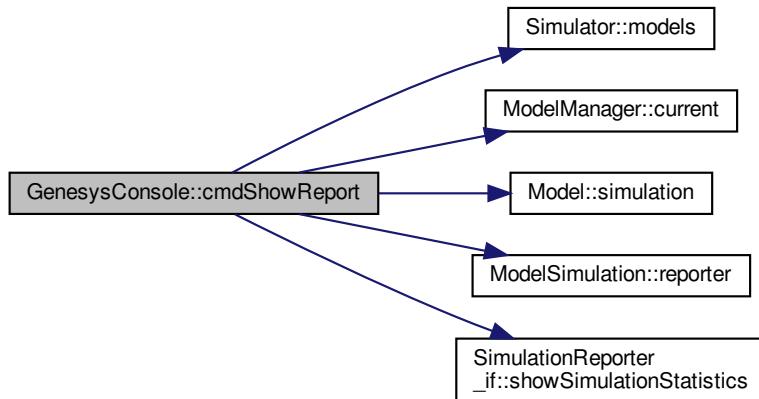


6.49.3.8 cmdShowReport()

```
void GenesysConsole::cmdShowReport ( )
```

Definition at line 91 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

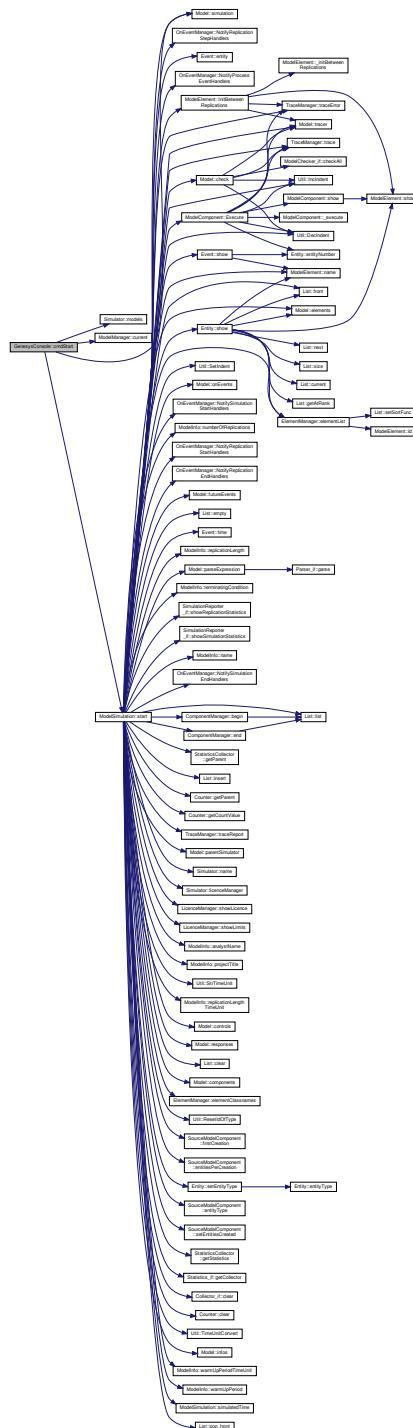


6.49.3.9 cmdStart()

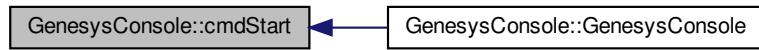
```
void GenesysConsole::cmdStart ( )
```

Definition at line 64 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

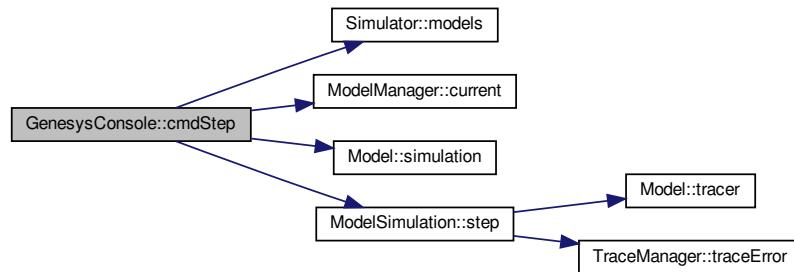


6.49.3.10 cmdStep()

```
void GenesysConsole::cmdStep( )
```

Definition at line 73 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

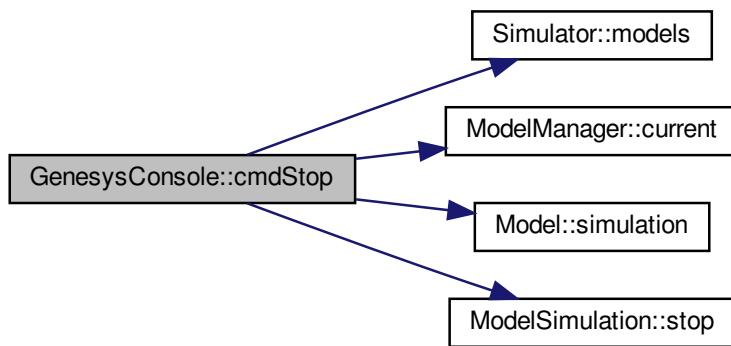


6.49.3.11 cmdStop()

```
void GenesysConsole::cmdStop( )
```

Definition at line 82 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

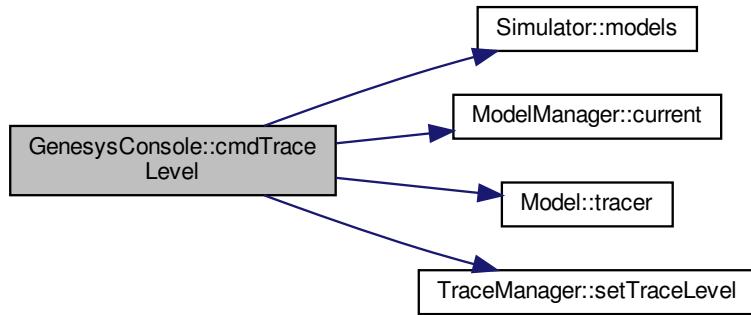


6.49.3.12 cmdTraceLevel()

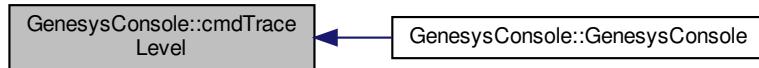
```
void GenesysConsole::cmdTraceLevel( )
```

Definition at line 44 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.49.3.13 cmdVersion()

```
void GenesysConsole::cmdVersion ( )
```

Definition at line 116 of file [GenesysConsole.cpp](#).

Here is the caller graph for this function:



6.49.3.14 main()

```
int GenesysConsole::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [GenesysApplication_if](#).

Definition at line 215 of file [GenesysConsole.cpp](#).

Here is the call graph for this function:



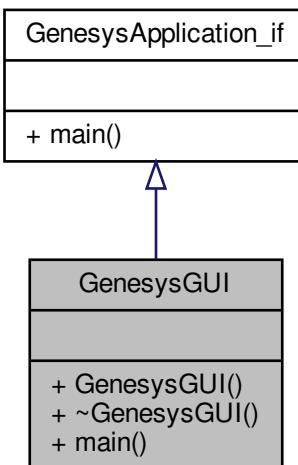
The documentation for this class was generated from the following files:

- [GenesysConsole.h](#)
- [GenesysConsole.cpp](#)

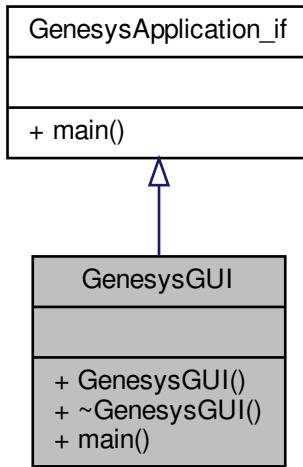
6.50 GenesysGUI Class Reference

```
#include <GenesysGUI.h>
```

Inheritance diagram for GenesysGUI:



Collaboration diagram for GenesysGUI:



Public Member Functions

- [GenesysGUI \(\)](#)
- [~GenesysGUI \(\)=default](#)
- [virtual int main \(int argc, char **argv\)](#)

6.50.1 Detailed Description

Definition at line 19 of file [GenesysGUI.h](#).

6.50.2 Constructor & Destructor Documentation

6.50.2.1 [GenesysGUI\(\)](#)

`GenesysGUI::GenesysGUI ()`

Definition at line 16 of file [GenesysGUI.cpp](#).

6.50.2.2 [~GenesysGUI\(\)](#)

`GenesysGUI::~GenesysGUI () [default]`

6.50.3 Member Function Documentation

6.50.3.1 main()

```
int GenesysGUI::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [GenesysApplication_if](#).

Definition at line 19 of file [GenesysGUI.cpp](#).

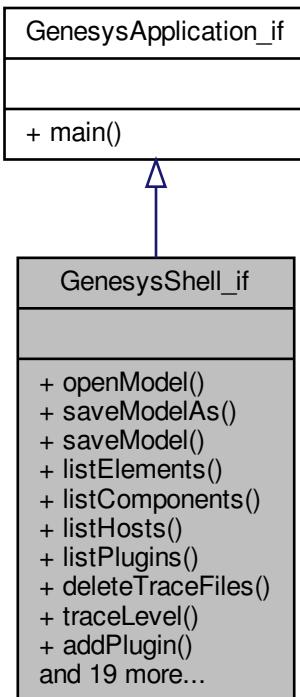
The documentation for this class was generated from the following files:

- [GenesysGUI.h](#)
- [GenesysGUI.cpp](#)

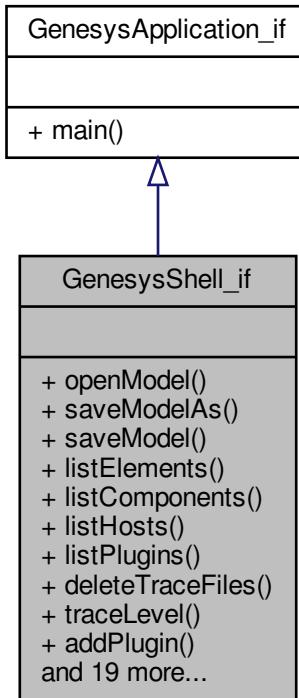
6.51 GenesysShell_if Class Reference

```
#include <GenesysShell_if.h>
```

Inheritance diagram for GenesysShell_if:



Collaboration diagram for GenesysShell_if:



Public Member Functions

- virtual void `openModel` (std::string filename)=0
- virtual void `saveModelAs` (std::string filename)=0
- virtual void `saveModel` ()=0
- virtual void `listElements` ()=0
- virtual void `listComponents` ()=0
- virtual void `listHosts` ()=0
- virtual void `listPlugins` ()=0
- virtual void `deleteTraceFiles` ()=0
- virtual void `traceLevel` (Util::TraceLevel tracelevel)=0
- virtual void `addPlugin` (std::string filename)=0
- virtual void `addFromFile` (std::string filename)=0
- virtual void `readCommandsFromFile` (std::string filename)=0
- virtual void `redirectTrace` (std::string trace, std::string dest, std::string filename)=0
- virtual void `closeModel` ()=0
- virtual void `createModel` ()=0
- virtual void `execLinuxCommand` (std::string command)=0
- virtual void `verboseMode` (bool on)=0
- virtual void `check` ()=0
- virtual void `getGenesysInfo` ()=0
- virtual void `getCommandLine` ()=0
- virtual void `sendFile` (std::string filename, std::string hostname, std::string portname)=0

- virtual void `setActivationCode` (std::string code)=0
- virtual void `receiveFile` (std::string filename)=0
- virtual void `startSimulation` ()=0
- virtual void `stepSimulation` ()=0
- virtual void `stopSimulation` ()=0
- virtual void `showInit` ()=0
- virtual void `showHelp` ()=0
- virtual void `showHostName` ()=0

6.51.1 Detailed Description

Definition at line 20 of file [GenesysShell_if.h](#).

6.51.2 Member Function Documentation

6.51.2.1 addFromFile()

```
virtual void GenesysShell_if::addFromFile (
    std::string filename ) [pure virtual]
```

6.51.2.2 addPlugin()

```
virtual void GenesysShell_if::addPlugin (
    std::string filename ) [pure virtual]
```

6.51.2.3 check()

```
virtual void GenesysShell_if::check ( ) [pure virtual]
```

6.51.2.4 closeModel()

```
virtual void GenesysShell_if::closeModel ( ) [pure virtual]
```

6.51.2.5 createModel()

```
virtual void GenesysShell_if::createModel ( ) [pure virtual]
```

6.51.2.6 deleteTraceFiles()

```
virtual void GenesysShell_if::deleteTraceFiles () [pure virtual]
```

6.51.2.7 execLinuxCommand()

```
virtual void GenesysShell_if::execLinuxCommand (
    std::string command ) [pure virtual]
```

6.51.2.8 getCommandLine()

```
virtual void GenesysShell_if::getCommandLine () [pure virtual]
```

6.51.2.9 getGenesysInfo()

```
virtual void GenesysShell_if::getGenesysInfo () [pure virtual]
```

6.51.2.10 listComponents()

```
virtual void GenesysShell_if::listComponents () [pure virtual]
```

6.51.2.11 listElements()

```
virtual void GenesysShell_if::listElements () [pure virtual]
```

6.51.2.12 listHosts()

```
virtual void GenesysShell_if::listHosts () [pure virtual]
```

6.51.2.13 listPlugins()

```
virtual void GenesysShell_if::listPlugins () [pure virtual]
```

6.51.2.14 openModel()

```
virtual void GenesysShell_if::openModel (
    std::string filename ) [pure virtual]
```

6.51.2.15 readCommandsFromFile()

```
virtual void GenesysShell_if::readCommandsFromFile (
    std::string filename ) [pure virtual]
```

6.51.2.16 receiveFile()

```
virtual void GenesysShell_if::receiveFile (
    std::string filename ) [pure virtual]
```

6.51.2.17 redirectTrace()

```
virtual void GenesysShell_if::redirectTrace (
    std::string trace,
    std::string dest,
    std::string filename ) [pure virtual]
```

6.51.2.18 saveModel()

```
virtual void GenesysShell_if::saveModel ( ) [pure virtual]
```

6.51.2.19 saveModelAs()

```
virtual void GenesysShell_if::saveModelAs (
    std::string filename ) [pure virtual]
```

6.51.2.20 sendFile()

```
virtual void GenesysShell_if::sendFile (
    std::string filename,
    std::string hostname,
    std::string portname ) [pure virtual]
```

6.51.2.21 setActivationCode()

```
virtual void GenesysShell_if::setActivationCode (
    std::string code ) [pure virtual]
```

6.51.2.22 showHelp()

```
virtual void GenesysShell_if::showHelp ( ) [pure virtual]
```

6.51.2.23 showHostName()

```
virtual void GenesysShell_if::showHostName ( ) [pure virtual]
```

6.51.2.24 showInit()

```
virtual void GenesysShell_if::showInit ( ) [pure virtual]
```

6.51.2.25 startSimulation()

```
virtual void GenesysShell_if::startSimulation ( ) [pure virtual]
```

6.51.2.26 stepSimulation()

```
virtual void GenesysShell_if::stepSimulation ( ) [pure virtual]
```

6.51.2.27 stopSimulation()

```
virtual void GenesysShell_if::stopSimulation ( ) [pure virtual]
```

6.51.2.28 traceLevel()

```
virtual void GenesysShell_if::traceLevel (
    Util::TraceLevel tracelevel ) [pure virtual]
```

6.51.2.29 verboseMode()

```
virtual void GenesysShell_if::verboseMode (  
    bool on ) [pure virtual]
```

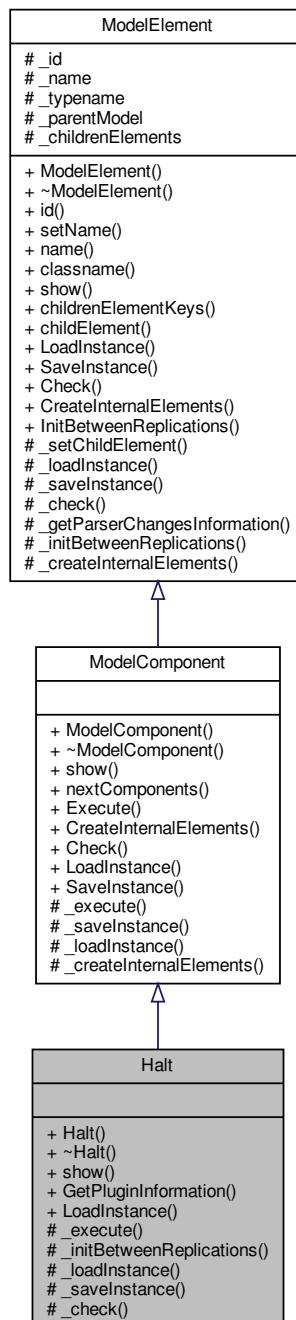
The documentation for this class was generated from the following file:

- [GenesysShell_if.h](#)

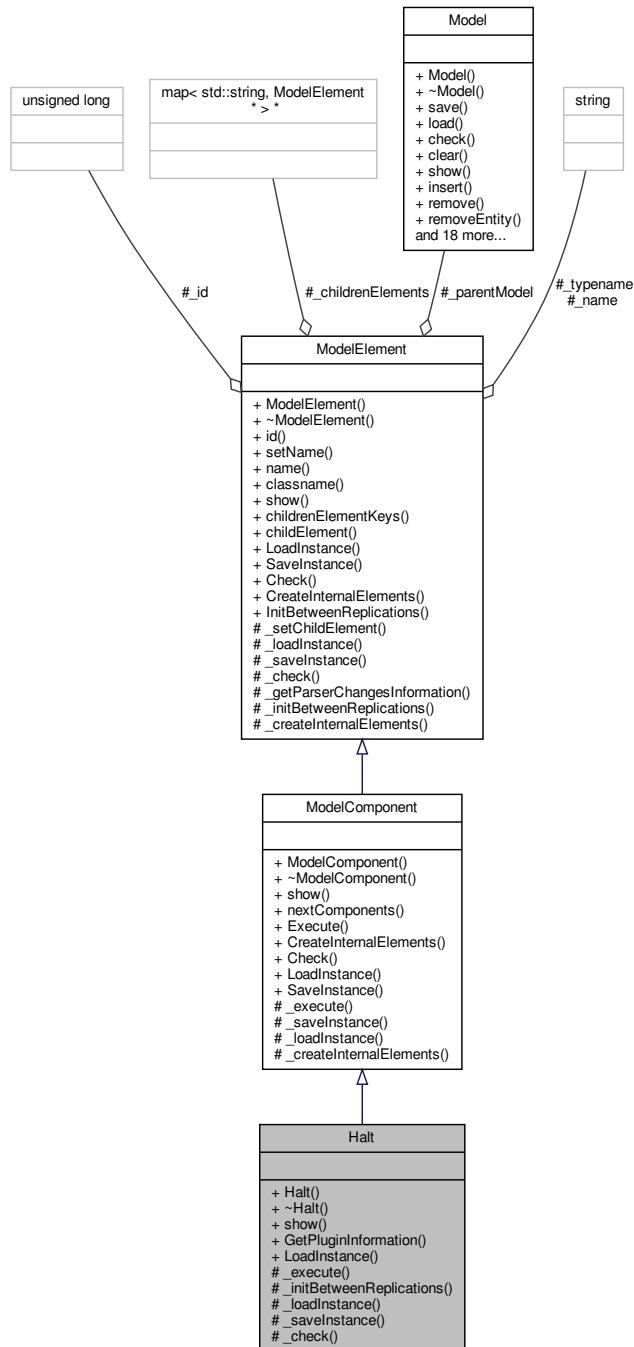
6.52 Halt Class Reference

```
#include <Halt.h>
```

Inheritance diagram for Halt:



Collaboration diagram for Halt:



Public Member Functions

- [Halt \(Model *model, std::string name=""\)](#)
- virtual [~Halt \(\)=default](#)
- virtual std::string [show \(\)](#)

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.52.1 Detailed Description

Halt module DESCRIPTION The `Halt` module changes the status of a transporter unit to inactive. If the transporter is busy at the time when an entity enters the `Halt` module, the status is considered busy and inactive until the entity that controls the transporter frees the unit. If the transporter is idle at the time when an entity halts the transporter, it is set to inactive immediately. Once a transporter unit has been halted, no entities will get control of the transporter until it is activated. TYPICAL USES Stop a forklift for scheduled maintenance Disable a broken gurney in an emergency room PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Transporter Name Name of the transporter to halt. Unit Number Determines which of the transporter units in the transporter set to halt.

Definition at line 39 of file `Halt.h`.

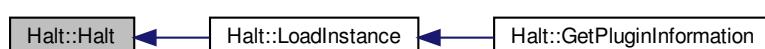
6.52.2 Constructor & Destructor Documentation

6.52.2.1 Halt()

```
Halt::Halt (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file `Halt.cpp`.

Here is the caller graph for this function:



6.52.2.2 ~Halt()

```
virtual Halt::~Halt ( ) [virtual], [default]
```

6.52.3 Member Function Documentation

6.52.3.1 _check()

```
bool Halt::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Halt.cpp](#).

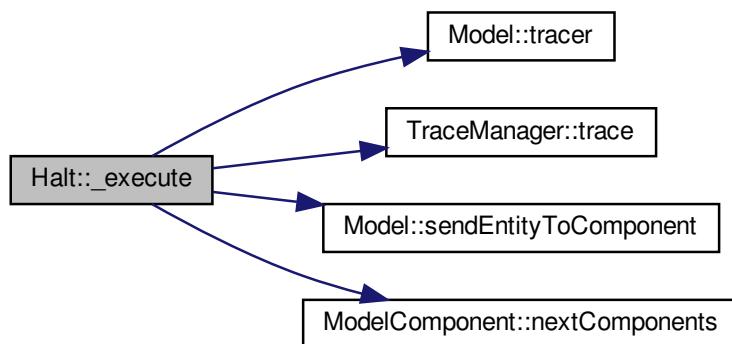
6.52.3.2 _execute()

```
void Halt::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Halt.cpp](#).

Here is the call graph for this function:



6.52.3.3 `_initBetweenReplications()`

```
void Halt::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [48](#) of file [Halt.cpp](#).

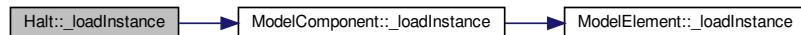
6.52.3.4 `_loadInstance()`

```
bool Halt::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [40](#) of file [Halt.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



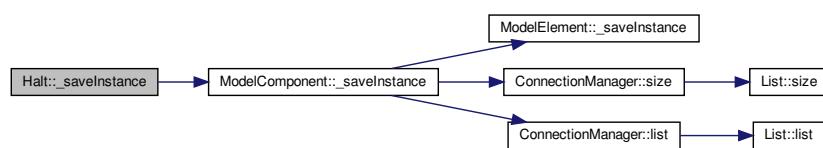
6.52.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Halt::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [51](#) of file [Halt.cpp](#).

Here is the call graph for this function:

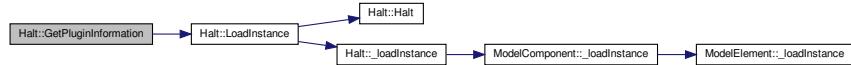


6.52.3.6 GetPluginInformation()

```
PluginInformation * Halt::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [Halt.cpp](#).

Here is the call graph for this function:

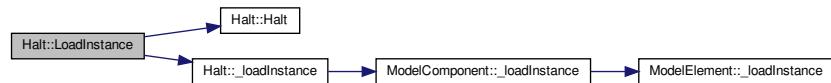


6.52.3.7 LoadInstance()

```
ModelComponent * Halt::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Halt.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.52.3.8 show()

```
std::string Halt::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [21](#) of file [Halt.cpp](#).

Here is the call graph for this function:



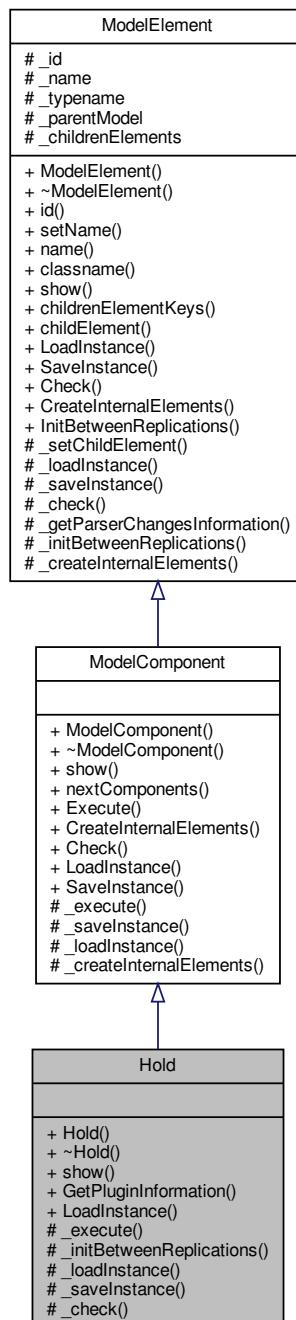
The documentation for this class was generated from the following files:

- [Halt.h](#)
- [Halt.cpp](#)

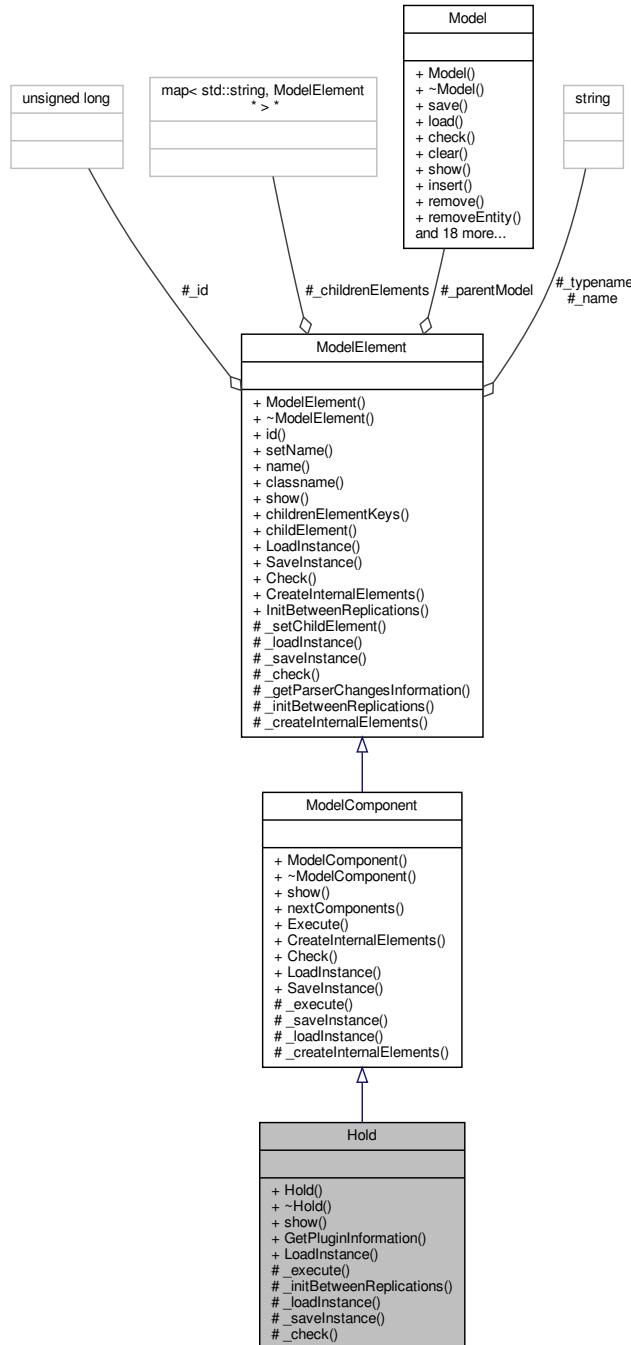
6.53 Hold Class Reference

```
#include <Hold.h>
```

Inheritance diagram for Hold:



Collaboration diagram for Hold:



Public Member Functions

- `Hold (Model *model, std::string name="")`
- virtual `~Hold ()=default`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.53.1 Detailed Description

Hold module DESCRIPTION This module will hold an entity in a queue to wait for a signal, wait for a specified condition to become true (scan), or be held infinitely (to be removed later with the **Remove** module). If the entity is holding for a signal, the **Signal** module is used elsewhere in the model to allow the entity to move on to the next module. If the entity is holding for a given condition to be true, the entity will remain at the module (either in a defined or internal queue) until the condition(s) becomes true. When the entity is in an infinite hold, the **Remove** module is used elsewhere in the model to allow the entity to continue processing. TYPICAL USES Waiting for a traffic light to turn green Holding a part for authorization Checking the status of a machine or operator to continue a process PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Indicates the reasoning for holding the entity within a specified or internal queue. Wait for **Signal** will hold the entity until a signal of the same value is received. Scan for Condition will hold the entity until the specified condition becomes true. Infinite **Hold** will hold the entity until it is removed from the queue by a **Remove** module. Wait for Value **Signal** code for the waiting entity. Applies only when Type is Wait for **Signal**. Limit Maximum number of waiting entities that will be released upon receipt of a signal. Applies only when Type is Wait for **Signal**. Condition Specifies the condition that will be evaluated to hold the entity at the module. If the condition is evaluated to true, the entity leaves the module immediately. If the condition is false, the entity will wait in the associated queue until the condition becomes true. Applies only when Type is Scan for Condition. Queue Type Determines the type of queue used to hold the entities. If **Queue** is selected, the queue name is specified. If **Set** is selected, the queue set and member in the set are specified. If Internal is selected, an internal queue is used to hold all waiting entities. Attribute and Expression are additional methods for defining the queue to be used. Queue Name This field is visible only if Queue Type is **Queue**, and it defines the symbol name of the queue. Set Name This field is visible only if Queue Type is **Set**, and it defines the queue set that contains the queue being referenced. Set Index This field is visible only if Queue Type is **Set**, and it defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entry for a queue set containing three members is an expression that evaluates to 1, 2, or 3. Attribute This field is visible only if Queue Type is **Attribute**. The attribute entered in this field will be evaluated to indicate which queue is to be used. Expression This field is visible only if Queue Type is Expression. The expression entered in this field will be evaluated to indicate which queue is to be used.

Definition at line 75 of file **Hold.h**.

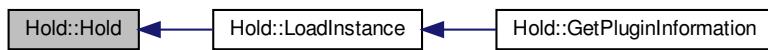
6.53.2 Constructor & Destructor Documentation

6.53.2.1 Hold()

```
Hold::Hold (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Hold.cpp](#).

Here is the caller graph for this function:



6.53.2.2 ~Hold()

```
virtual Hold::~Hold ( ) [virtual], [default]
```

6.53.3 Member Function Documentation

6.53.3.1 _check()

```
bool Hold::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Hold.cpp](#).

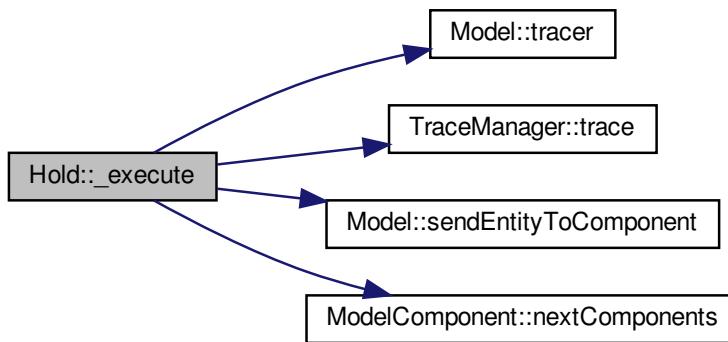
6.53.3.2 `_execute()`

```
void Hold::_execute (
    Entity * entity) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Hold.cpp](#).

Here is the call graph for this function:



6.53.3.3 `_initBetweenReplications()`

```
void Hold::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Hold.cpp](#).

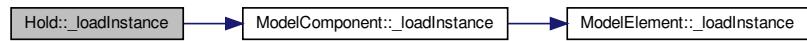
6.53.3.4 `_loadInstance()`

```
bool Hold::_loadInstance (
    std::map< std::string, std::string > * fields) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 40 of file [Hold.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



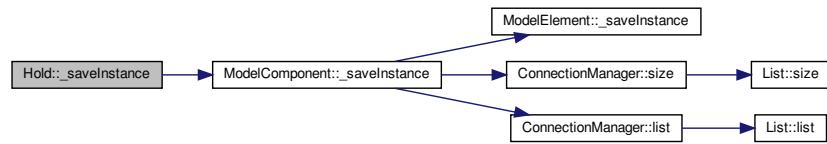
6.53.3.5 `_saveInstance()`

```
std::map< std::string, std::string > * Hold::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [51](#) of file [Hold.cpp](#).

Here is the call graph for this function:



6.53.3.6 `GetPluginInformation()`

```
PluginInformation * Hold::GetPluginInformation ( ) [static]
```

Definition at line [63](#) of file [Hold.cpp](#).

Here is the call graph for this function:

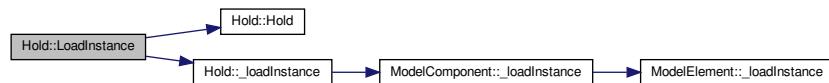


6.53.3.7 LoadInstance()

```
ModelComponent * Hold::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 25 of file [Hold.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



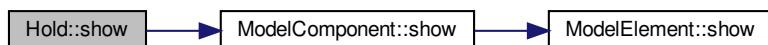
6.53.3.8 show()

```
std::string Hold::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Hold.cpp](#).

Here is the call graph for this function:



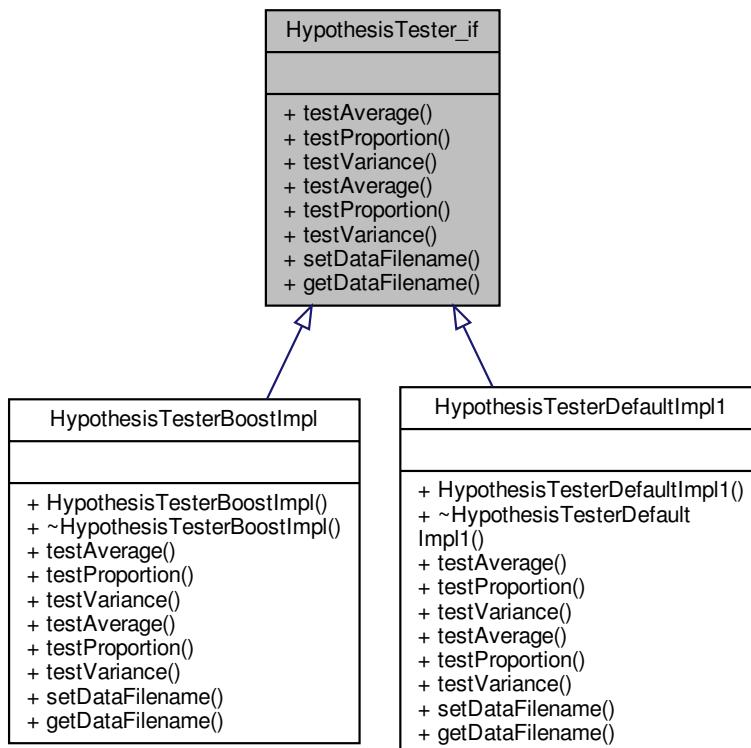
The documentation for this class was generated from the following files:

- [Hold.h](#)
- [Hold.cpp](#)

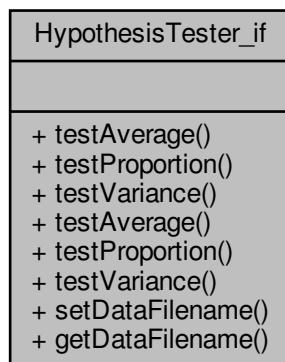
6.54 HypothesisTester_if Class Reference

```
#include <HypothesisTester_if.h>
```

Inheritance diagram for HypothesisTester_if:



Collaboration diagram for HypothesisTester_if:



Public Types

- enum `H1Comparition` { `DIFFERENT` = 1, `LESS_THAN` = 2, `GREATER_THAN` = 3 }

Public Member Functions

- virtual double `testAverage` (double confidencelevel, double avg, `H1Comparition` comp)=0
- virtual double `testProportion` (double confidencelevel, double prop, `H1Comparition` comp)=0
- virtual double `testVariance` (double confidencelevel, double var, `H1Comparition` comp)=0
- virtual double `testAverage` (double confidencelevel, std::string secondPopulationDataFilename, `H1Comparition` comp)=0
- virtual double `testProportion` (double confidencelevel, std::string secondPopulationDataFilename, `H1Comparition` comp)=0
- virtual double `testVariance` (double confidencelevel, std::string secondPopulationDataFilename, `H1Comparition` comp)=0
- virtual void `setDataFilename` (std::string datafilename)=0
- virtual std::string `getDataFilename` ()=0

6.54.1 Detailed Description

Interface for parametric hypothesis tests based on a datafile.

Definition at line 22 of file [HypothesisTester_if.h](#).

6.54.2 Member Enumeration Documentation**6.54.2.1 H1Comparition**

```
enum HypothesisTester_if::H1Comparition
```

Enumerator

<code>DIFFERENT</code>	
<code>LESS_THAN</code>	
<code>GREATER_THAN</code>	

Definition at line 25 of file [HypothesisTester_if.h](#).

6.54.3 Member Function Documentation**6.54.3.1 getDataFilename()**

```
virtual std::string HypothesisTester_if::getDataFilename ( ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

6.54.3.2 setDataFilename()

```
virtual void HypothesisTester_if::setDataFilename (
    std::string datafilename ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

Here is the caller graph for this function:

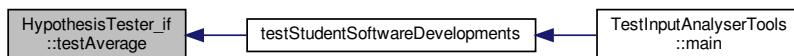


6.54.3.3 testAverage() [1/2]

```
virtual double HypothesisTester_if::testAverage (
    double confidencelevel,
    double avg,
    H1Comparition comp ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

Here is the caller graph for this function:



6.54.3.4 testAverage() [2/2]

```
virtual double HypothesisTester_if::testAverage (
    double confidencelevel,
    std::string secondPopulationDatafilename,
    H1Comparition comp ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

6.54.3.5 testProportion() [1/2]

```
virtual double HypothesisTester_if::testProportion (
    double confidencelevel,
    double prop,
    H1Comparition comp ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

6.54.3.6 testProportion() [2/2]

```
virtual double HypothesisTester_if::testProportion (
    double confidencelevel,
    std::string secondPopulationDataFilename,
    H1Comparition comp ) [pure virtual]
```

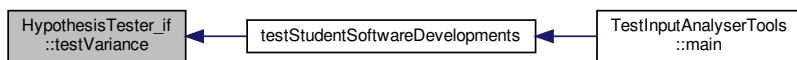
Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

6.54.3.7 testVariance() [1/2]

```
virtual double HypothesisTester_if::testVariance (
    double confidencelevel,
    double var,
    H1Comparition comp ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

Here is the caller graph for this function:



6.54.3.8 testVariance() [2/2]

```
virtual double HypothesisTester_if::testVariance (
    double confidencelevel,
    std::string secondPopulationDataFilename,
    H1Comparition comp ) [pure virtual]
```

Implemented in [HypothesisTesterBoostImpl](#), and [HypothesisTesterDefaultImpl1](#).

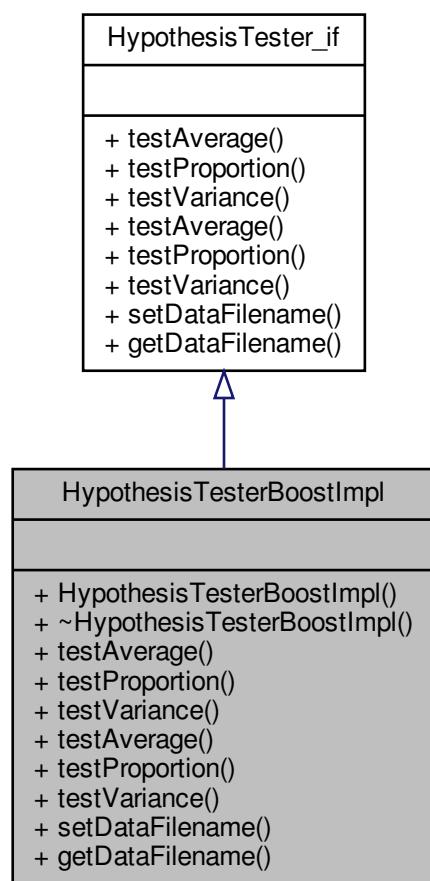
The documentation for this class was generated from the following file:

- [HypothesisTester_if.h](#)

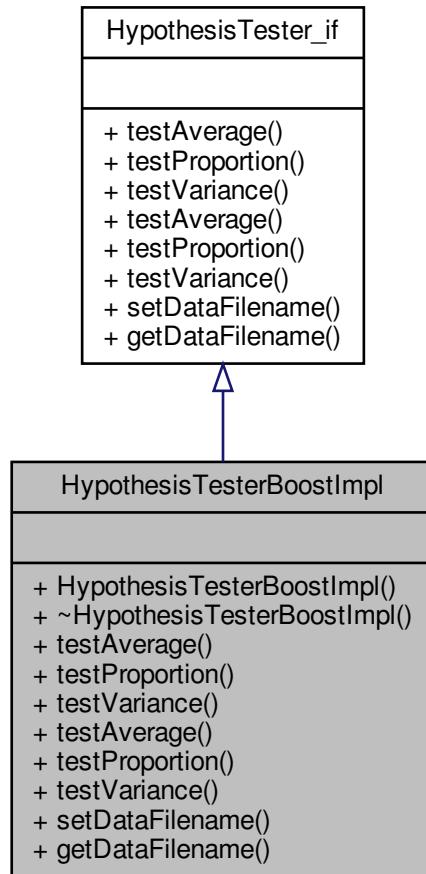
6.55 HypothesisTesterBoostImpl Class Reference

```
#include <HypothesisTesterBoostImpl.h>
```

Inheritance diagram for HypothesisTesterBoostImpl:



Collaboration diagram for HypothesisTesterBoostImpl:



Public Member Functions

- `HypothesisTesterBoostImpl ()`
- virtual `~HypothesisTesterBoostImpl ()=default`
- virtual double `testAverage` (double confidencelevel, double avg, `H1Comparition` comp)
- virtual double `testProportion` (double confidencelevel, double prop, `H1Comparition` comp)
- virtual double `testVariance` (double confidencelevel, double var, `H1Comparition` comp)
- virtual double `testAverage` (double confidencelevel, std::string secondPopulationDataFilename, `H1Comparition` comp)
- virtual double `testProportion` (double confidencelevel, std::string secondPopulationDataFilename, `H1Comparition` comp)
- virtual double `testVariance` (double confidencelevel, std::string secondPopulationDataFilename, `H1Comparition` comp)
- virtual void `setDataFilename` (std::string dataFilename)
- virtual std::string `getDataFilename ()`

Additional Inherited Members

6.55.1 Detailed Description

Definition at line 20 of file [HypothesisTesterBoostImpl.h](#).

6.55.2 Constructor & Destructor Documentation

6.55.2.1 [HypothesisTesterBoostImpl\(\)](#)

```
HypothesisTesterBoostImpl::HypothesisTesterBoostImpl ()
```

Definition at line 16 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.2.2 [~HypothesisTesterBoostImpl\(\)](#)

```
virtual HypothesisTesterBoostImpl::~HypothesisTesterBoostImpl () [virtual], [default]
```

6.55.3 Member Function Documentation

6.55.3.1 [getDataFilename\(\)](#)

```
std::string HypothesisTesterBoostImpl::getDataFilename () [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 45 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.2 [setDataFilename\(\)](#)

```
void HypothesisTesterBoostImpl::setDataFilename (
    std::string datafilename ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 42 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.3 testAverage() [1/2]

```
double HypothesisTesterBoostImpl::testAverage (
    double confidencelevel,
    double avg,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 18 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.4 testAverage() [2/2]

```
double HypothesisTesterBoostImpl::testAverage (
    double confidencelevel,
    std::string secondPopulationDataFilename,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 30 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.5 testProportion() [1/2]

```
double HypothesisTesterBoostImpl::testProportion (
    double confidencelevel,
    double prop,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 22 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.6 testProportion() [2/2]

```
double HypothesisTesterBoostImpl::testProportion (
    double confidencelevel,
    std::string secondPopulationDataFilename,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 34 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.7 `testVariance()` [1/2]

```
double HypothesisTesterBoostImpl::testVariance (
    double confidencelevel,
    double var,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 26 of file [HypothesisTesterBoostImpl.cpp](#).

6.55.3.8 `testVariance()` [2/2]

```
double HypothesisTesterBoostImpl::testVariance (
    double confidencelevel,
    std::string secondPopulationDatafilename,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 38 of file [HypothesisTesterBoostImpl.cpp](#).

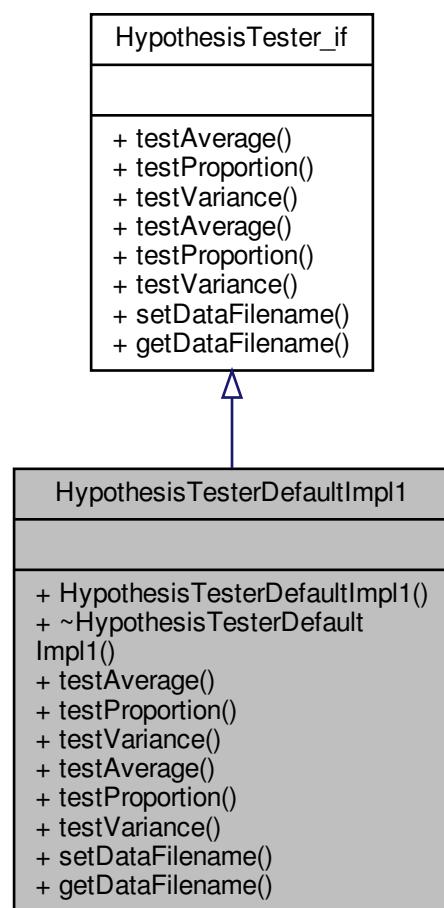
The documentation for this class was generated from the following files:

- [HypothesisTesterBoostImpl.h](#)
- [HypothesisTesterBoostImpl.cpp](#)

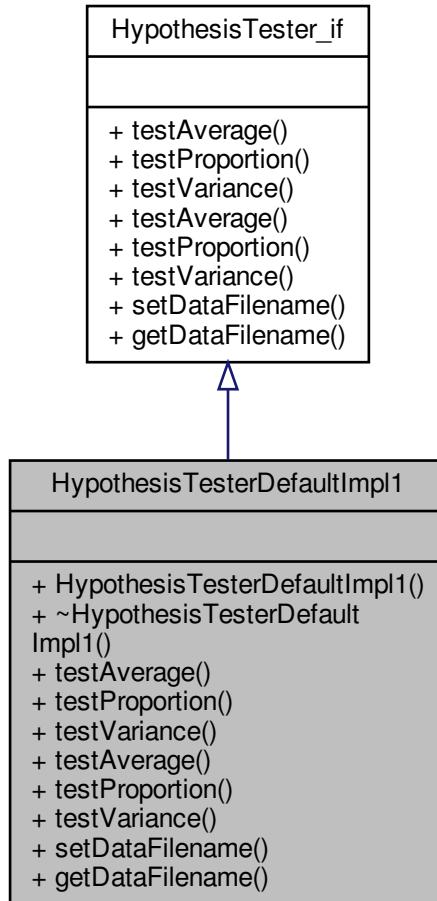
6.56 HypothesisTesterDefaultImpl1 Class Reference

```
#include <HypothesisTesterDefaultImpl1.h>
```

Inheritance diagram for HypothesisTesterDefaultImpl1:



Collaboration diagram for HypothesisTesterDefaultImpl1:



Public Member Functions

- [HypothesisTesterDefaultImpl1 \(\)](#)
- virtual [~HypothesisTesterDefaultImpl1 \(\)=default](#)
- virtual double [testAverage \(double confidencelevel, double avg, H1Comparition comp\)](#)
- virtual double [testProportion \(double confidencelevel, double prop, H1Comparition comp\)](#)
- virtual double [testVariance \(double confidencelevel, double var, H1Comparition comp\)](#)
- virtual double [testAverage \(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp\)](#)
- virtual double [testProportion \(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp\)](#)
- virtual double [testVariance \(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp\)](#)
- virtual void [setDataFilename \(std::string dataFilename\)](#)
- virtual std::string [getDataFilename \(\)](#)

Additional Inherited Members

6.56.1 Detailed Description

Definition at line 20 of file [HypothesisTesterDefaultImpl1.h](#).

6.56.2 Constructor & Destructor Documentation

6.56.2.1 HypothesisTesterDefaultImpl1()

```
HypothesisTesterDefaultImpl1::HypothesisTesterDefaultImpl1 ()
```

Definition at line 17 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.2.2 ~HypothesisTesterDefaultImpl1()

```
virtual HypothesisTesterDefaultImpl1::~HypothesisTesterDefaultImpl1 () [virtual], [default]
```

6.56.3 Member Function Documentation

6.56.3.1 getDataFilename()

```
std::string HypothesisTesterDefaultImpl1::getDataFilename () [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 49 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.2 setDataFilename()

```
void HypothesisTesterDefaultImpl1::setDataFilename (
    std::string datafilename ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 46 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.3 testAverage() [1/2]

```
double HypothesisTesterDefaultImpl1::testAverage (
    double confidencelevel,
    double avg,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 22 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.4 testAverage() [2/2]

```
double HypothesisTesterDefaultImpl1::testAverage (
    double confidencelevel,
    std::string secondPopulationDataFilename,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 34 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.5 testProportion() [1/2]

```
double HypothesisTesterDefaultImpl1::testProportion (
    double confidencelevel,
    double prop,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 26 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.6 testProportion() [2/2]

```
double HypothesisTesterDefaultImpl1::testProportion (
    double confidencelevel,
    std::string secondPopulationDataFilename,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 38 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.7 testVariance() [1/2]

```
double HypothesisTesterDefaultImpl1::testVariance (
    double confidencelevel,
    double var,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 30 of file [HypothesisTesterDefaultImpl1.cpp](#).

6.56.3.8 testVariance() [2/2]

```
double HypothesisTesterDefaultImpl1::testVariance (
    double confidencelevel,
    std::string secondPopulationDatafilename,
    H1Comparition comp ) [virtual]
```

Implements [HypothesisTester_if](#).

Definition at line 42 of file [HypothesisTesterDefaultImpl1.cpp](#).

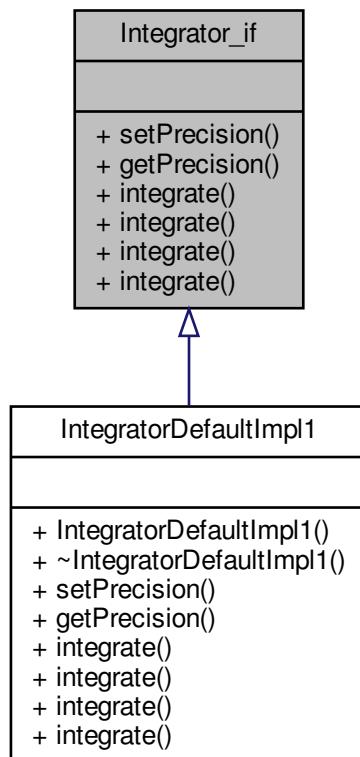
The documentation for this class was generated from the following files:

- [HypothesisTesterDefaultImpl1.h](#)
- [HypothesisTesterDefaultImpl1.cpp](#)

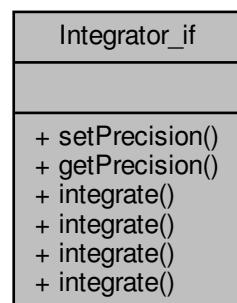
6.57 Integrator_if Class Reference

```
#include <Integrator_if.h>
```

Inheritance diagram for Integrator_if:



Collaboration diagram for Integrator_if:



Public Member Functions

- virtual void `setPrecision` (double e)=0

- virtual double [getPrecision \(\)=0](#)
- virtual double [integrate \(double min, double max, double\(*f\)\(double, double\), double p2\)=0](#)
- virtual double [integrate \(double min, double max, double\(*f\)\(double, double, double\), double p2, double p3\)=0](#)
- virtual double [integrate \(double min, double max, double\(*f\)\(double, double, double, double\), double p2, double p3, double p4\)=0](#)
- virtual double [integrate \(double min, double max, double\(*f\)\(double, double, double, double, double\), double p2, double p3, double p4, double p5\)=0](#)

6.57.1 Detailed Description

Interface used by classes that perform the numerical integration of functions with one to four parameters. It is mainly used for calculating the probability of theoretical distributions, from its probability distribution functions.

Definition at line 21 of file [Integrator_if.h](#).

6.57.2 Member Function Documentation

6.57.2.1 [getPrecision\(\)](#)

```
virtual double Integrator_if::getPrecision ( ) [pure virtual]
```

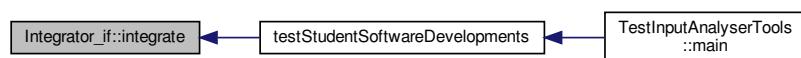
Implemented in [IntegratorDefaultImpl1](#).

6.57.2.2 [integrate\(\)](#) [1/4]

```
virtual double Integrator_if::integrate (
    double min,
    double max,
    double(*) (double, double) f,
    double p2 ) [pure virtual]
```

Implemented in [IntegratorDefaultImpl1](#).

Here is the caller graph for this function:



6.57.2.3 `integrate()` [2/4]

```
virtual double Integrator_if::integrate (
    double min,
    double max,
    double(*)(double, double, double) f,
    double p2,
    double p3 ) [pure virtual]
```

Implemented in [IntegratorDefaultImpl1](#).

6.57.2.4 `integrate()` [3/4]

```
virtual double Integrator_if::integrate (
    double min,
    double max,
    double(*)(double, double, double, double) f,
    double p2,
    double p3,
    double p4 ) [pure virtual]
```

Implemented in [IntegratorDefaultImpl1](#).

6.57.2.5 `integrate()` [4/4]

```
virtual double Integrator_if::integrate (
    double min,
    double max,
    double(*)(double, double, double, double, double) f,
    double p2,
    double p3,
    double p4,
    double p5 ) [pure virtual]
```

Implemented in [IntegratorDefaultImpl1](#).

6.57.2.6 `setPrecision()`

```
virtual void Integrator_if::setPrecision (
    double e ) [pure virtual]
```

Implemented in [IntegratorDefaultImpl1](#).

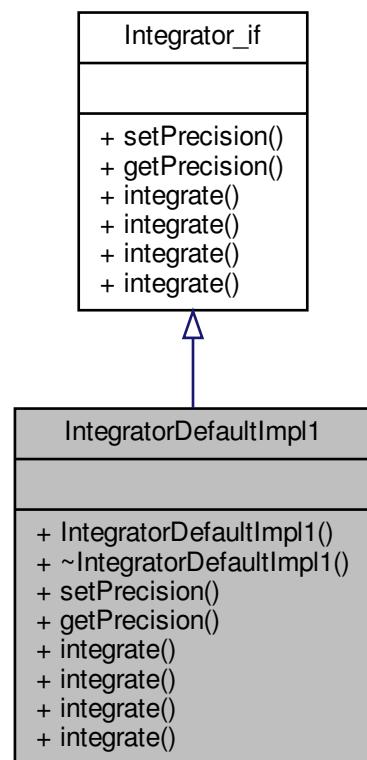
The documentation for this class was generated from the following file:

- [Integrator_if.h](#)

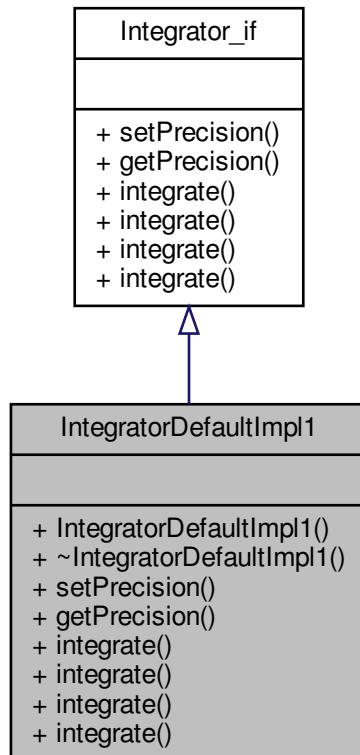
6.58 IntegratorDefaultImpl1 Class Reference

```
#include <IntegratorDefaultImpl1.h>
```

Inheritance diagram for IntegratorDefaultImpl1:



Collaboration diagram for IntegratorDefaultImpl1:



Public Member Functions

- `IntegratorDefaultImpl1 ()`
`https://codereview.stackexchange.com/questions/200289/implementing-numerical-integration`
- `virtual ~IntegratorDefaultImpl1 ()=default`
- `virtual void setPrecision (double e)`
- `virtual double getPrecision ()`
- `virtual double integrate (double min, double max, double(*f)(double, double), double p2)`
- `virtual double integrate (double min, double max, double(*f)(double, double, double), double p2, double p3)`
- `virtual double integrate (double min, double max, double(*f)(double, double, double, double), double p2, double p3, double p4)`
- `virtual double integrate (double min, double max, double(*f)(double, double, double, double, double), double p2, double p3, double p4, double p5)`

6.58.1 Detailed Description

Definition at line 19 of file [IntegratorDefaultImpl1.h](#).

6.58.2 Constructor & Destructor Documentation

6.58.2.1 IntegratorDefaultImpl1()

```
IntegratorDefaultImpl1::IntegratorDefaultImpl1 ( )
```

<https://codereview.stackexchange.com/questions/200289/implementing-numerical-integration>

Definition at line 19 of file [IntegratorDefaultImpl1.cpp](#).

6.58.2.2 ~IntegratorDefaultImpl1()

```
virtual IntegratorDefaultImpl1::~IntegratorDefaultImpl1 ( ) [virtual], [default]
```

6.58.3 Member Function Documentation

6.58.3.1 getPrecision()

```
double IntegratorDefaultImpl1::getPrecision ( ) [virtual]
```

Implements [Integrator_if](#).

Definition at line 28 of file [IntegratorDefaultImpl1.cpp](#).

6.58.3.2 integrate() [1/4]

```
double IntegratorDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double) f,
    double p2 ) [virtual]
```

Implements [Integrator_if](#).

Definition at line 32 of file [IntegratorDefaultImpl1.cpp](#).

6.58.3.3 integrate() [2/4]

```
double IntegratorDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double, double) f,
    double p2,
    double p3 ) [virtual]
```

Implements [Integrator_if](#).

Definition at line 50 of file [IntegratorDefaultImpl1.cpp](#).

6.58.3.4 [integrate\(\)](#) [3/4]

```
double IntegratorDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double, double, double) f,
    double p2,
    double p3,
    double p4 ) [virtual]
```

Implements [Integrator_if](#).

Definition at line 67 of file [IntegratorDefaultImpl1.cpp](#).

6.58.3.5 [integrate\(\)](#) [4/4]

```
double IntegratorDefaultImpl1::integrate (
    double min,
    double max,
    double(*)(double, double, double, double, double) f,
    double p2,
    double p3,
    double p4,
    double p5 ) [virtual]
```

Implements [Integrator_if](#).

Definition at line 84 of file [IntegratorDefaultImpl1.cpp](#).

6.58.3.6 [setPrecision\(\)](#)

```
void IntegratorDefaultImpl1::setPrecision (
    double e ) [virtual]
```

Implements [Integrator_if](#).

Definition at line 24 of file [IntegratorDefaultImpl1.cpp](#).

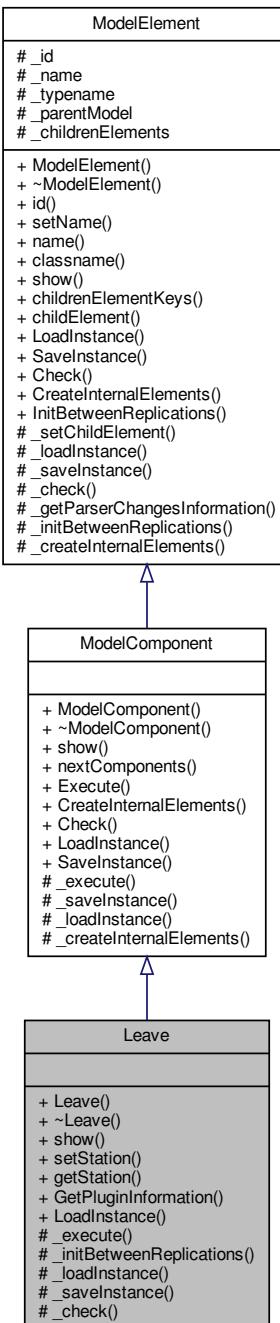
The documentation for this class was generated from the following files:

- [IntegratorDefaultImpl1.h](#)
- [IntegratorDefaultImpl1.cpp](#)

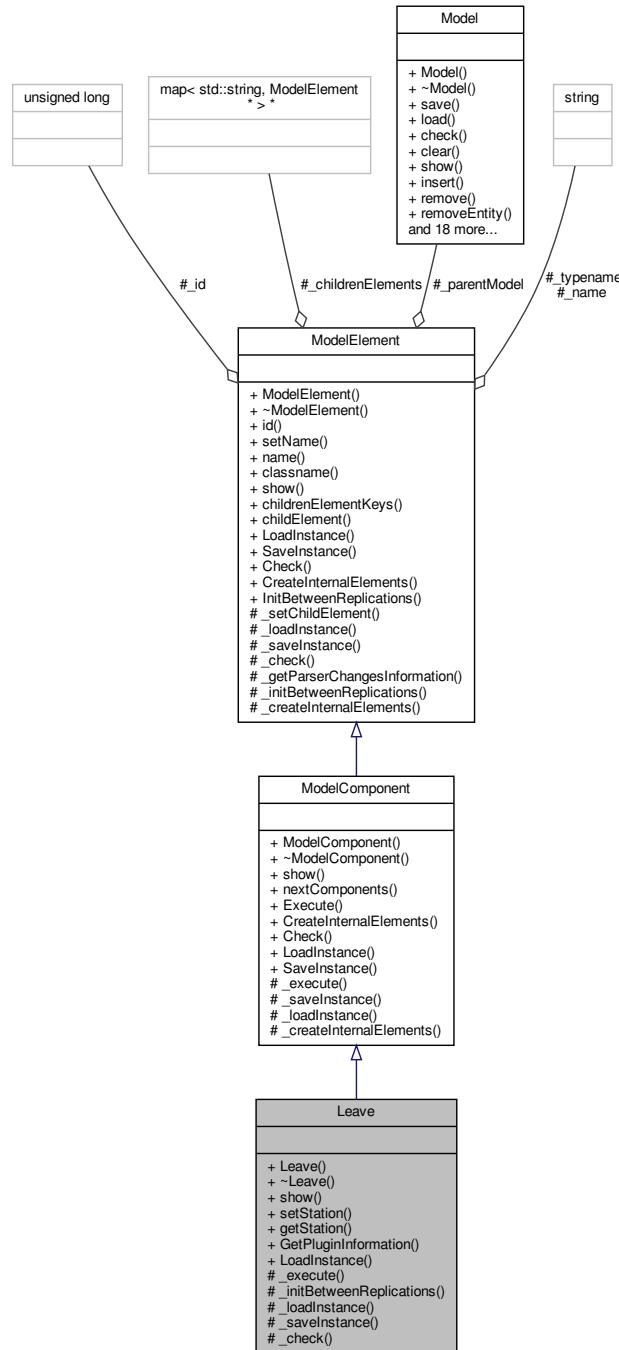
6.59 Leave Class Reference

```
#include <Leave.h>
```

Inheritance diagram for Leave:



Collaboration diagram for Leave:



Public Member Functions

- `Leave (Model *model, std::string name= "")`
- virtual `~Leave ()=default`
- virtual `std::string show ()`
- `void setStation (Station *_station)`
- `Station * getStation () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.59.1 Detailed Description

Leave module DESCRIPTION The `Leave` module is used to transfer an entity to a station or module. An entity may be transferred in two ways. It can be transferred to a module that defines a station by referencing the station and routing, conveying, or transporting to that station, or a graphical connection can be used to transfer an entity to another module. When an entity arrives at a `Leave` module, it may wait to obtain a transfer device (resource, transporter, or conveyor). When the transfer device has been obtained, the entity may experience a loading delay. Finally, the entity is transferred from this module to a destination module or station. TYPICAL USES The end of a part's production in a series of parallel processes where the part needs a forklift to be transferred to shipping PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Allocation Type of category to which the entity's incurred delay time and cost will be added. Delay Specifies a load time incurred after getting a transfer device. Units Time units used for the delay time. Transfer Out Determines whether a resource (`Seize Resource`), transporter (`Request Transporter`), or conveyor (`Access Conveyor`) is required prior to transferring the entity out of this module. Priority Indicates the priority of the module when either seizing a resource or requesting a transporter when there are entities waiting for that resource/transporter from other modules. This field is not visible when the Transfer Type is None or `Access Conveyor`. Transporter Name Name of the transporter to request. Queue Type Type of queue, either a single `Queue`, queue `Set`, Internal queue, `Attribute`, or Expression. Queue Name Name of the individual queue. Queue Set Name Name of the queue set. Set Index Defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. Queue Attribute Name The attribute name that will be evaluated to indicate which queue is to be used. Queue Expression The expression that will be evaluated to indicate which queue is to be used. Selection Rule Method of selecting among available transporters in a set. Cyclical will cycle through available members. Random will randomly select a member. Preferred Order will always select the first available member. Specific Member requires an input attribute value to specify which member of the set (previously saved in the Save `Attribute` field). Largest Distance selects the transporter farthest away, and Smallest Distance selects the closest transporter. Save `Attribute` Attribute name used to store the index number into the set of the member that is chosen. This attribute can later be referenced with the Specific Member selection rule. Active when Transfer Out is `Request Transporter`. Index `Set Attribute` name whose value identifies the index number into the set of the member requested. The entity must have a value for the attribute before utilizing this option. Resource Type Type of resource for seizing, either specifying a particular `Resource`, selecting from a pool of resources (that is, a resource `Set`), `Attribute`, or Expression. Resource Name Name of the resource to seize. Conveyor Name Name of the conveyor to access.

of Cells Number of contiguous cells the entity requires.

Connect Type Determines if the entity is to `Route`, Convey, or Transport to another station or Connect to another module. Move Time Time to route from this module to the destination station. Units Time units used for the move time. Station Type The entity's destination station type either an individual `Station`, a station based on an `Attribute` or Expression value, or By `Sequence`. Station Name Name of the individual destination station. Attribute Name The attribute name that will be evaluated to indicate the station. Expression The expression that will be evaluated to indicate the station.

Definition at line 93 of file `Leave.h`.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 Leave()

```
Leave::Leave (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Leave.cpp](#).

Here is the caller graph for this function:



6.59.2.2 ~Leave()

```
virtual Leave::~Leave ( ) [virtual], [default]
```

6.59.3 Member Function Documentation

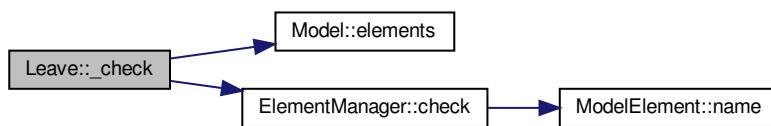
6.59.3.1 _check()

```
bool Leave::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 67 of file [Leave.cpp](#).

Here is the call graph for this function:



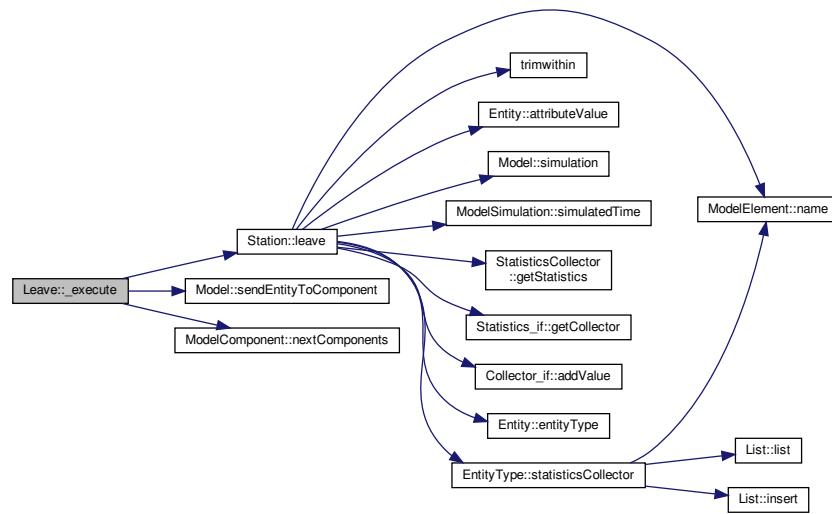
6.59.3.2 `_execute()`

```
void Leave::_execute (
    Entity * entity )  [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 43 of file [Leave.cpp](#).

Here is the call graph for this function:



6.59.3.3 `_initBetweenReplications()`

```
void Leave::_initBetweenReplications ( )  [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Leave.cpp](#).

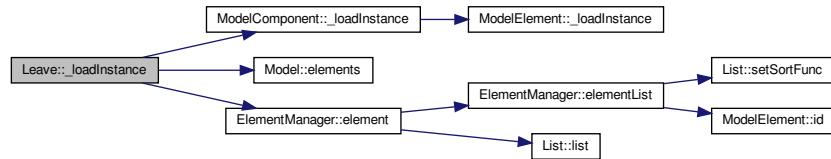
6.59.3.4 `_loadInstance()`

```
bool Leave::_loadInstance (
    std::map< std::string, std::string > * fields )  [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 48 of file [Leave.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



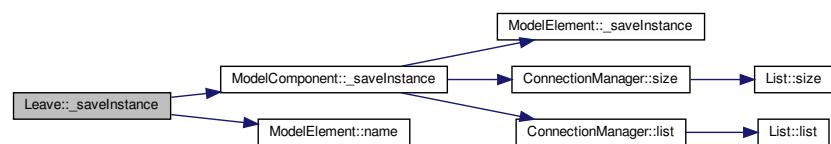
6.59.3.5 _saveInstance()

```
std::map< std::string, std::string > * Leave::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 61 of file [Leave.cpp](#).

Here is the call graph for this function:

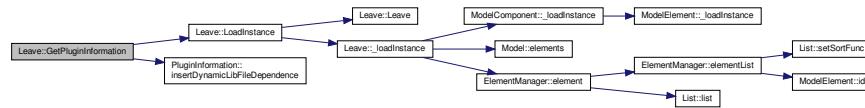


6.59.3.6 GetPluginInformation()

```
PluginInformation * Leave::GetPluginInformation ( ) [static]
```

Definition at line 73 of file [Leave.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.3.7 getStation()

`Station * Leave::getStation () const`

Definition at line 39 of file [Leave.cpp](#).

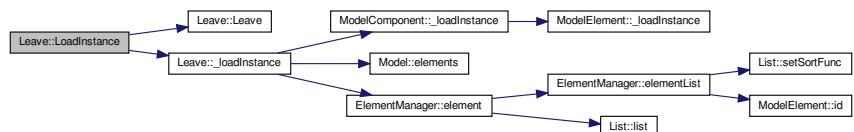
6.59.3.8 LoadInstance()

```

ModelComponent * Leave::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
  
```

Definition at line 25 of file [Leave.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

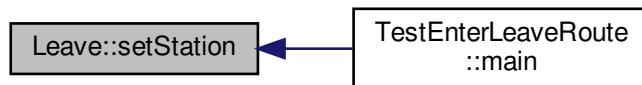


6.59.3.9 setStation()

```
void Leave::setStation (  
    Station * _station )
```

Definition at line 35 of file [Leave.cpp](#).

Here is the caller graph for this function:



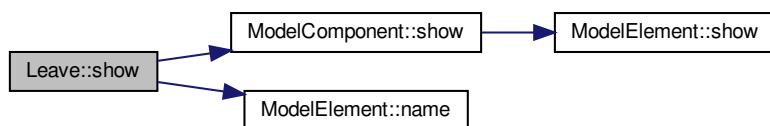
6.59.3.10 show()

```
std::string Leave::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Leave.cpp](#).

Here is the call graph for this function:



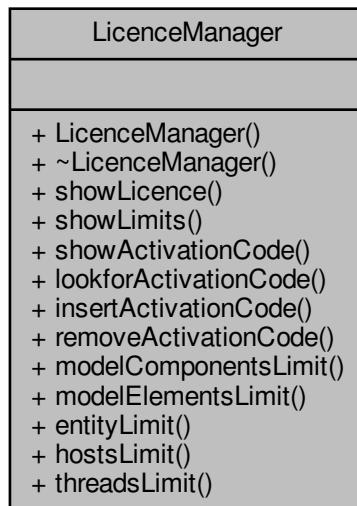
The documentation for this class was generated from the following files:

- [Leave.h](#)
- [Leave.cpp](#)

6.60 LicenceManager Class Reference

```
#include <LicenceManager.h>
```

Collaboration diagram for LicenceManager:



Public Member Functions

- `LicenceManager (Simulator *simulator)`
- `virtual ~LicenceManager ()=default`
- `const std::string showLicence () const`
- `const std::string showLimits () const`
- `const std::string showActivationCode () const`
- `bool lookforActivationCode ()`
- `bool insertActivationCode ()`
- `void removeActivationCode ()`
- `unsigned int modelComponentsLimit ()`
- `unsigned int modelElementsLimit ()`
- `unsigned int entityLimit ()`
- `unsigned int hostsLimit ()`
- `unsigned int threadsLimit ()`

6.60.1 Detailed Description

Definition at line 21 of file [LicenceManager.h](#).

6.60.2 Constructor & Destructor Documentation

6.60.2.1 LicenceManager()

```
LicenceManager::LicenceManager (   
    Simulator * simulator )
```

Definition at line 16 of file [LicenceManager.cpp](#).

6.60.2.2 ~LicenceManager()

```
virtual LicenceManager::~LicenceManager ( ) [virtual], [default]
```

6.60.3 Member Function Documentation

6.60.3.1 entityLimit()

```
unsigned int LicenceManager::entityLimit ( )
```

Definition at line 73 of file [LicenceManager.cpp](#).

6.60.3.2 hostsLimit()

```
unsigned int LicenceManager::hostsLimit ( )
```

Definition at line 77 of file [LicenceManager.cpp](#).

6.60.3.3 insertActivationCode()

```
bool LicenceManager::insertActivationCode ( )
```

Definition at line 56 of file [LicenceManager.cpp](#).

6.60.3.4 lookforActivationCode()

```
bool LicenceManager::lookforActivationCode ( )
```

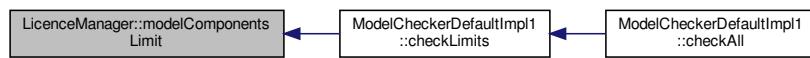
Definition at line 51 of file [LicenceManager.cpp](#).

6.60.3.5 modelComponentsLimit()

```
unsigned int LicenceManager::modelComponentsLimit ( )
```

Definition at line 65 of file [LicenceManager.cpp](#).

Here is the caller graph for this function:



6.60.3.6 modelElementsLimit()

```
unsigned int LicenceManager::modelElementsLimit ( )
```

Definition at line 69 of file [LicenceManager.cpp](#).

Here is the caller graph for this function:



6.60.3.7 removeActivationCode()

```
void LicenceManager::removeActivationCode ( )
```

Definition at line 61 of file [LicenceManager.cpp](#).

6.60.3.8 showActivationCode()

```
const std::string LicenceManager::showActivationCode ( ) const
```

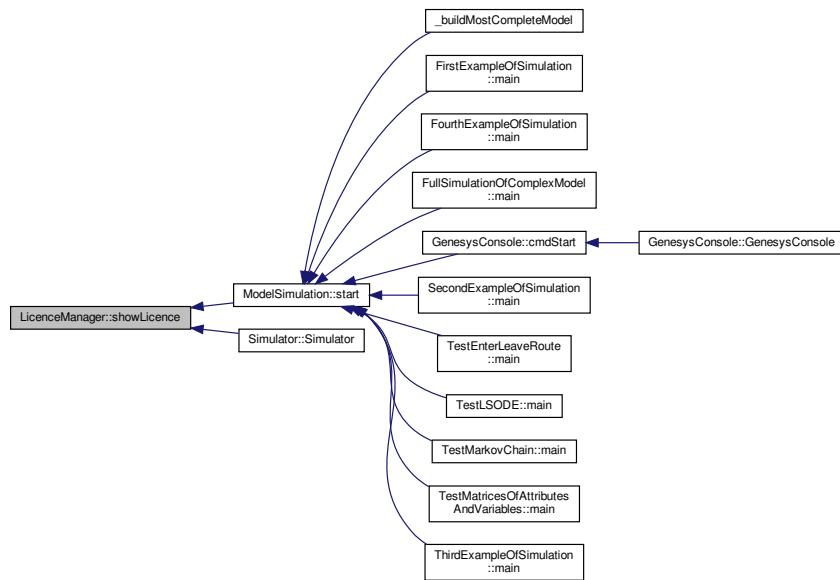
Definition at line 46 of file [LicenceManager.cpp](#).

6.60.3.9 showLicence()

```
const std::string LicenceManager::showLicence () const
```

Definition at line 32 of file [LicenceManager.cpp](#).

Here is the caller graph for this function:

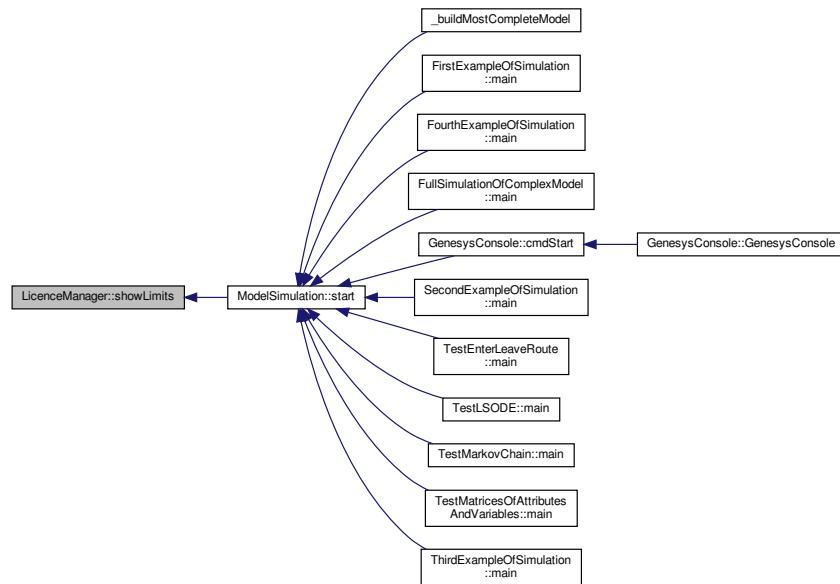


6.60.3.10 showLimits()

```
const std::string LicenceManager::showLimits () const
```

Definition at line 36 of file [LicenceManager.cpp](#).

Here is the caller graph for this function:



6.60.3.11 threadsLimit()

```
unsigned int LicenceManager::threadsLimit ( )
```

Definition at line 81 of file [LicenceManager.cpp](#).

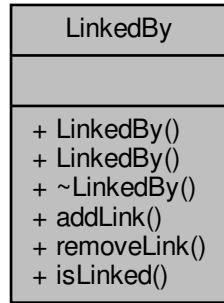
The documentation for this class was generated from the following files:

- [LicenceManager.h](#)
- [LicenceManager.cpp](#)

6.61 LinkedBy Class Reference

```
#include <LinkedBy.h>
```

Collaboration diagram for LinkedBy:



Public Member Functions

- [LinkedBy \(\)](#)
- [LinkedBy \(const LinkedBy &orig\)](#)
- [virtual ~LinkedBy \(\)](#)
- [void addLink \(\)](#)
- [void removeLink \(\)](#)
- [bool isLinked \(\)](#)

6.61.1 Detailed Description

Definition at line 17 of file [LinkedBy.h](#).

6.61.2 Constructor & Destructor Documentation

6.61.2.1 [LinkedBy\(\) \[1/2\]](#)

`LinkedBy::LinkedBy ()`

Definition at line 16 of file [LinkedBy.cpp](#).

6.61.2.2 [LinkedBy\(\) \[2/2\]](#)

`LinkedBy::LinkedBy (`
 `const LinkedBy & orig)`

Definition at line 19 of file [LinkedBy.cpp](#).

6.61.2.3 ~LinkedBy()

```
LinkedBy::~LinkedBy ( ) [virtual]
```

Definition at line 22 of file [LinkedBy.cpp](#).

6.61.3 Member Function Documentation

6.61.3.1 addLink()

```
void LinkedBy::addLink ( )
```

Definition at line 25 of file [LinkedBy.cpp](#).

6.61.3.2 isLinked()

```
bool LinkedBy::isLinked ( )
```

Definition at line 33 of file [LinkedBy.cpp](#).

6.61.3.3 removeLink()

```
void LinkedBy::removeLink ( )
```

Definition at line 29 of file [LinkedBy.cpp](#).

The documentation for this class was generated from the following files:

- [LinkedBy.h](#)
- [LinkedBy.cpp](#)

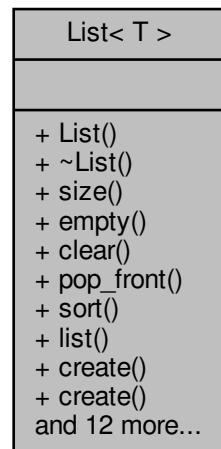
6.62 List< T > Class Template Reference

```
#include <List.h>
```

Inheritance diagram for List< T >:



Collaboration diagram for List< T >:



Public Types

- using `CompFunct` = `std::function< bool(const T, const T) >`

Public Member Functions

- `List ()`
 - virtual `~List ()`=default
 - unsigned int `size ()`
 - bool `empty ()`
 - void `clear ()`
 - void `pop_front ()`
 - template<class Compare >
void `sort (Compare comp)`
 - `std::list<T > * list () const`
 - T `create ()`

- template<typename U >
T **create** (U arg)
- std::string **show** ()
- std::list< T >::iterator **find** (T element)
- void **insert** (T element)
- void **remove** (T element)
- void **setAtRank** (unsigned int rank, T element)
- T **getAtRank** (unsigned int rank)
- T **next** ()
- T **front** ()
- T **last** ()
- T **previous** ()
- T **current** ()
- void **setSortFunc** (CompFunct _sortFunc)

6.62.1 Detailed Description

```
template<typename T>
class List< T >
```

List corresponds to an extended version of the list that must guarantee the consistency of the elements that make up the simulation model.

Definition at line 32 of file [List.h](#).

6.62.2 Member Typedef Documentation

6.62.2.1 CompFunct

```
template<typename T>
using List< T >::CompFunct = std::function<bool(const T, const T) >
```

Definition at line 34 of file [List.h](#).

6.62.3 Constructor & Destructor Documentation

6.62.3.1 List()

```
template<typename T >
List< T >::List ( )
```

Definition at line 74 of file [List.h](#).

6.62.3.2 ~List()

```
template<typename T>
virtual List< T >::~List ( ) [virtual], [default]
```

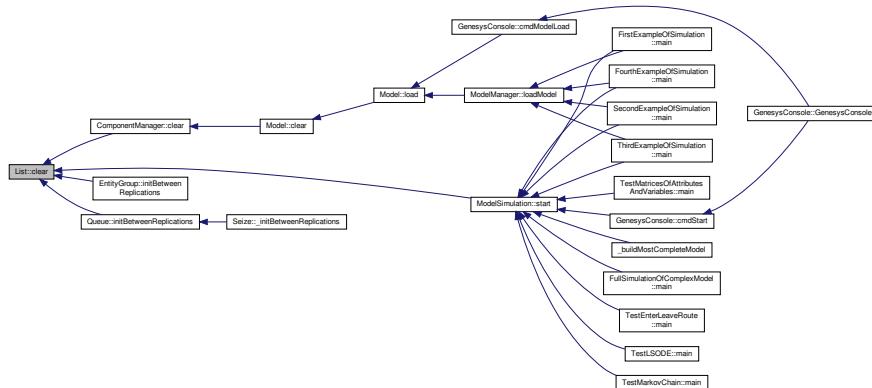
6.62.4 Member Function Documentation

6.62.4.1 clear()

```
template<typename T >
void List< T >::clear ( )
```

Definition at line 142 of file [List.h](#).

Here is the caller graph for this function:



6.62.4.2 create() [1/2]

```
template<typename T >
T List< T >::create ( )
```

Definition at line 137 of file [List.h](#).

6.62.4.3 create() [2/2]

```
template<typename T >
template<typename U >
T List< T >::create (
    U arg )
```

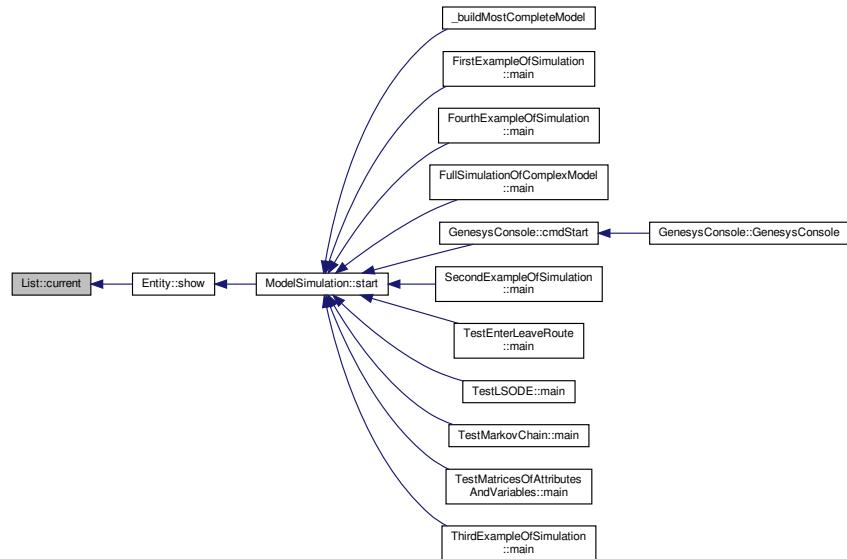
Definition at line 244 of file [List.h](#).

6.62.4.4 current()

```
template<typename T >
T List< T >::current ( )
```

Definition at line 232 of file [List.h](#).

Here is the caller graph for this function:

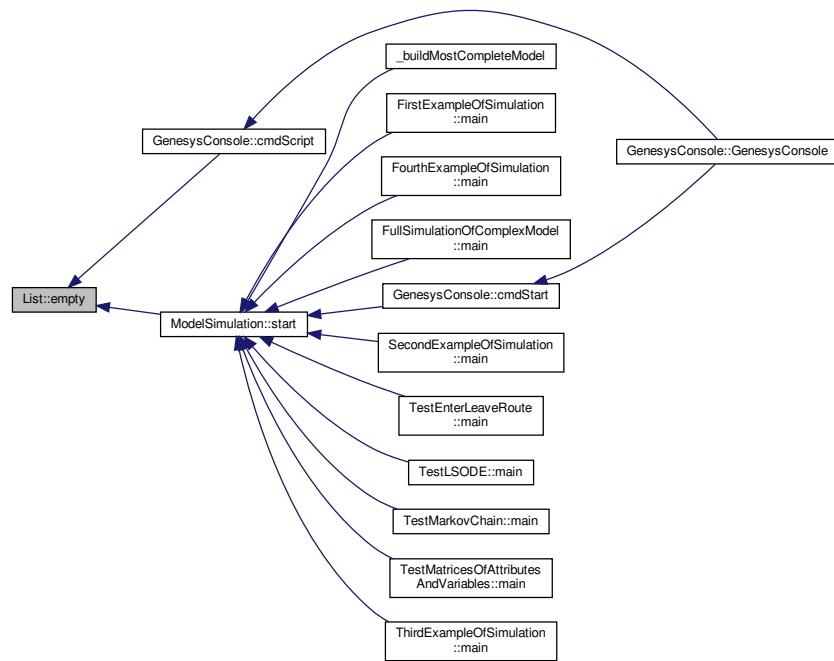


6.62.4.5 empty()

```
template<typename T >
bool List< T >::empty ( )
```

Definition at line 115 of file [List.h](#).

Here is the caller graph for this function:



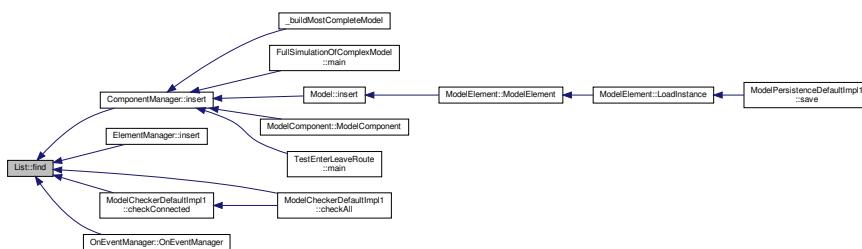
6.62.4.6 `find()`

```

template<typename T>
std::list< T >::iterator List< T >::find (
    T element )
  
```

Definition at line 183 of file [List.h](#).

Here is the caller graph for this function:

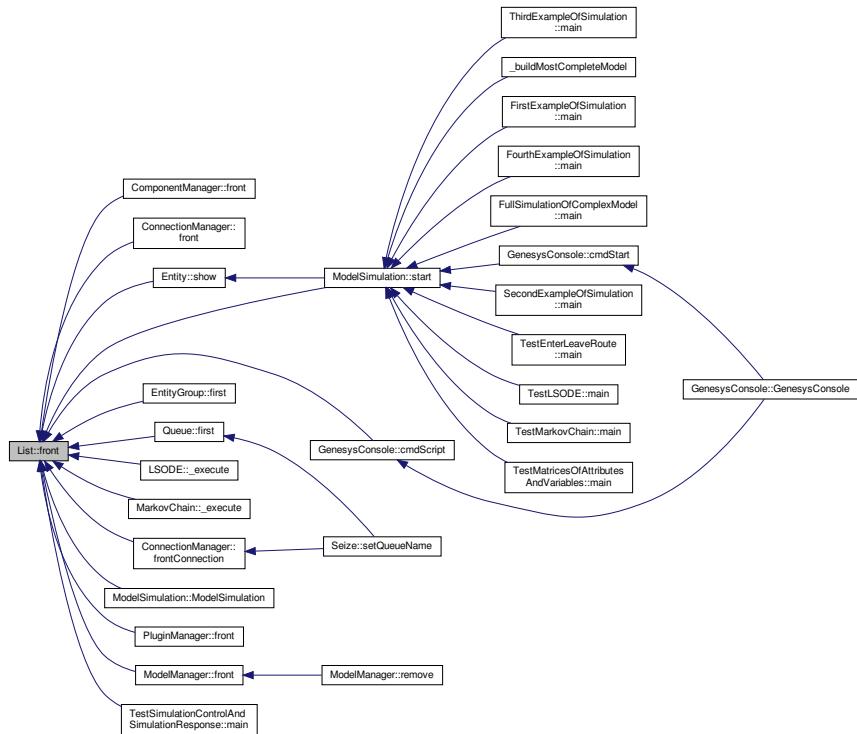


6.62.4.7 front()

```
template<typename T>
T List< T >::front ( )
```

Definition at line 208 of file [List.h](#).

Here is the caller graph for this function:

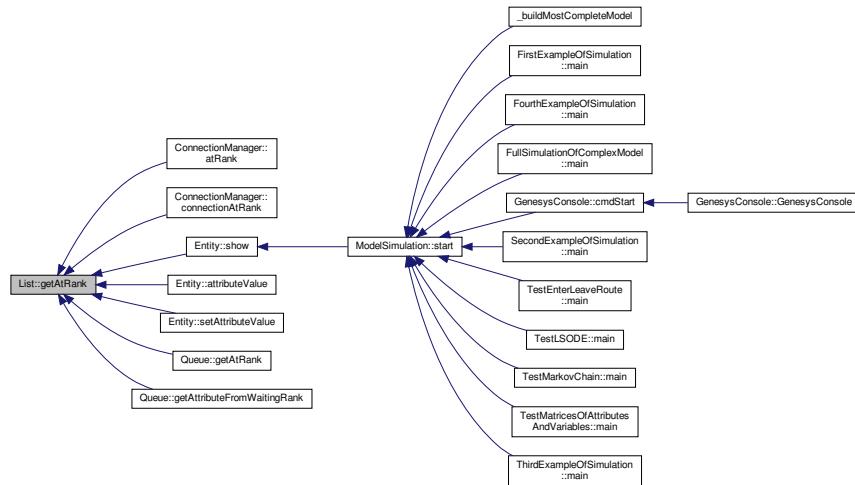


6.62.4.8 getAtRank()

```
template<typename T>
T List< T >::getAtRank (
    unsigned int rank )
```

Definition at line 147 of file [List.h](#).

Here is the caller graph for this function:



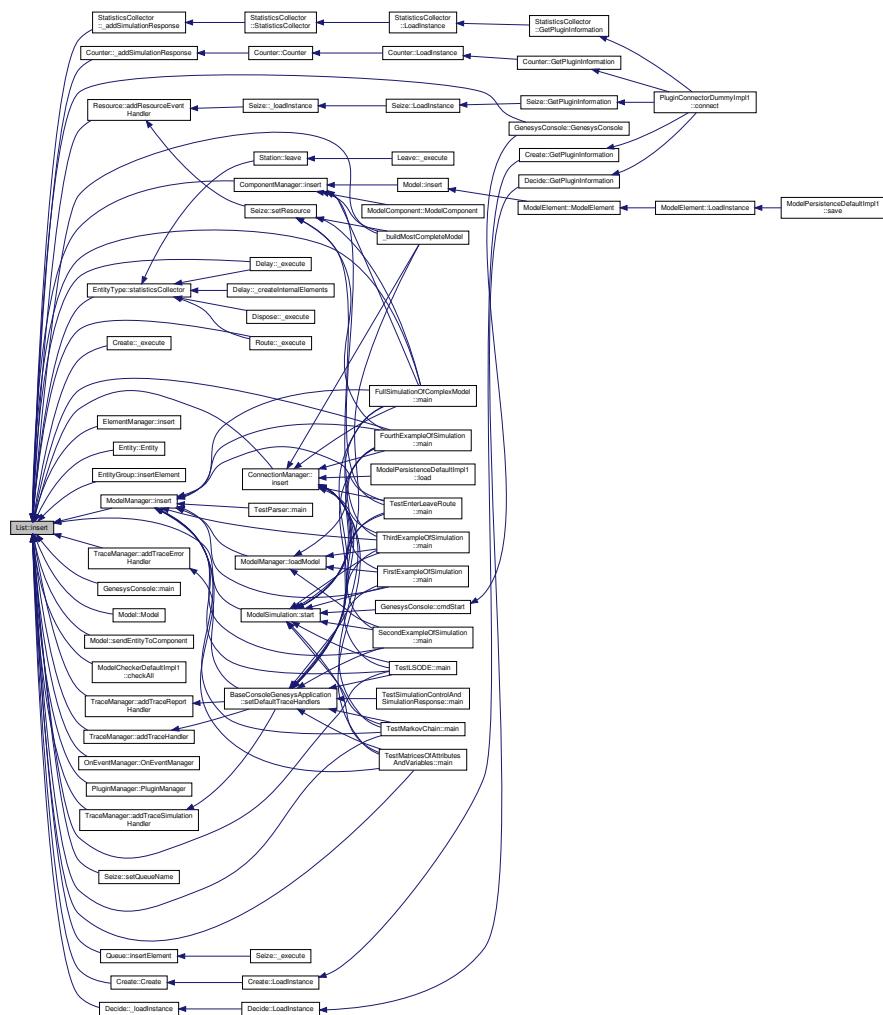
6.62.4.9 insert()

```

template<typename T>
void List< T >::insert (
    T element )
  
```

Definition at line 110 of file [List.h](#).

Here is the caller graph for this function:

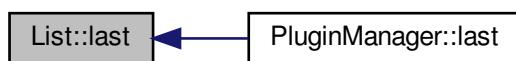


6.62.4.10 last()

```
template<typename T >
T List< T >::last ( )
```

Definition at line 217 of file [List.h](#).

Here is the caller graph for this function:

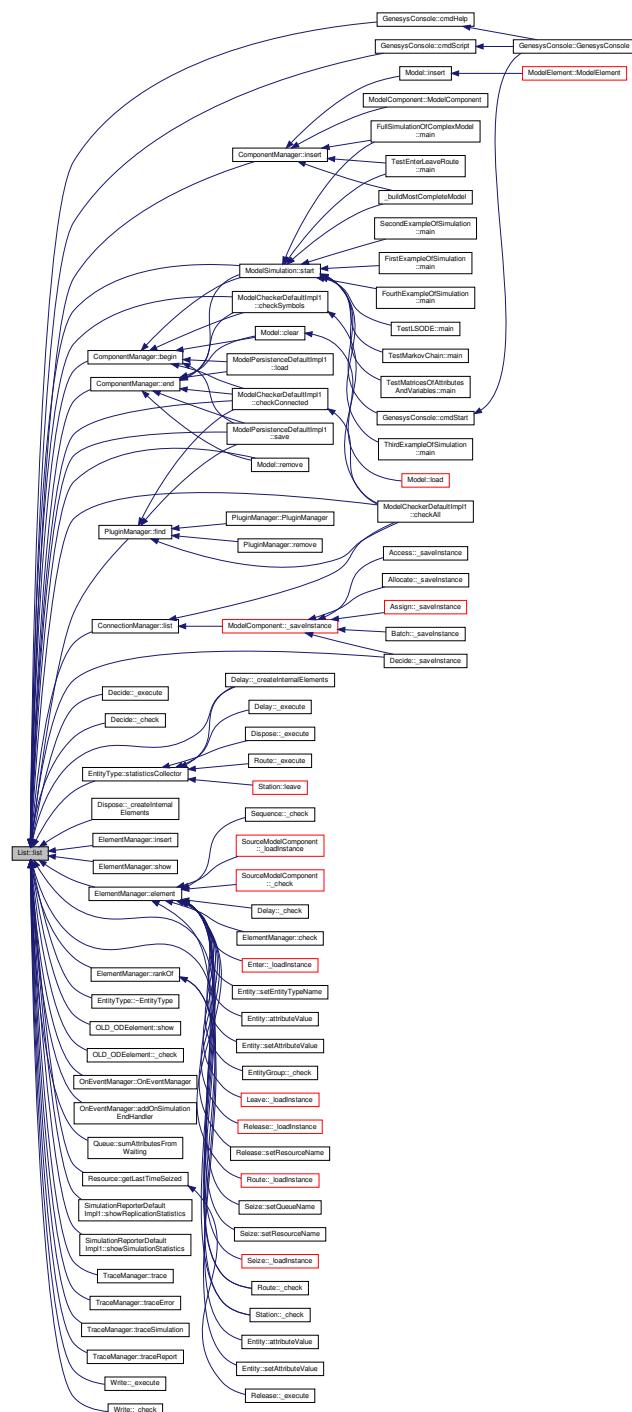


6.62.4.11 list()

```
template<typename T >
std::list< T > * List< T >::list ( ) const
```

Definition at line 81 of file [List.h](#).

Here is the caller graph for this function:

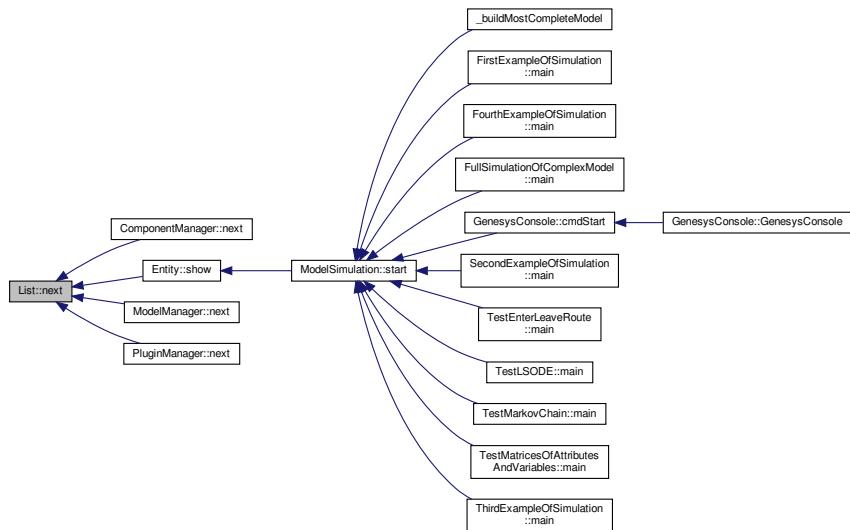


6.62.4.12 next()

```
template<typename T>
T List< T >::next ( )
```

Definition at line 173 of file [List.h](#).

Here is the caller graph for this function:

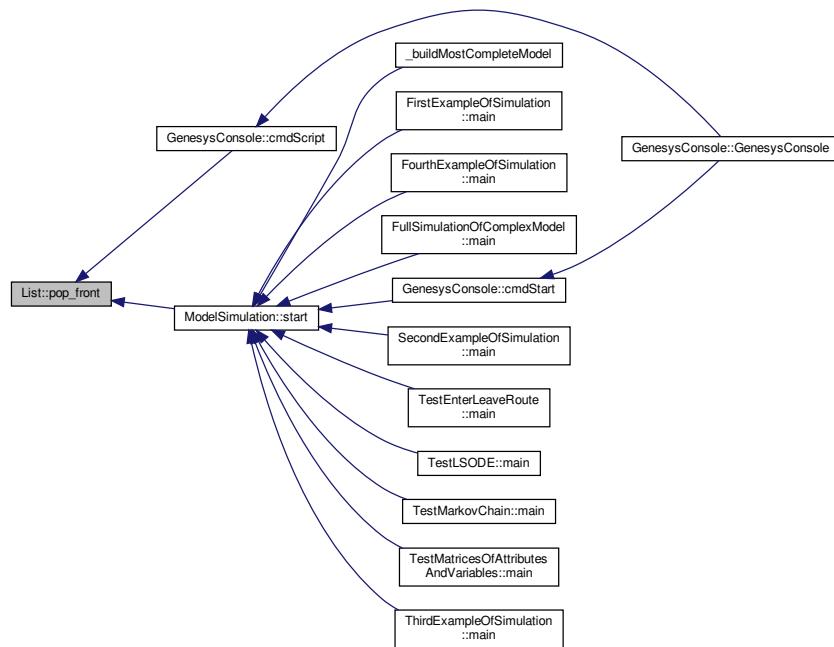


6.62.4.13 pop_front()

```
template<typename T>
void List< T >::pop_front ( )
```

Definition at line 120 of file [List.h](#).

Here is the caller graph for this function:



6.62.4.14 previous()

```
template<typename T >
T List< T >::previous ( )
```

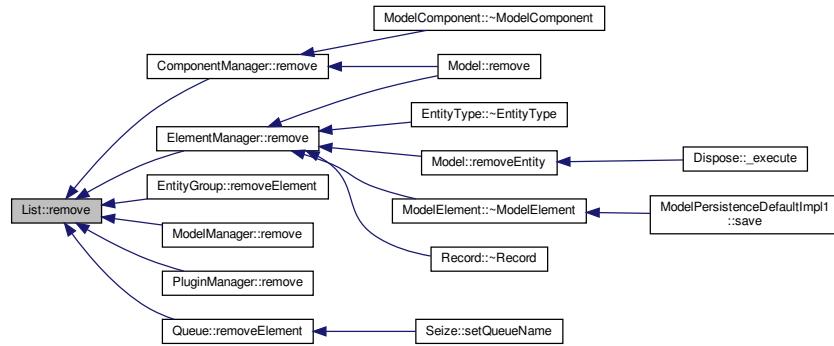
Definition at line 226 of file [List.h](#).

6.62.4.15 remove()

```
template<typename T>
void List< T >::remove (
    T element )
```

Definition at line 129 of file [List.h](#).

Here is the caller graph for this function:



6.62.4.16 setAtRank()

```

template<typename T>
void List< T >::setAtRank (
    unsigned int rank,
    T element )
  
```

Definition at line 160 of file [List.h](#).

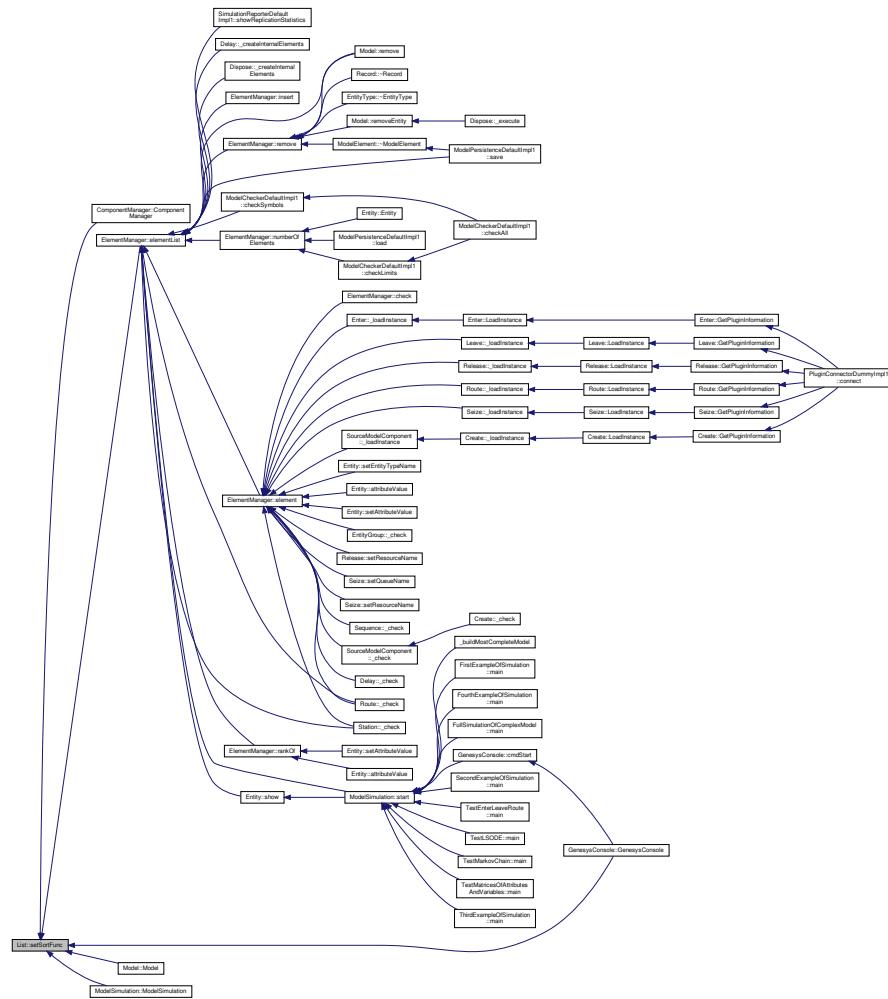
6.62.4.17 setSortFunc()

```

template<typename T >
void List< T >::setSortFunc (
    CompFunct _sortFunc )
  
```

Definition at line 238 of file [List.h](#).

Here is the caller graph for this function:

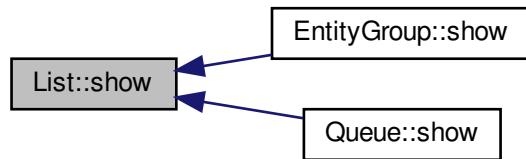


6.62.4.18 show()

```
template<typename T >
std::string List< T >::show ( )
```

Definition at line 99 of file [List.h](#).

Here is the caller graph for this function:

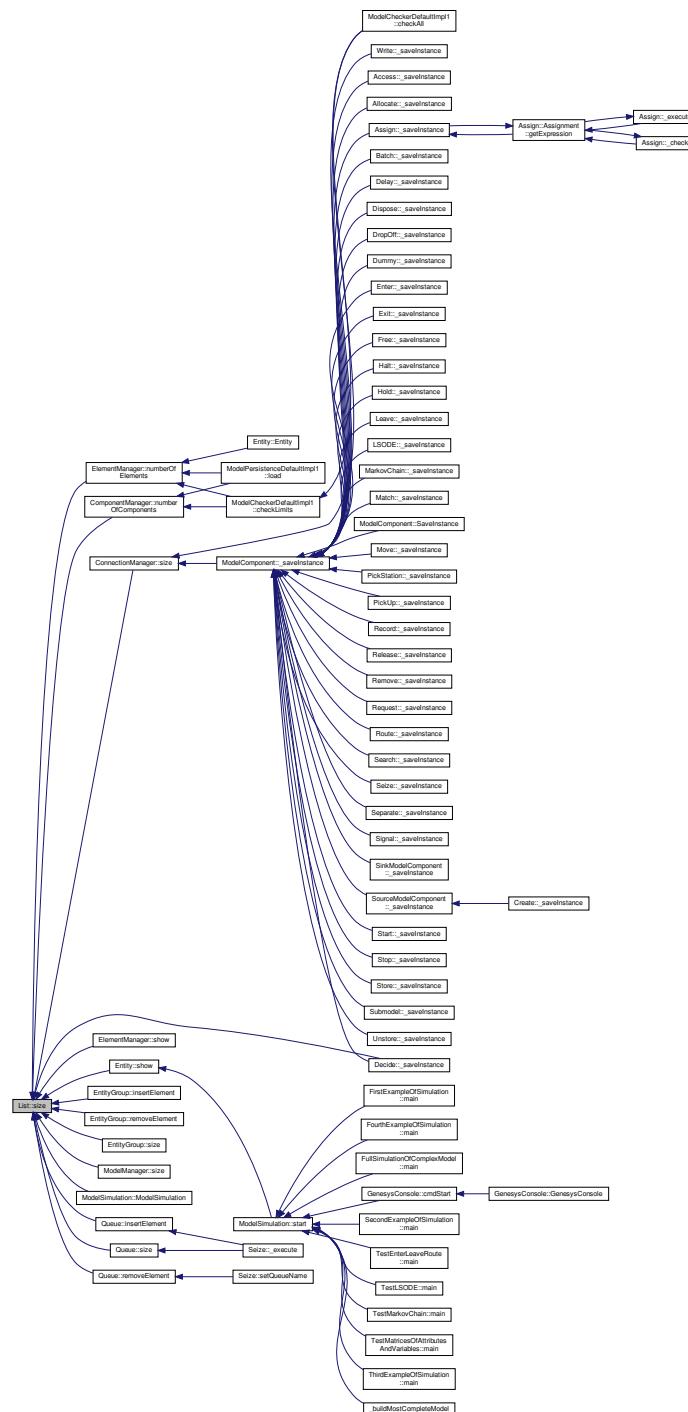


6.62.4.19 size()

```
template<typename T>
unsigned int List< T >::size ( )
```

Definition at line 86 of file [List.h](#).

Here is the caller graph for this function:



6.62.4.20 sort()

```
template<typename T >
template<class Compare >
void List< T >::sort (
    Compare comp )
```

Definition at line 250 of file [List.h](#).

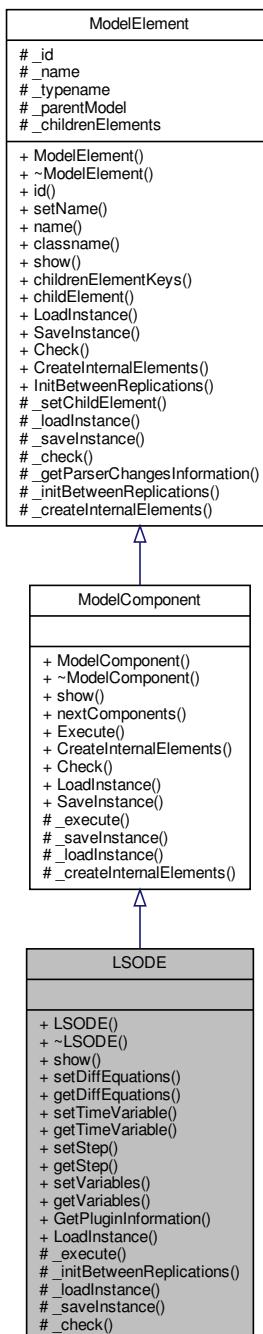
The documentation for this class was generated from the following file:

- [List.h](#)

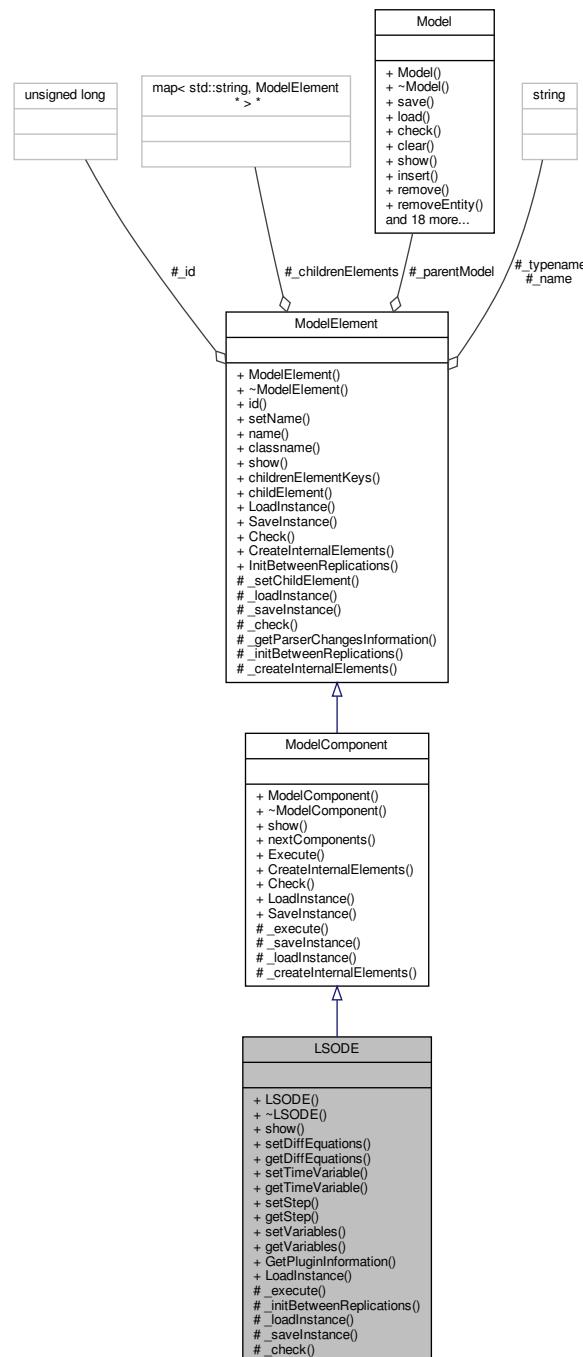
6.63 LSODE Class Reference

```
#include <LSODE.h>
```

Inheritance diagram for LSODE:



Collaboration diagram for LSODE:



Public Member Functions

- **LSODE (Model *model, std::string name="")**
- virtual **~LSODE ()=default**
- virtual std::string **show ()**
- void **setDiffEquations (Formula *formula)**
- **Formula * getDiffEquations () const**

- void `setTimeVariable (Variable *_timeVariable)`
- `Variable * getTimeVariable () const`
- void `setStep (double _step)`
- double `getStep () const`
- void `setVariables (Variable *_variables)`
- `Variable * getVariables () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.63.1 Detailed Description

This component ...

Definition at line 24 of file [LSODE.h](#).

6.63.2 Constructor & Destructor Documentation

6.63.2.1 LSODE()

```
LSODE::LSODE (
    Model * model,
    std::string name = "")
```

Definition at line 17 of file [LSODE.cpp](#).

Here is the caller graph for this function:



6.63.2.2 ~LSODE()

```
virtual LSODE::~LSODE ( ) [virtual], [default]
```

6.63.3 Member Function Documentation

6.63.3.1 _check()

```
bool LSODE::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 149 of file [LSODE.cpp](#).

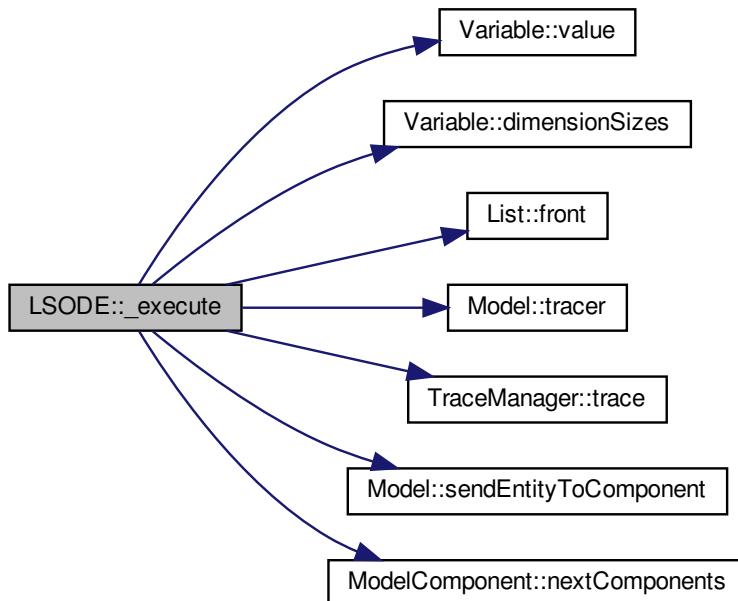
6.63.3.2 _execute()

```
void LSODE::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 116 of file [LSODE.cpp](#).

Here is the call graph for this function:



6.63.3.3 `_initBetweenReplications()`

```
void LSODE::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 140 of file [LSODE.cpp](#).

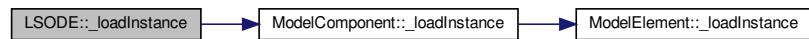
6.63.3.4 `_loadInstance()`

```
bool LSODE::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 132 of file [LSODE.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



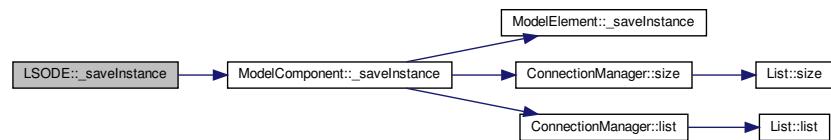
6.63.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * LSODE::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 143 of file [LSODE.cpp](#).

Here is the call graph for this function:



6.63.3.6 getDiffEquations()

```
Formula * LSODE::getDiffEquations ( ) const
```

Definition at line 38 of file [LSODE.cpp](#).

6.63.3.7 GetPluginInformation()

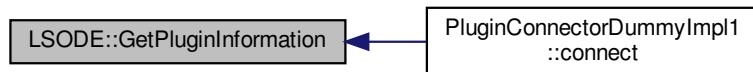
```
PluginInformation * LSODE::GetPluginInformation ( ) [static]
```

Definition at line 155 of file [LSODE.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.63.3.8 getStep()

```
double LSODE::getStep ( ) const
```

Definition at line 54 of file [LSODE.cpp](#).

6.63.3.9 getTimeVariable()

```
Variable * LSODE::getTimeVariable ( ) const
```

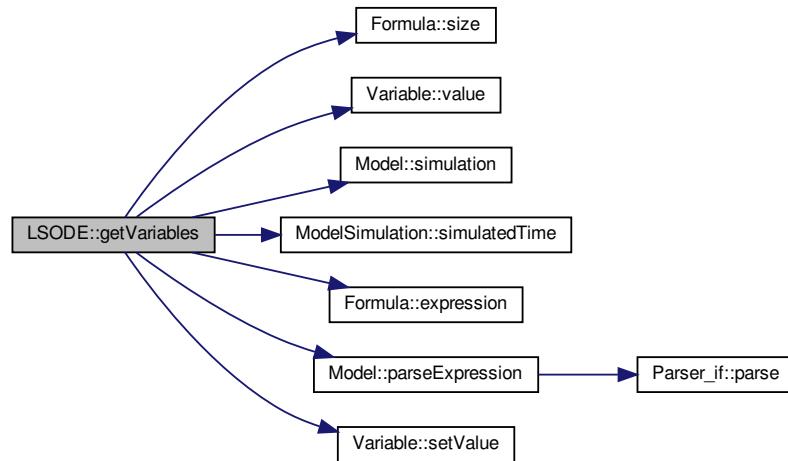
Definition at line 46 of file [LSODE.cpp](#).

6.63.3.10 getVariables()

```
Variable * LSODE::getVariables ( ) const
```

Definition at line 62 of file [LSODE.cpp](#).

Here is the call graph for this function:



6.63.3.11 LoadInstance()

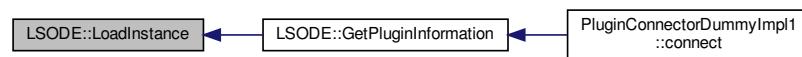
```
ModelComponent * LSODE::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 24 of file [LSODE.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.63.3.12 setDiffEquations()

```
void LSODE::setDiffEquations (
    Formula * formula )
```

Definition at line [34](#) of file [LSODE.cpp](#).

Here is the caller graph for this function:



6.63.3.13 setStep()

```
void LSODE::setStep (
    double _step )
```

Definition at line [50](#) of file [LSODE.cpp](#).

Here is the caller graph for this function:



6.63.3.14 setTimeVariable()

```
void LSODE::setTimeVariable (
    Variable * _timeVariable )
```

Definition at line [42](#) of file [LSODE.cpp](#).

Here is the caller graph for this function:

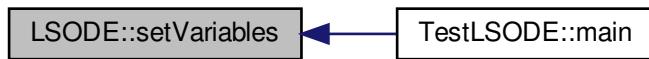


6.63.3.15 setVariables()

```
void LSODE::setVariables (
    Variable * _variables )
```

Definition at line 58 of file [LSODE.cpp](#).

Here is the caller graph for this function:



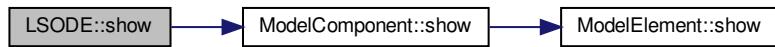
6.63.3.16 show()

```
std::string LSODE::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [LSODE.cpp](#).

Here is the call graph for this function:



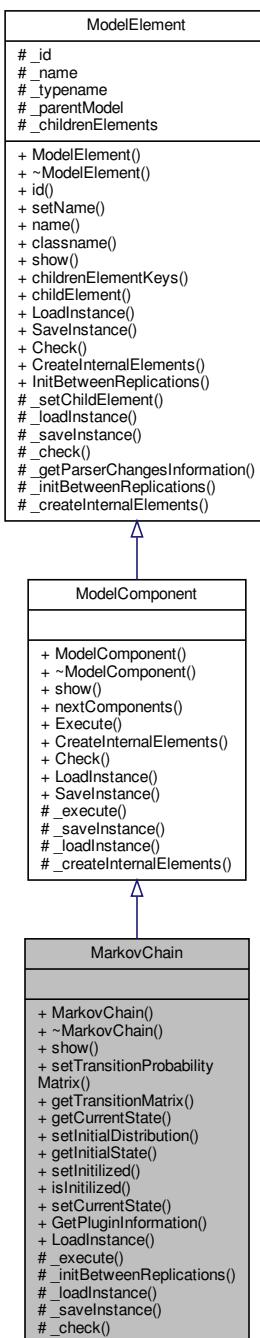
The documentation for this class was generated from the following files:

- [LSODE.h](#)
- [LSODE.cpp](#)

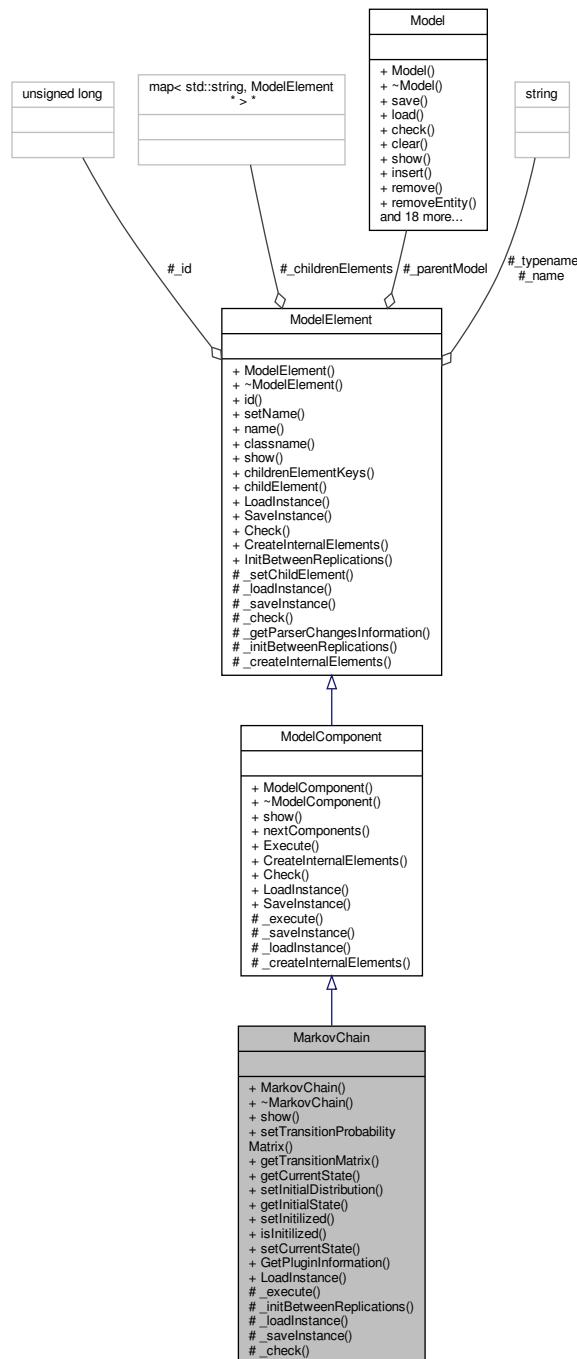
6.64 MarkovChain Class Reference

```
#include <MarkovChain.h>
```

Inheritance diagram for MarkovChain:



Collaboration diagram for MarkovChain:



Public Member Functions

- `MarkovChain (Model *model, std::string name="")`
- `virtual ~MarkovChain ()=default`
- `virtual std::string show ()`
- `void setTransitionProbabilityMatrix (Variable *_transitionMatrix)`
- `Variable * getTransitionMatrix () const`

- `Variable * getCurrentState () const`
- `void setInitialDistribution (Variable *_initialDistribution)`
- `Variable * getInitialState () const`
- `void setInitialized (bool _initialized)`
- `bool isInitialized () const`
- `void setCurrentState (Variable *_currentState)`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.64.1 Detailed Description

Definition at line 20 of file [MarkovChain.h](#).

6.64.2 Constructor & Destructor Documentation

6.64.2.1 `MarkovChain()`

```
MarkovChain::MarkovChain (
    Model * model,
    std::string name = "" )
```

Definition at line 21 of file [MarkovChain.cpp](#).

Here is the caller graph for this function:



6.64.2.2 ~MarkovChain()

```
virtual MarkovChain::~MarkovChain ( ) [virtual], [default]
```

6.64.3 Member Function Documentation

6.64.3.1 _check()

```
bool MarkovChain::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 125 of file [MarkovChain.cpp](#).

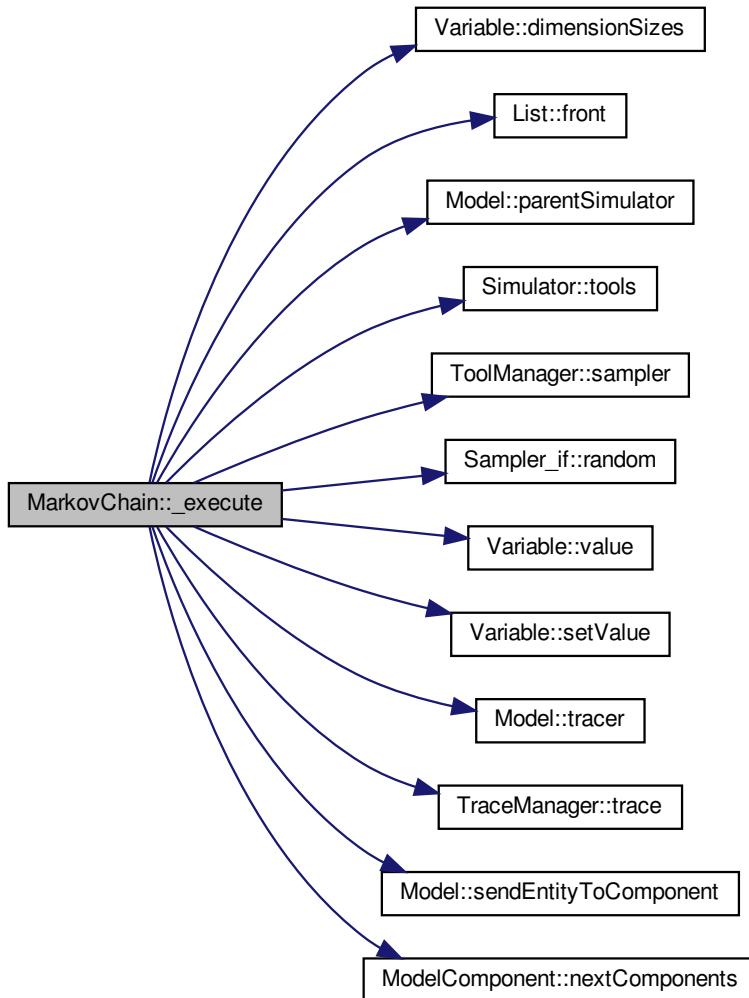
6.64.3.2 _execute()

```
void MarkovChain::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 70 of file [MarkovChain.cpp](#).

Here is the call graph for this function:



6.64.3.3 `_initBetweenReplications()`

```
void MarkovChain::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 115 of file [MarkovChain.cpp](#).

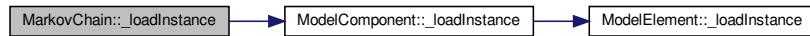
6.64.3.4 `_loadInstance()`

```
bool MarkovChain::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 107 of file [MarkovChain.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



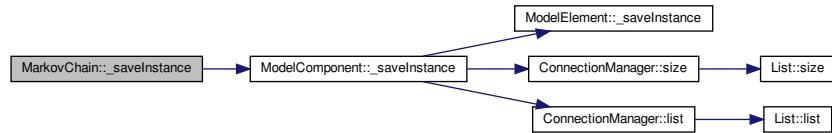
6.64.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * MarkovChain::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 119 of file [MarkovChain.cpp](#).

Here is the call graph for this function:



6.64.3.6 `getCurrentState()`

```
Variable * MarkovChain::getCurrentState ( ) const
```

Definition at line 46 of file [MarkovChain.cpp](#).

6.64.3.7 `getInitialState()`

```
Variable * MarkovChain::getInitialState ( ) const
```

Definition at line 58 of file [MarkovChain.cpp](#).

6.64.3.8 `GetPluginInformation()`

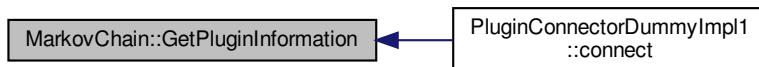
```
PluginInformation * MarkovChain::GetPluginInformation ( ) [static]
```

Definition at line 131 of file [MarkovChain.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.64.3.9 `getTransitionMatrix()`

```
Variable * MarkovChain::getTransitionMatrix ( ) const
```

Definition at line 42 of file [MarkovChain.cpp](#).

6.64.3.10 `isInitialized()`

```
bool MarkovChain::isInitialized ( ) const
```

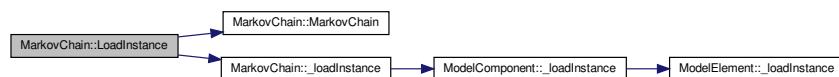
Definition at line 66 of file [MarkovChain.cpp](#).

6.64.3.11 LoadInstance()

```
ModelComponent * MarkovChain::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 28 of file [MarkovChain.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.64.3.12 setCurrentState()

```
void MarkovChain::setCurrentState (
    Variable * _currentState )
```

Definition at line 50 of file [MarkovChain.cpp](#).

Here is the caller graph for this function:



6.64.3.13 setInitialDistribution()

```
void MarkovChain::setInitialDistribution (
    Variable * _initialDistribution )
```

Definition at line 54 of file [MarkovChain.cpp](#).

Here is the caller graph for this function:



6.64.3.14 setInitialized()

```
void MarkovChain::setInitialized (
    bool _initialized )
```

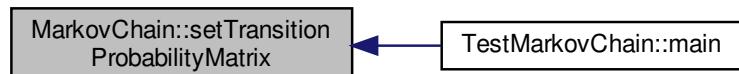
Definition at line 62 of file [MarkovChain.cpp](#).

6.64.3.15 setTransitionProbabilityMatrix()

```
void MarkovChain::setTransitionProbabilityMatrix (
    Variable * _transitionMatrix )
```

Definition at line 38 of file [MarkovChain.cpp](#).

Here is the caller graph for this function:



6.64.3.16 show()

```
std::string MarkovChain::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [24](#) of file [MarkovChain.cpp](#).

Here is the call graph for this function:



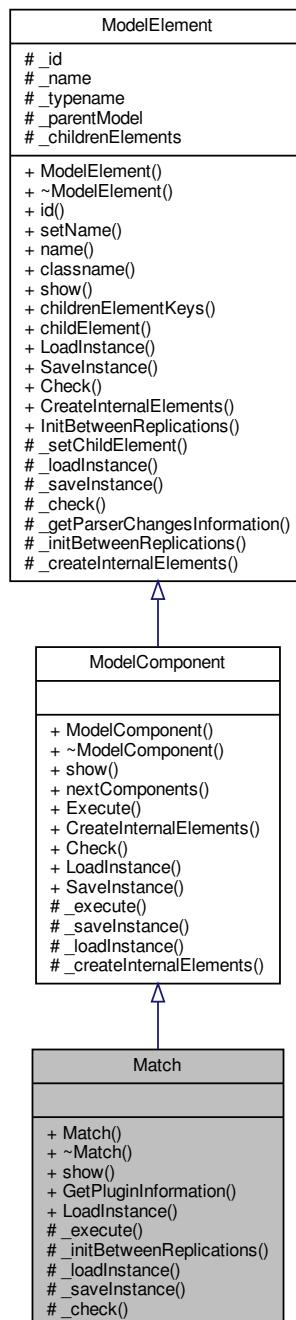
The documentation for this class was generated from the following files:

- [MarkovChain.h](#)
- [MarkovChain.cpp](#)

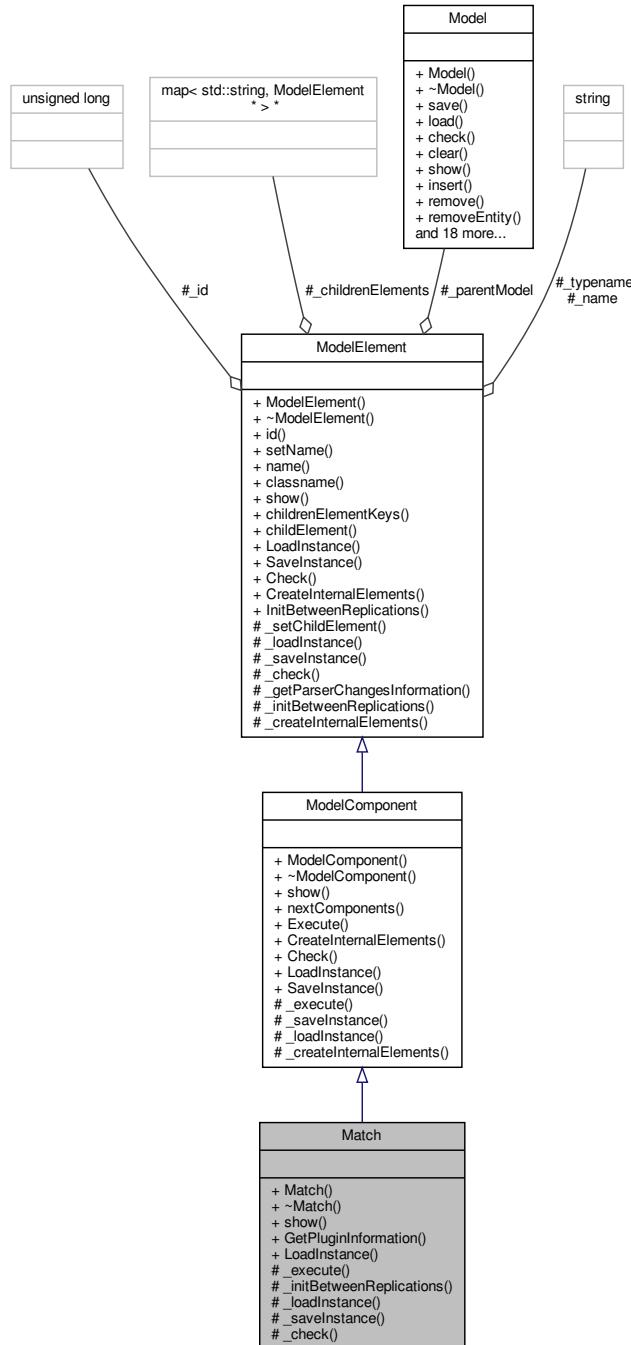
6.65 Match Class Reference

```
#include <Match.h>
```

Inheritance diagram for Match:



Collaboration diagram for Match:



Public Member Functions

- [Match \(Model *model, std::string name=""\)](#)
- virtual [~Match \(\)=default](#)
- virtual std::string [show \(\)](#)

Static Public Member Functions

- static [PluginInformation * GetPluginInformation \(\)](#)
- static [ModelComponent * LoadInstance \(Model *model, std::map< std::string, std::string > *fields\)](#)

Protected Member Functions

- virtual void [_execute \(Entity *entity\)](#)
- virtual void [_initBetweenReplications \(\)](#)
- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)

Additional Inherited Members

6.65.1 Detailed Description

[Match](#) module DESCRIPTION The [Match](#) module brings together a specified number of entities waiting in different queues. The match may be accomplished when there is at least one entity in each of the desired queues. Additionally, an attribute may be specified such that the entities waiting in the queues must have the same attribute values before the match is initiated. When an entity arrives at the [Match](#) module, it is placed in one of up to five associated queues, based on the entry point to which it is connected. Entities will remain in their respective queues until a match exists. Once a match exists, one entity from each queue is released to be matched. The matched entities are then synchronized to depart from the module. TYPICAL USES Assembling a part Gathering various products for a customer order Synchronizing a customer exit with a filled order Prompt Description Name Unique module identifier displayed on the module shape. Number to [Match](#) Number of matching entities that must reside in different queues before a match may be completed. Type Method for matching the incoming entities. If Type is Any Entities, one entity must reside in each queue for a match to be made. If Type is Based on [Attribute](#), one entity must reside in each queue with the same attribute value. Attribute Name [Attribute](#) name that is used for identifying an arriving entity's match value. Applies only when Type is Based on [Attribute](#).

Definition at line 47 of file [Match.h](#).

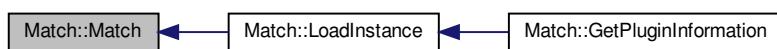
6.65.2 Constructor & Destructor Documentation

6.65.2.1 Match()

```
Match::Match (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Match.cpp](#).

Here is the caller graph for this function:



6.65.2.2 ~Match()

```
virtual Match::~Match ( ) [virtual], [default]
```

6.65.3 Member Function Documentation

6.65.3.1 _check()

```
bool Match::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Match.cpp](#).

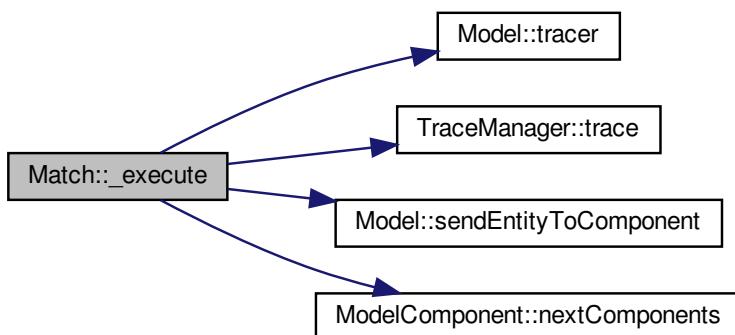
6.65.3.2 _execute()

```
void Match::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Match.cpp](#).

Here is the call graph for this function:



6.65.3.3 `_initBetweenReplications()`

```
void Match::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Match.cpp](#).

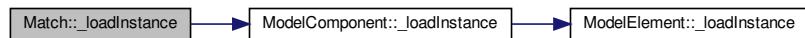
6.65.3.4 `_loadInstance()`

```
bool Match::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

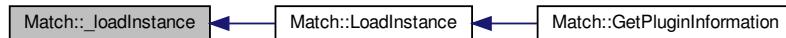
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Match.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



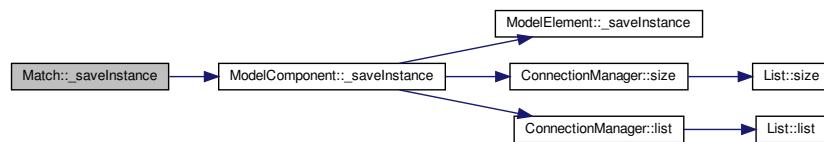
6.65.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Match::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Match.cpp](#).

Here is the call graph for this function:

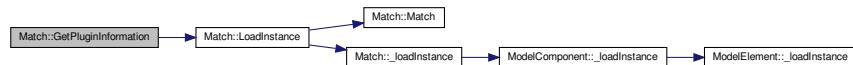


6.65.3.6 GetPluginInformation()

```
PluginInformation * Match::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Match.cpp](#).

Here is the call graph for this function:



6.65.3.7 LoadInstance()

```
ModelComponent * Match::LoadInstance ( Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Match.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



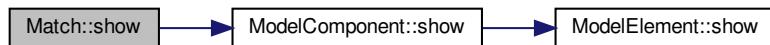
6.65.3.8 show()

```
std::string Match::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [Match.cpp](#).

Here is the call graph for this function:



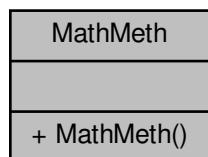
The documentation for this class was generated from the following files:

- [Match.h](#)
- [Match.cpp](#)

6.66 MathMeth Class Reference

```
#include <MathMeth.h>
```

Collaboration diagram for MathMeth:



Public Member Functions

- [MathMeth \(\)](#)

6.66.1 Detailed Description

Definition at line [19](#) of file [MathMeth.h](#).

6.66.2 Constructor & Destructor Documentation

6.66.2.1 MathMeth()

```
MathMeth::MathMeth ( )
```

Definition at line 16 of file [MathMeth.cpp](#).

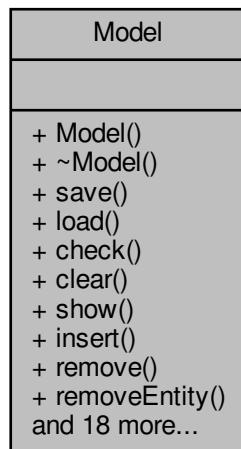
The documentation for this class was generated from the following files:

- [MathMeth.h](#)
- [MathMeth.cpp](#)

6.67 Model Class Reference

```
#include <Model.h>
```

Collaboration diagram for Model:



Public Member Functions

- [Model \(Simulator *simulator\)](#)
- virtual [~Model \(\)=default](#)
- [bool save \(std::string filename\)](#)
- [bool load \(std::string filename\)](#)
- [bool check \(\)](#)

Checks the integrity and consistency of the model, possibly corrects some inconsistencies, and returns if the model is in position to the simulated.

- void `clear ()`
- void `show ()`
- bool `insert (ModelElement *elemOrComp)`

Insert a new `ModelElement` or `ModelComponent` into the model (since 20191015). It's a generic access to `ComponentManager->insert()` or `ModelElement->insert()`
- void `remove (ModelElement *elemOrComp)`

Remove a new `ModelElement` or `ModelComponent` into the model (since 20191015). It's a generic access to `ComponentManager->remove()` or `ModelElement->remove()`
- void `removeEntity (Entity *entity, bool collectStatistics)`
- void `sendEntityToComponent (Entity *entity, Connection *connection, double timeDelay)`

Used by components (`ModelComponent`) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event (`Event`).
- void `sendEntityToComponent (Entity *entity, ModelComponent *component, double timeDelay, unsigned int componentInputNumber=0)`

Used by components (`ModelComponent`) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event (`Event`).
- double `parseExpression (const std::string expression)`
- double `parseExpression (const std::string expression, bool *success, std::string *errorMessage)`
- bool `checkExpression (const std::string expression, const std::string expressionName, std::string *errorMessage)`
- `Util::identification id () const`
- `List< SimulationControl * > * controls () const`

Returns a list of values that can be externally controlled (changed). They usually correspond to input parameters in the simulation model that must be changed for an experimental design.
- `List< SimulationResponse * > * responses () const`

Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.
- `OnEventManager * onEvents () const`
- `ElementManager * elements () const`

Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).
- `ComponentManager * components () const`

The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components (`SourceComponentModel`).
- `ModellInfo * infos () const`
- `Simulator * parentSimulator () const`
- `ModelSimulation * simulation () const`

Provides access to the class that manages the model simulation.
- `List< Event * > * futureEvents () const`
- void `setTraceManager (TraceManager *_traceManager)`
- `TraceManager * tracer () const`

Provides access to the class that performs the trace of simulation and replications.
- bool `hasChanged () const`

6.67.1 Detailed Description

`Model` is probably the most important class of Genesys kernel. It represents a discrete event-driven simulation model. Each model is responsible for controlling its own simulation, ie, for sequentially processing events and collecting statistical results. A model is mainly represented by a collection of components (`ModelComponent`), adequately configurated and connected, and a collection of under layered element (`ModelElement`).

Definition at line 43 of file `Model.h`.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 Model()

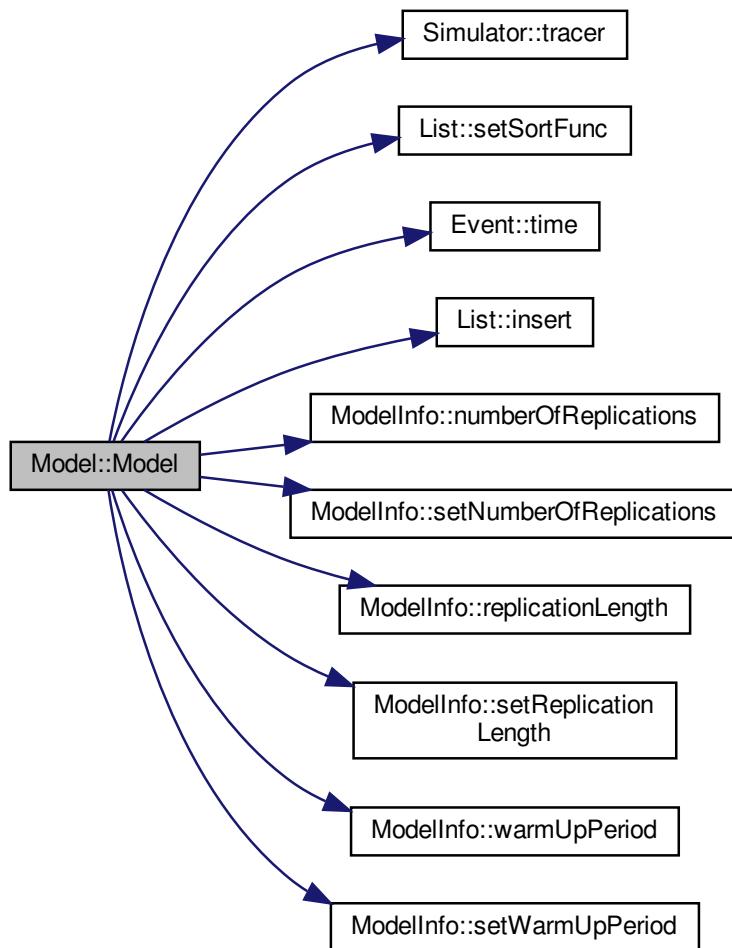
```
Model::Model ( Simulator * simulator )
```

The future events list must be chronologicaly sorted

Events are sorted chronologically

Definition at line 30 of file [Model.cpp](#).

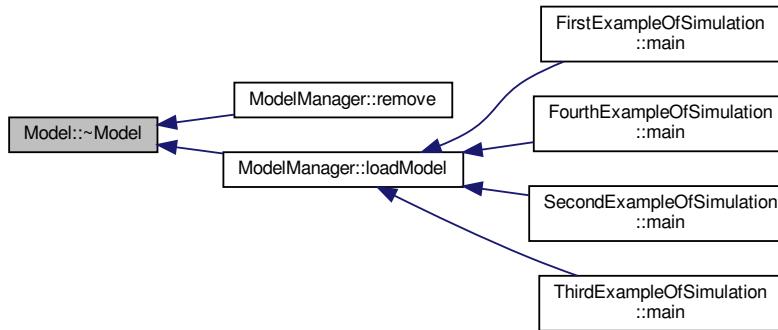
Here is the call graph for this function:



6.67.2.2 ~Model()

```
virtual Model::~Model ( ) [virtual], [default]
```

Here is the caller graph for this function:



6.67.3 Member Function Documentation

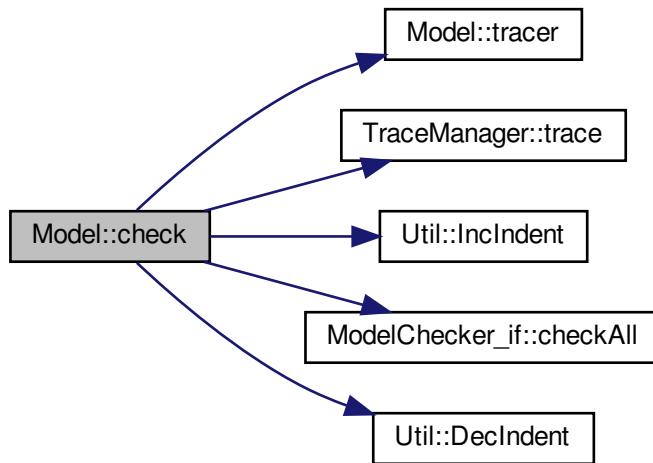
6.67.3.1 check()

```
bool Model::check ( )
```

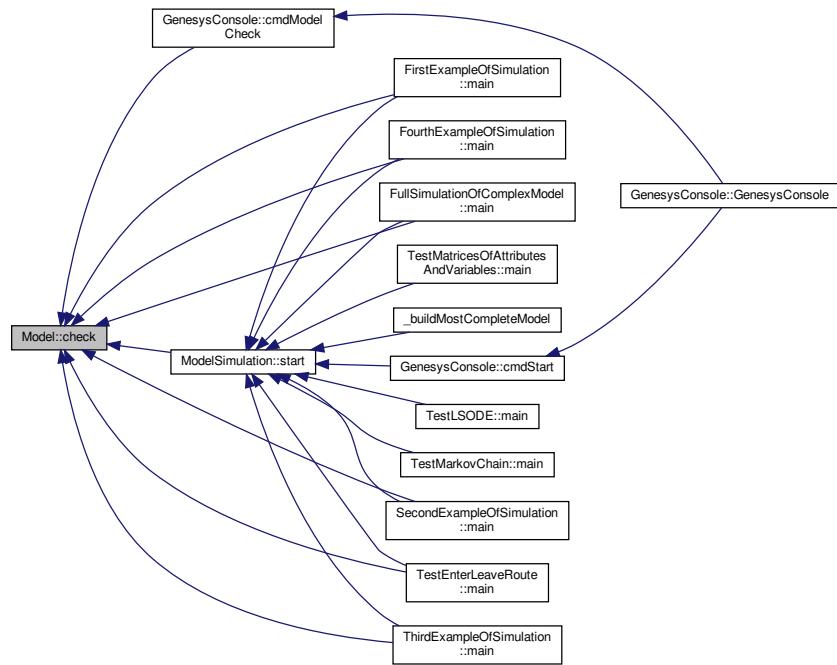
Checks the integrity and consistency of the model, possibly corrects some inconsistencies, and returns if the model is in position to the simulated.

Definition at line 235 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.67.3.2 checkExpression()

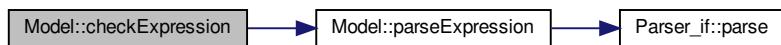
```

bool Model::checkExpression (
    const std::string expression,
    const std::string expressionName,
    std::string * errorMessage )

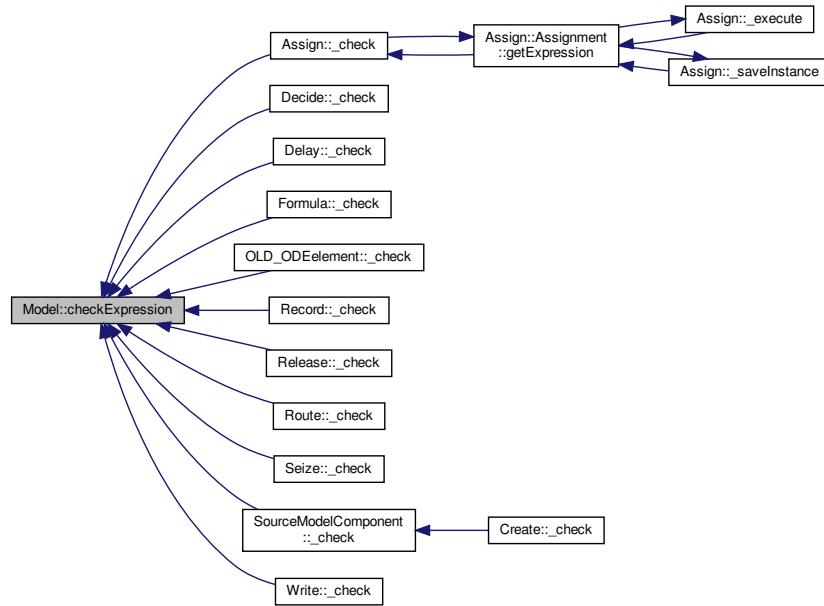
```

Definition at line 115 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

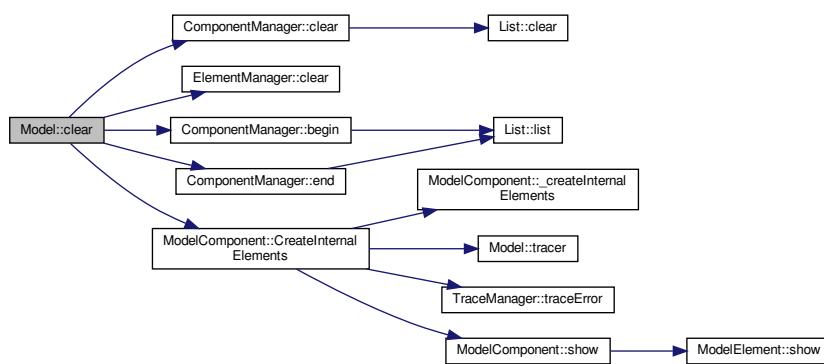


6.67.3.3 clear()

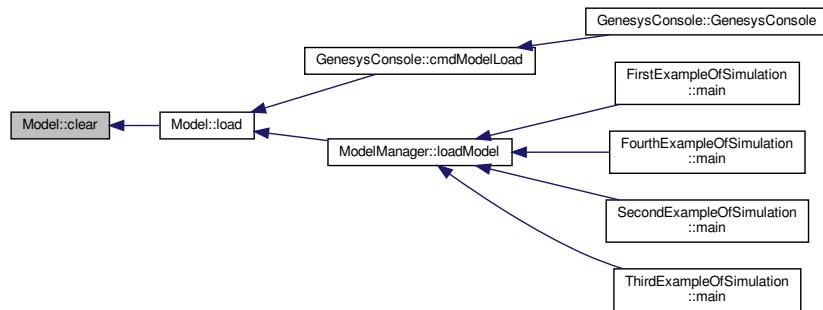
```
void Model::clear ( )
```

Definition at line 214 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



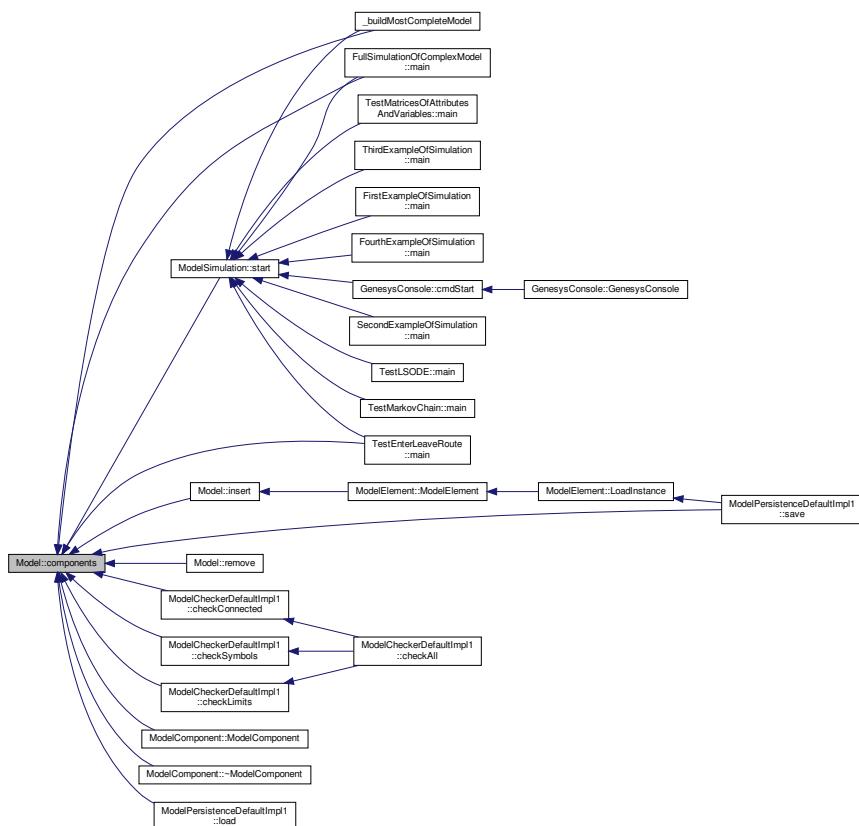
6.67.3.4 components()

```
ComponentManager * Model::components ( ) const
```

The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components (SourceComponentModel).

Definition at line 284 of file [Model.cpp](#).

Here is the caller graph for this function:



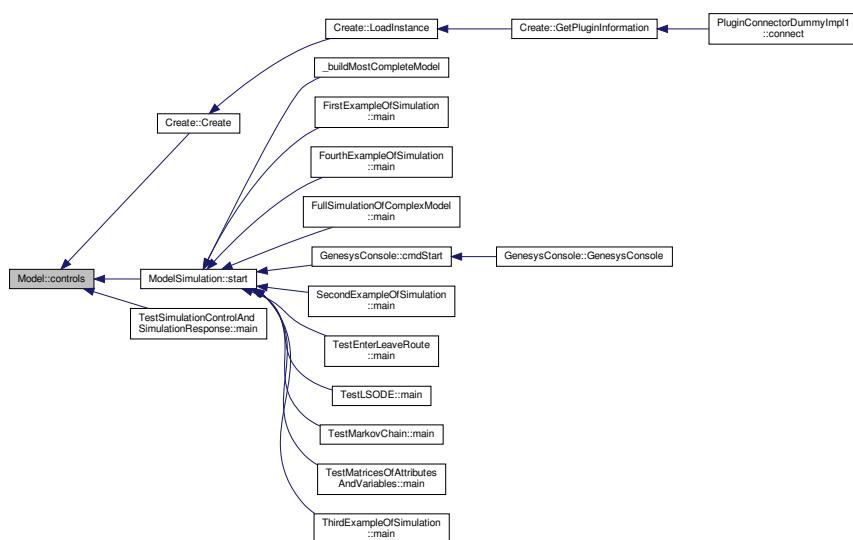
6.67.3.5 controls()

```
List< SimulationControl * > * Model::controls ( ) const
```

Returns a list of values that can be externally controlled (changed). They usually correspond to input parameters in the simulation model that must be changed for an experimental design.

Definition at line 288 of file [Model.cpp](#).

Here is the caller graph for this function:



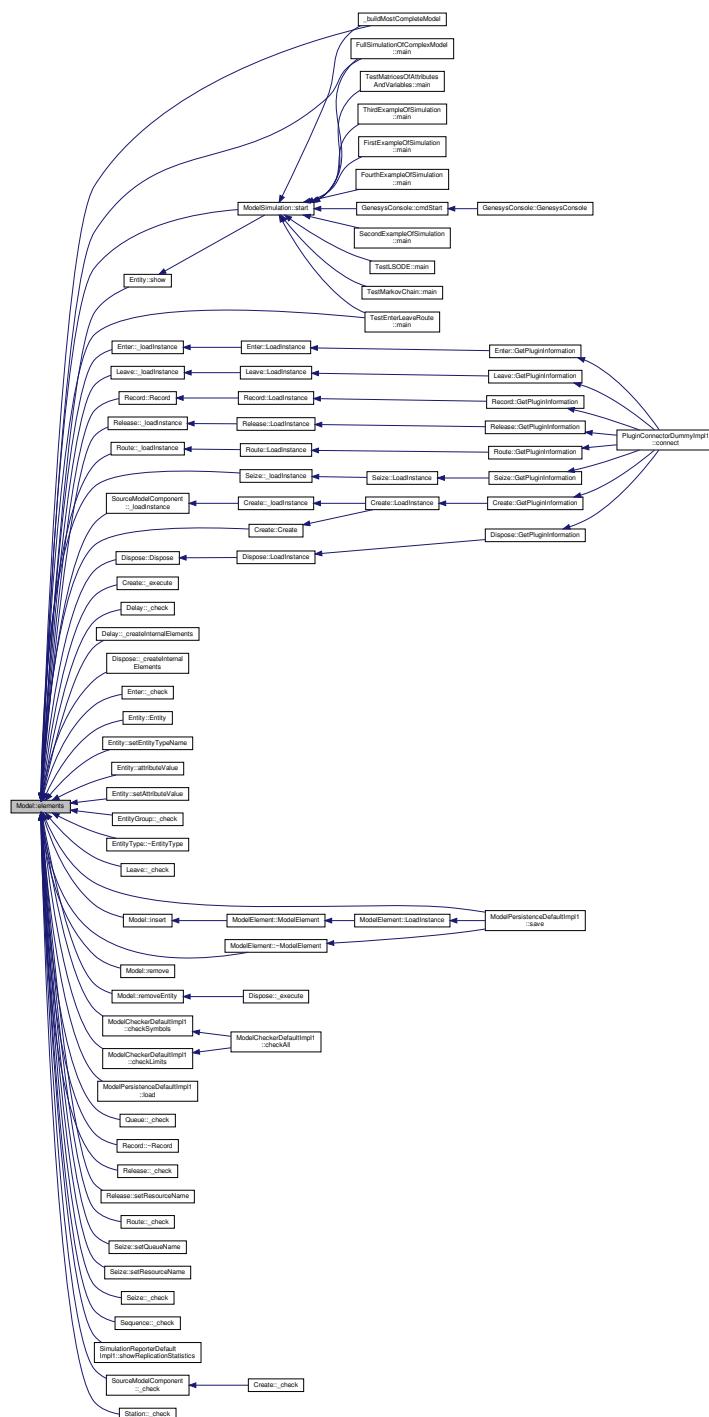
6.67.3.6 elements()

```
ElementManager * Model::elements ( ) const
```

Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).

Definition at line 300 of file [Model.cpp](#).

Here is the caller graph for this function:

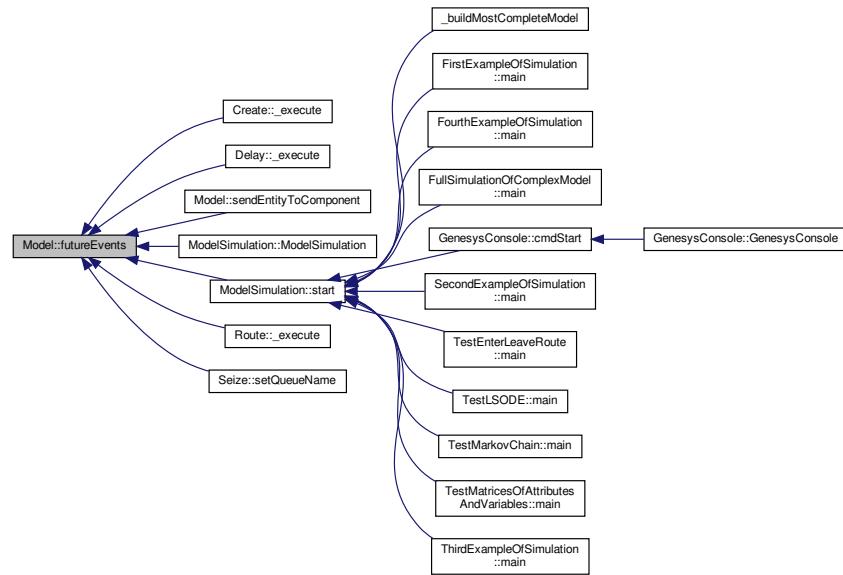


6.67.3.7 futureEvents()

```
List< Event * > * Model::futureEvents ( ) const
```

Definition at line 263 of file [Model.cpp](#).

Here is the caller graph for this function:

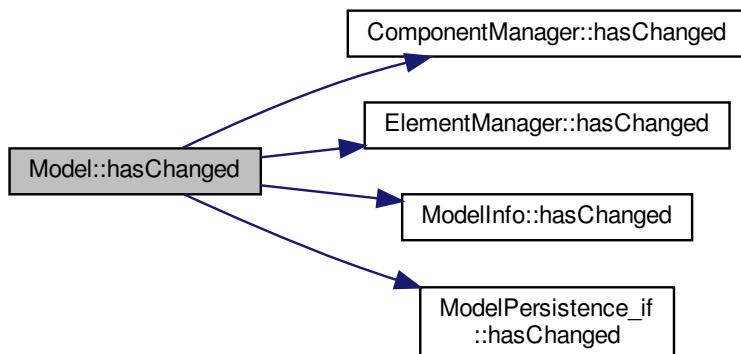


6.67.3.8 hasChanged()

```
bool Model::hasChanged ( ) const
```

Definition at line 275 of file [Model.cpp](#).

Here is the call graph for this function:



6.67.3.9 id()

```
Util::identification Model::id () const
```

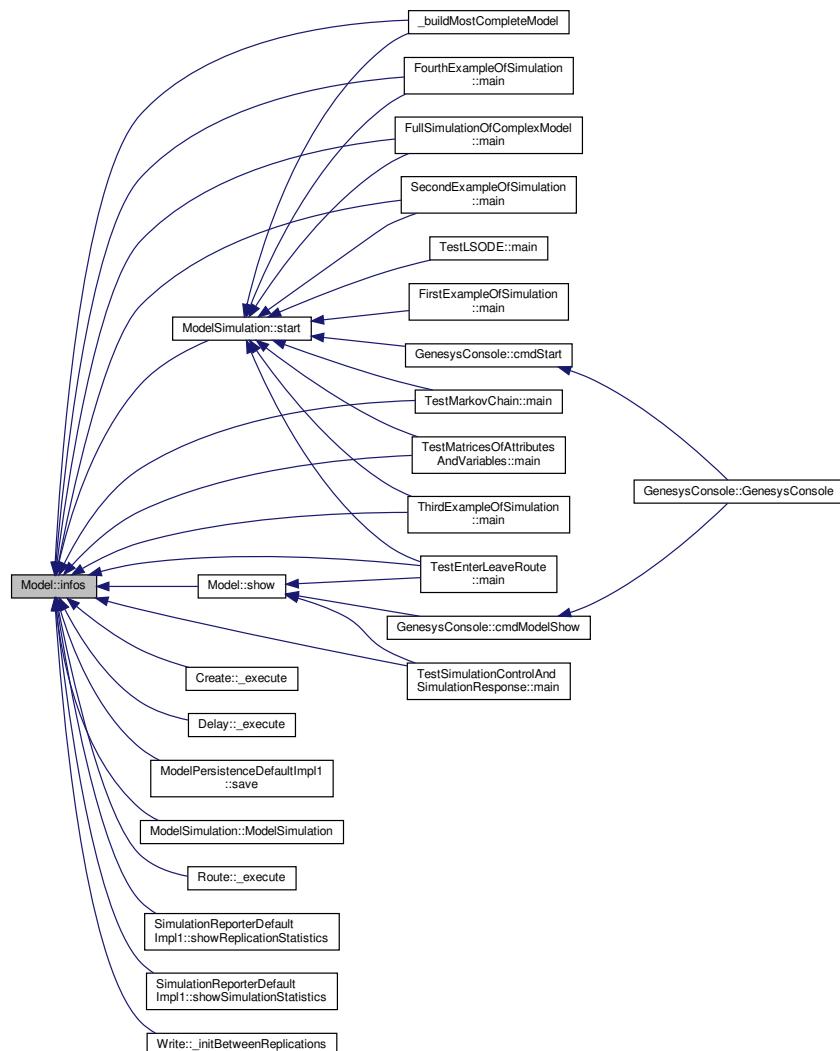
Definition at line 316 of file [Model.cpp](#).

6.67.3.10 infos()

```
ModelInfo * Model::infos () const
```

Definition at line 304 of file [Model.cpp](#).

Here is the caller graph for this function:



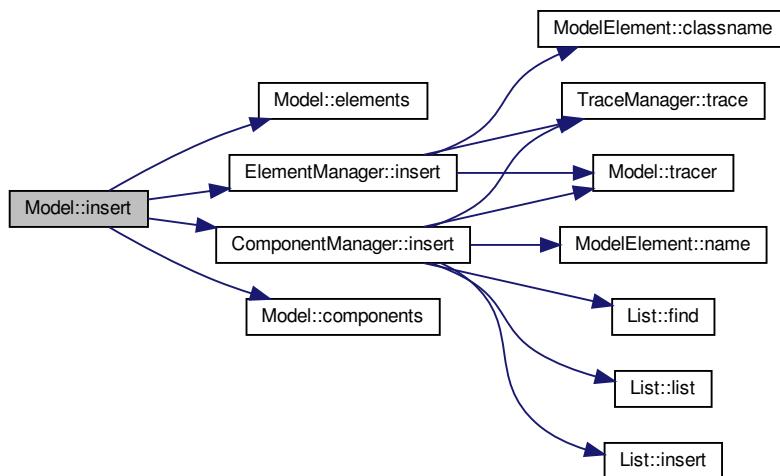
6.67.3.11 insert()

```
bool Model::insert (
    ModelElement * elemOrComp )
```

Insert a new [ModelElement](#) or [ModelComponent](#) into the model (since 20191015). It's a generic access to ComponentManager->[insert\(\)](#) or ModelElemento->[insert\(\)](#)

Definition at line 147 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

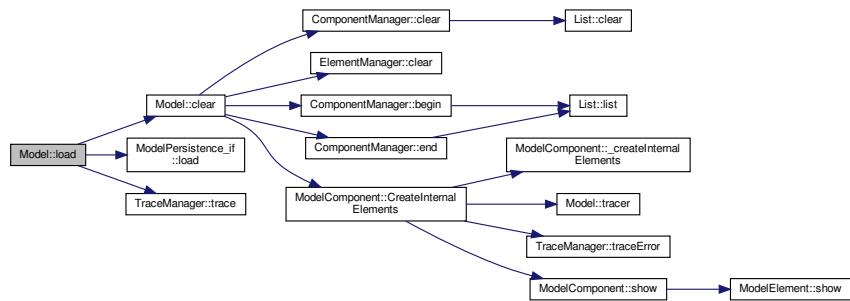


6.67.3.12 load()

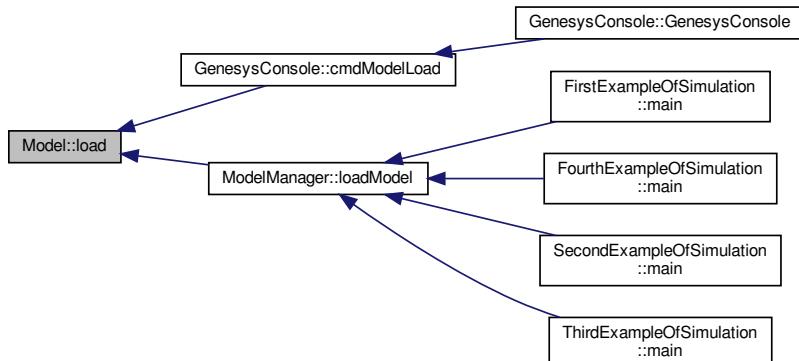
```
bool Model::load (
    std::string filename )
```

Definition at line 97 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

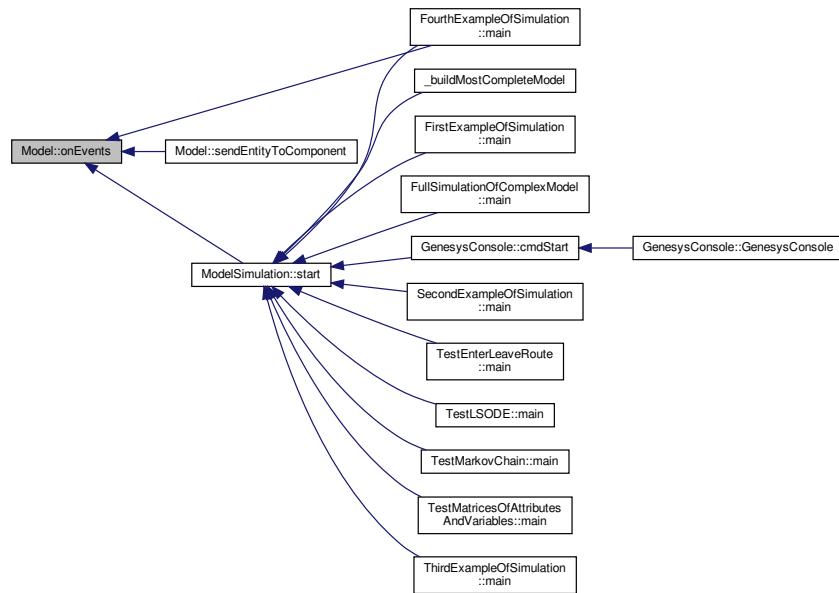


6.67.3.13 onEvents()

```
OnEventManager * Model::onEvents( ) const
```

Definition at line 296 of file [Model.cpp](#).

Here is the caller graph for this function:

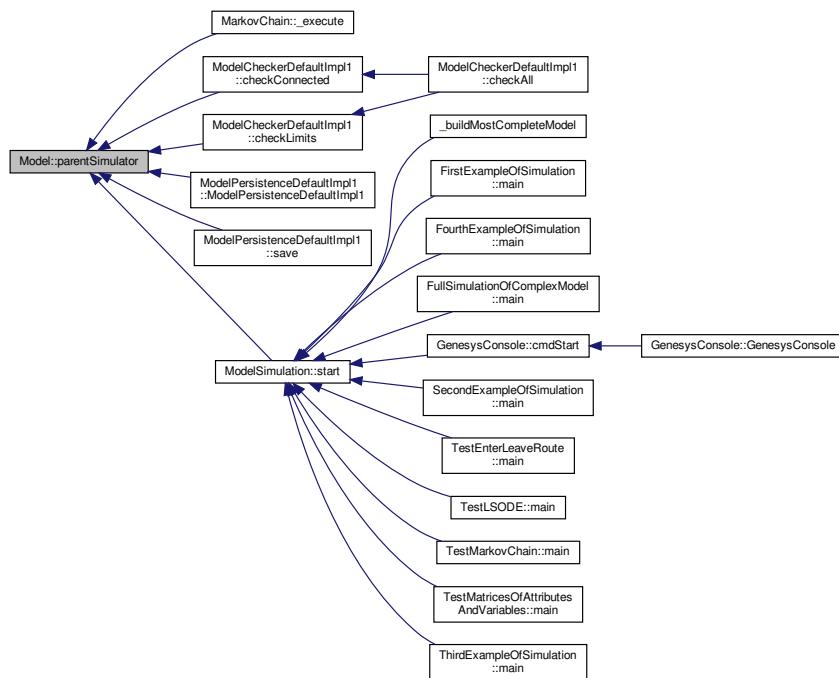


6.67.3.14 parentSimulator()

`Simulator * Model::parentSimulator () const`

Definition at line 308 of file [Model.cpp](#).

Here is the caller graph for this function:



6.67.3.15 parseExpression() [1/2]

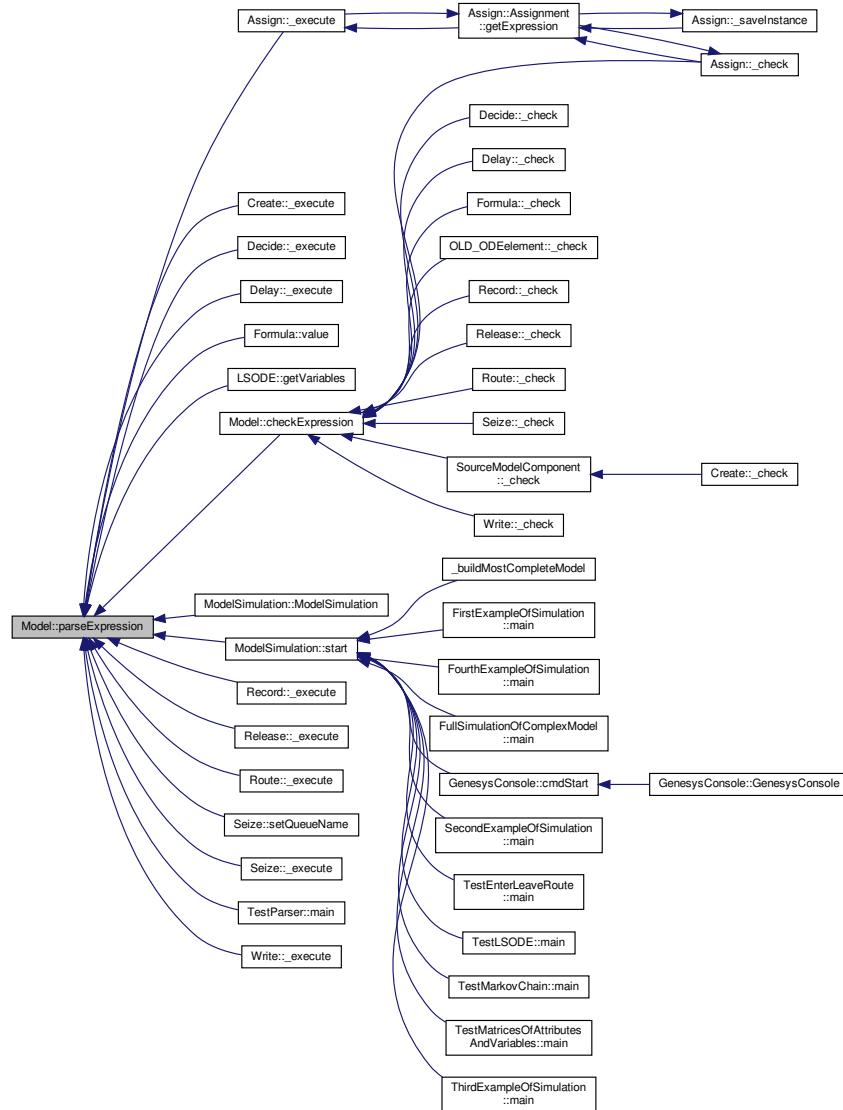
```
double Model::parseExpression (
    const std::string expression )
```

Definition at line 107 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.67.3.16 parseExpression() [2 / 2]

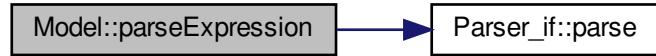
```

double Model::parseExpression (
    const std::string expression,
    bool * success,
    std::string * errorMessage )

```

Definition at line 125 of file [Model.cpp](#).

Here is the call graph for this function:



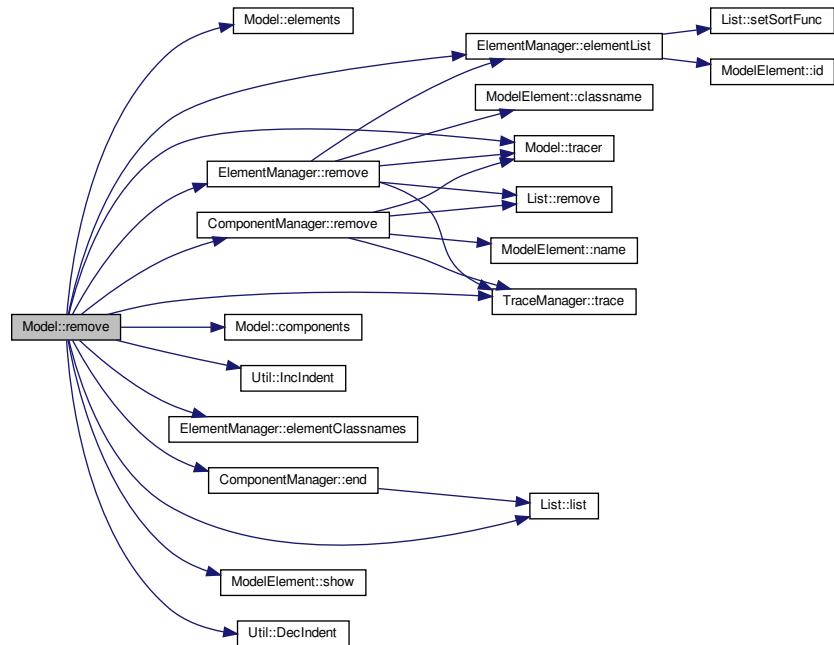
6.67.3.17 remove()

```
void Model::remove (
```

Remove a new `ModelElement` or `ModelComponent` into the model (since 20191015). It's a generic access to `ComponentManager->remove()` or `ModelElement->remove()`

Definition at line 155 of file [Model.cpp](#).

Here is the call graph for this function:



6.67.3.18 removeEntity()

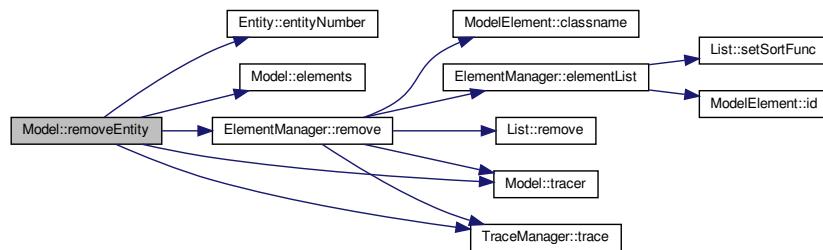
```
void Model::removeEntity (
```

`Entity * entity,`

```
        bool collectStatistics )
```

Definition at line 256 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



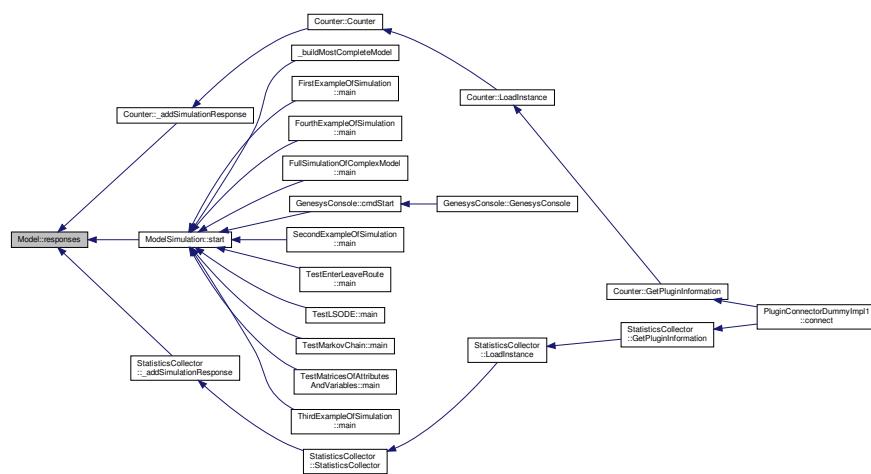
6.67.3.19 responses()

```
List< SimulationResponse * > * Model::responses ( ) const
```

Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.

Definition at line 292 of file [Model.cpp](#).

Here is the caller graph for this function:

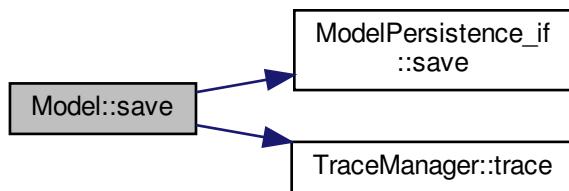


6.67.3.20 save()

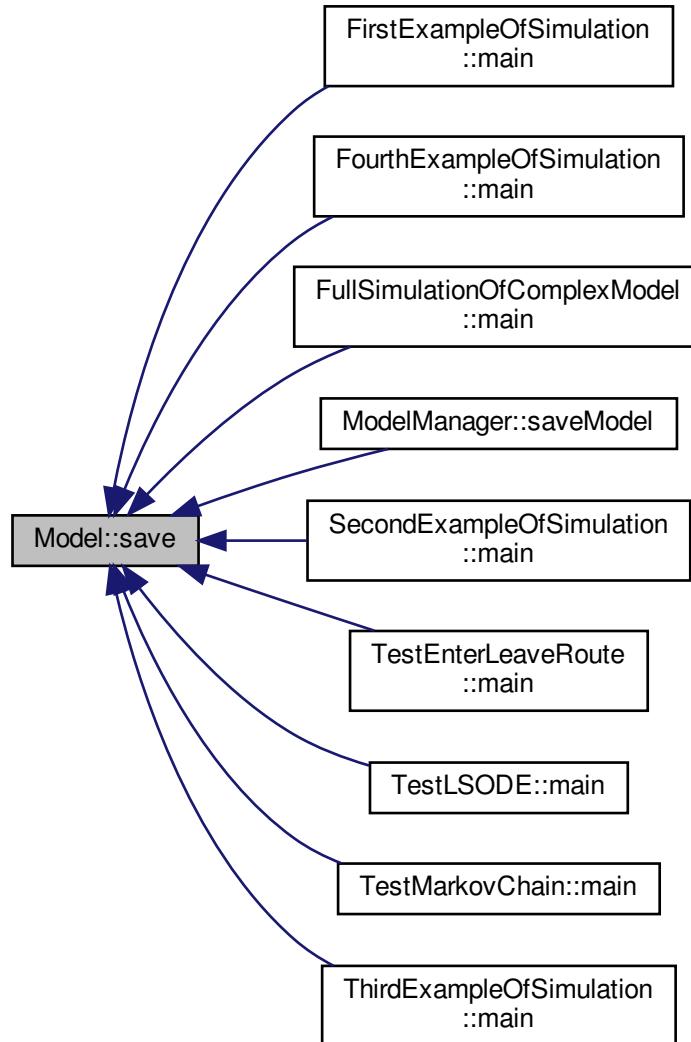
```
bool Model::save (
    std::string filename )
```

Definition at line 86 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



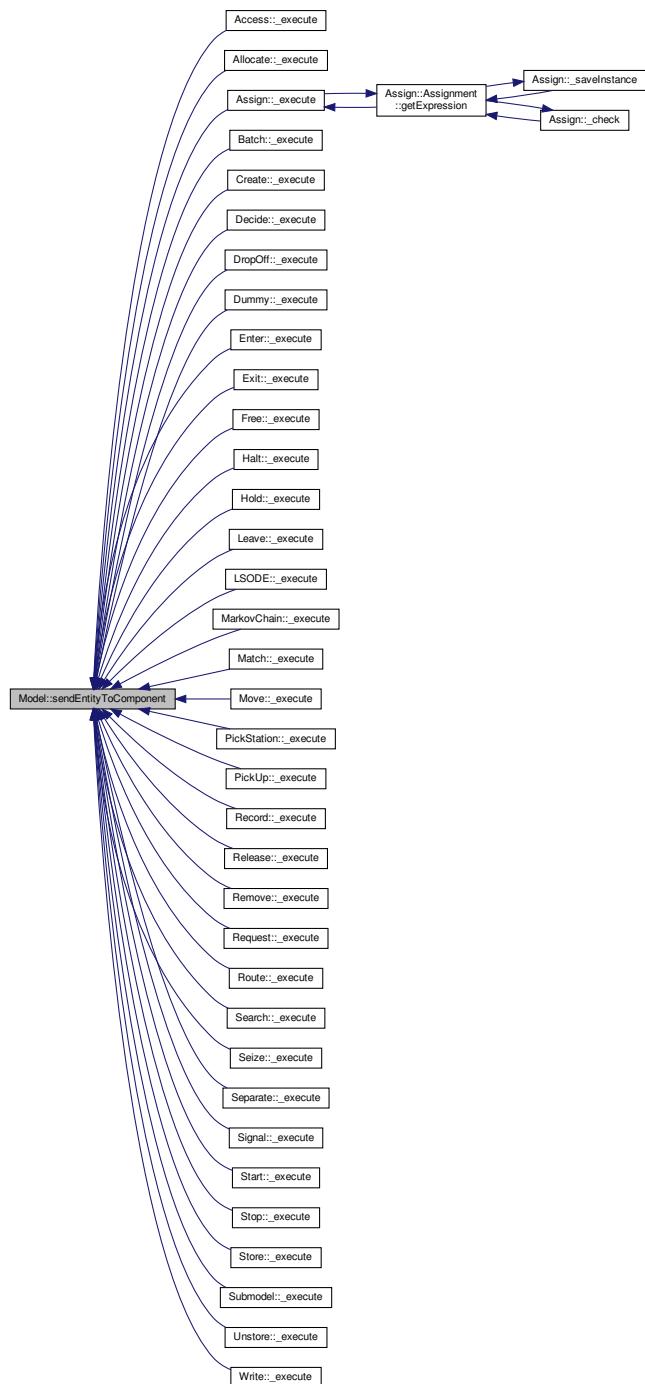
6.67.3.21 sendEntityToComponent() [1/2]

```
void Model::sendEntityToComponent (
    Entity * entity,
    Connection * connection,
    double timeDelay )
```

Used by components ([ModelComponent](#)) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event ([Event](#)).

Definition at line 67 of file [Model.cpp](#).

Here is the caller graph for this function:



6.67.3.22 sendEntityToComponent() [2 / 2]

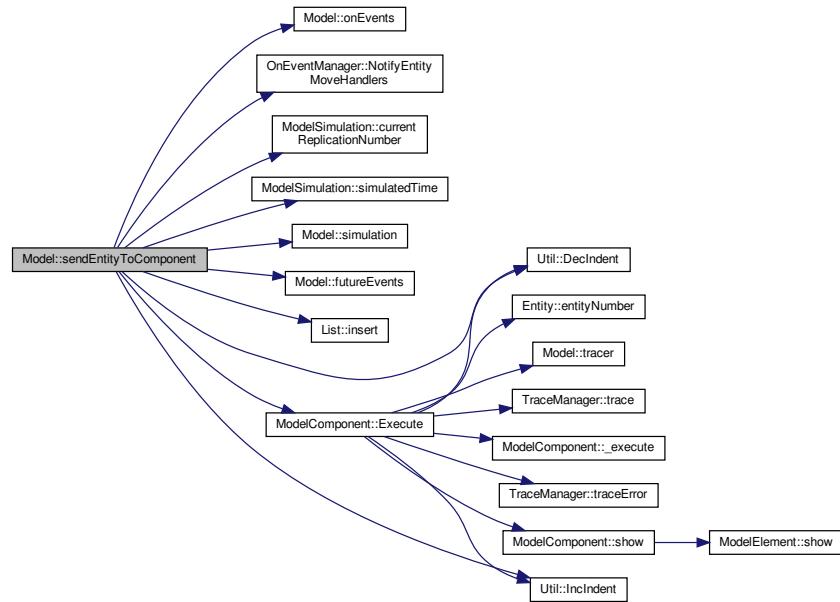
```
void Model::sendEntityToComponent (
    Entity * entity,
    ModelComponent * component,
```

```
double timeDelay,
unsigned int componentInputNumber = 0 )
```

Used by components ([ModelComponent](#)) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event ([Event](#)).

Definition at line 71 of file [Model.cpp](#).

Here is the call graph for this function:



6.67.3.23 setTraceManager()

```
void Model::setTraceManager (
    TraceManager * _traceManager )
```

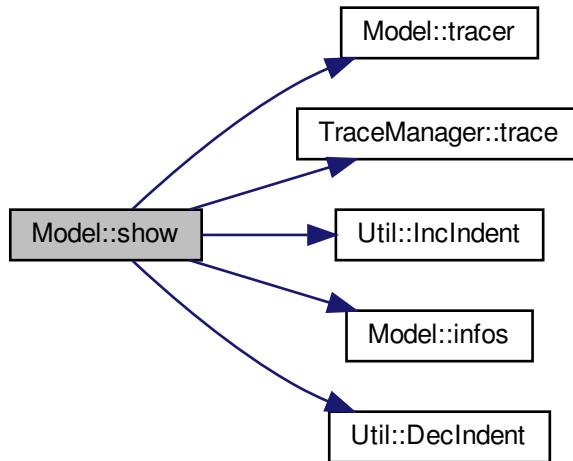
Definition at line 267 of file [Model.cpp](#).

6.67.3.24 show()

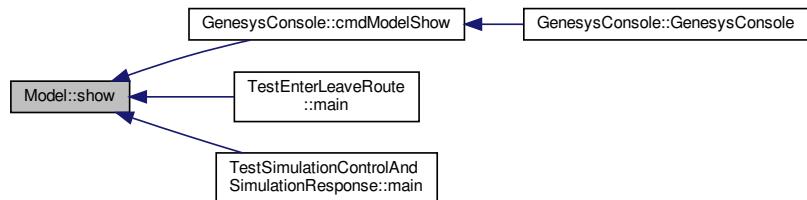
```
void Model::show ( )
```

Definition at line 130 of file [Model.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



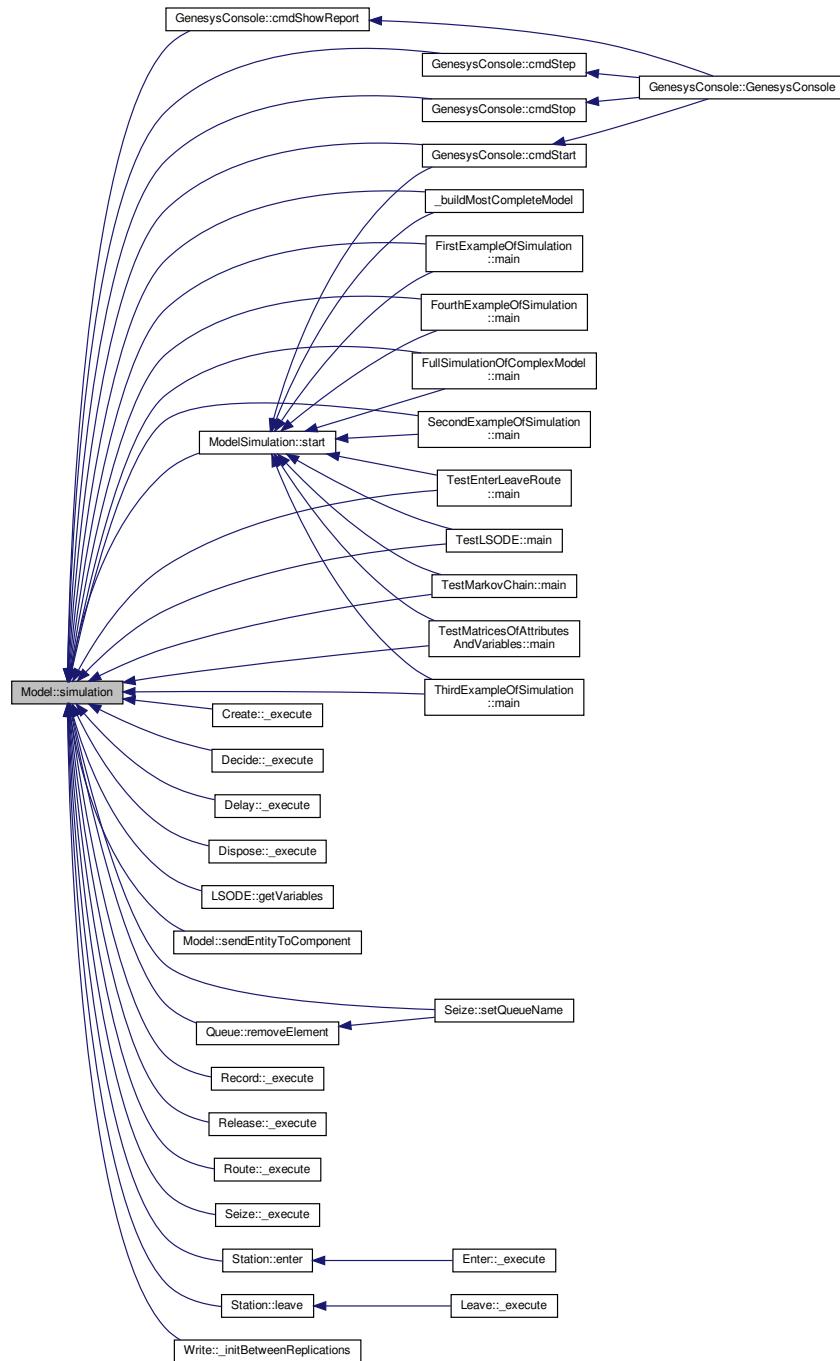
6.67.3.25 simulation()

```
ModelSimulation * Model::simulation ( ) const
```

Provides access to the class that manages the model simulation.

Definition at line 312 of file [Model.cpp](#).

Here is the caller graph for this function:



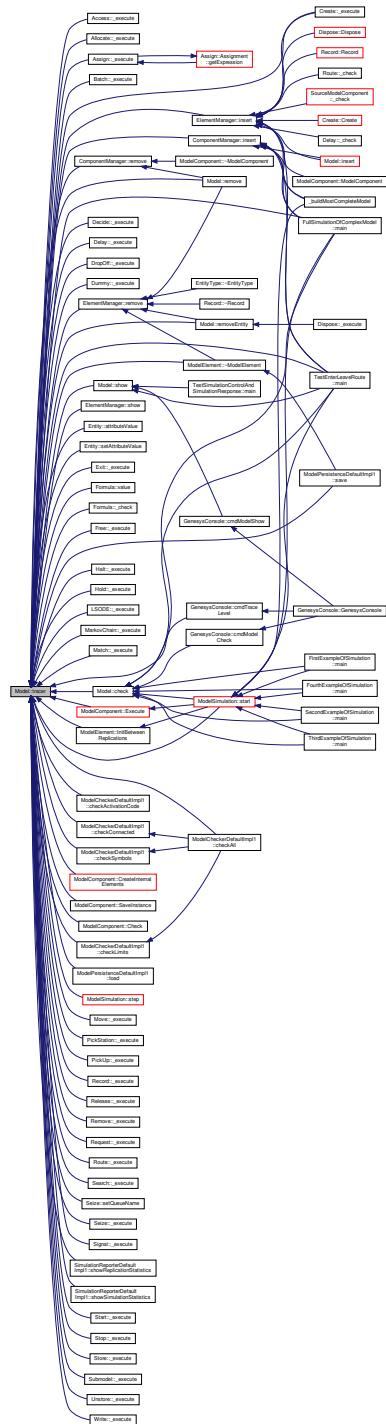
6.67.3.26 tracer()

```
TraceManager * Model::tracer ( ) const
```

Provides access to the class that performs the trace of simulation and replications.

Definition at line 271 of file [Model.cpp](#).

Here is the caller graph for this function:



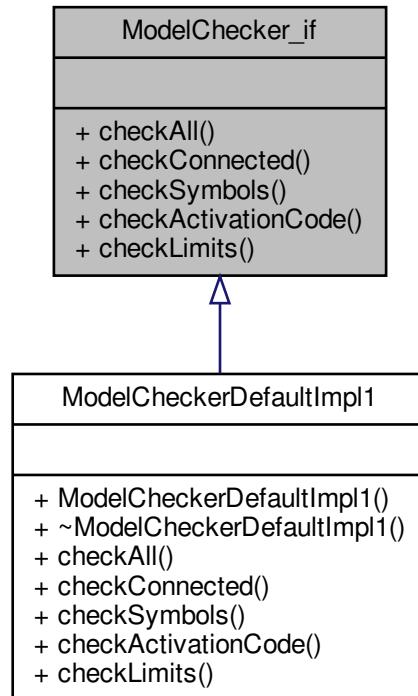
The documentation for this class was generated from the following files:

- Model.h
 - Model.cpp

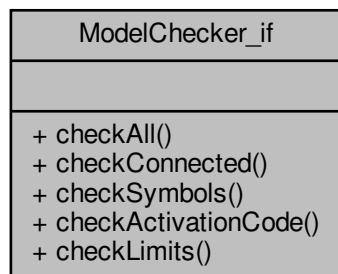
6.68 ModelChecker_if Class Reference

```
#include <ModelChecker_if.h>
```

Inheritance diagram for ModelChecker_if:



Collaboration diagram for ModelChecker_if:



Public Member Functions

- virtual bool `checkAll ()=0`
- virtual bool `checkConnected ()=0`
- virtual bool `checkSymbols ()=0`
- virtual bool `checkActivationCode ()=0`
- virtual bool `checkLimits ()=0`

6.68.1 Detailed Description

The ModelChecker is responsible for verifying the model consistency, fixing inconsistencies whenever possible
Definition at line 26 of file [ModelChecker_if.h](#).

6.68.2 Member Function Documentation

6.68.2.1 checkActivationCode()

```
virtual bool ModelChecker_if::checkActivationCode ( ) [pure virtual]
```

Checks if user-defined strings for symbols required by components, usually expressions or functions, are valid or references existing and valid elements.

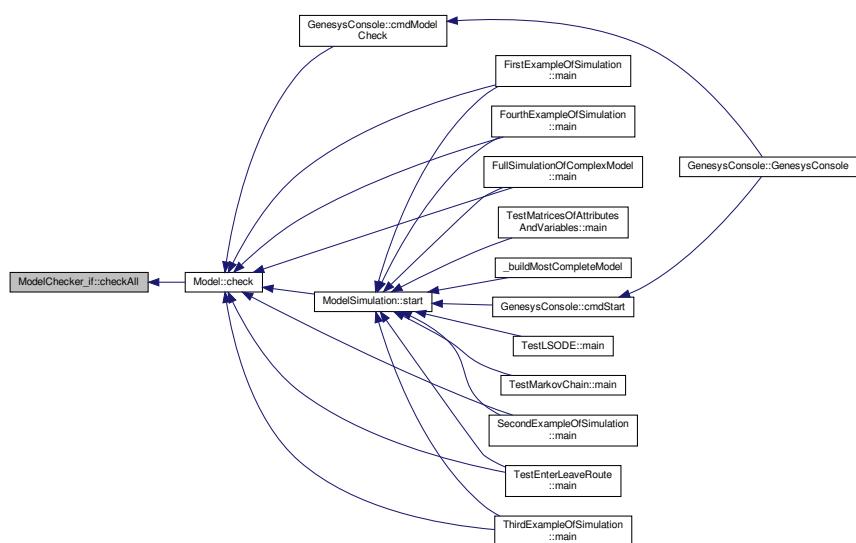
Implemented in [ModelCheckerDefaultImpl1](#).

6.68.2.2 checkAll()

```
virtual bool ModelChecker_if::checkAll ( ) [pure virtual]
```

Implemented in [ModelCheckerDefaultImpl1](#).

Here is the caller graph for this function:



6.68.2.3 checkConnected()

```
virtual bool ModelChecker_if::checkConnected () [pure virtual]
```

Invokes all other checks and returns true only if all of them returned true

Implemented in [ModelCheckerDefaultImpl1](#).

6.68.2.4 checkLimits()

```
virtual bool ModelChecker_if::checkLimits () [pure virtual]
```

Checks if the installed version has acquired a valid activation code for commercial use

Implemented in [ModelCheckerDefaultImpl1](#).

6.68.2.5 checkSymbols()

```
virtual bool ModelChecker_if::checkSymbols () [pure virtual]
```

Checks if components are consistently connected to other to form a valid process-oriented model, describing how entities proceed to the flow

Implemented in [ModelCheckerDefaultImpl1](#).

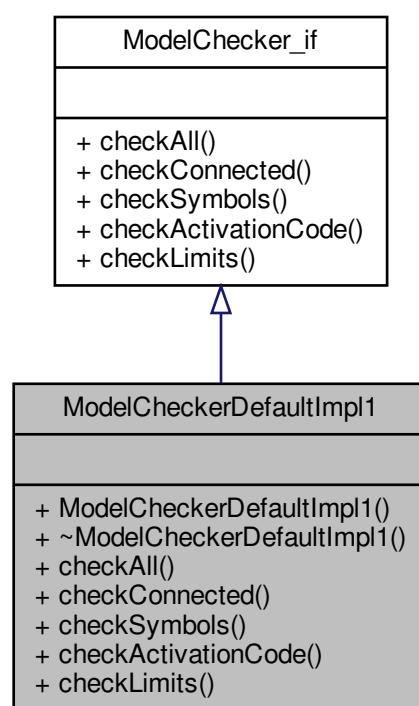
The documentation for this class was generated from the following file:

- [ModelChecker_if.h](#)

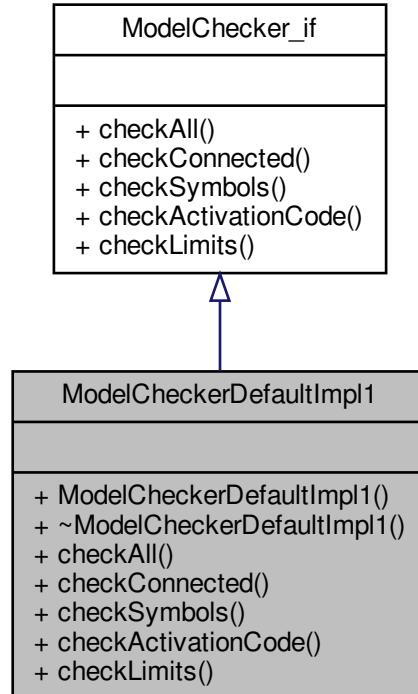
6.69 ModelCheckerDefaultImpl1 Class Reference

```
#include <ModelCheckerDefaultImpl1.h>
```

Inheritance diagram for ModelCheckerDefaultImpl1:



Collaboration diagram for ModelCheckerDefaultImpl1:



Public Member Functions

- `ModelCheckerDefaultImpl1 (Model *model)`
- `virtual ~ModelCheckerDefaultImpl1 ()=default`
- `virtual bool checkAll ()`
- `virtual bool checkConnected ()`
- `virtual bool checkSymbols ()`
- `virtual bool checkActivationCode ()`
- `virtual bool checkLimits ()`

6.69.1 Detailed Description

Definition at line 21 of file [ModelCheckerDefaultImpl1.h](#).

6.69.2 Constructor & Destructor Documentation

6.69.2.1 ModelCheckerDefaultImpl1()

```
ModelCheckerDefaultImpl1::ModelCheckerDefaultImpl1 (
    Model * model )
```

Definition at line 22 of file [ModelCheckerDefaultImpl1.cpp](#).

6.69.2.2 ~ModelCheckerDefaultImpl1()

```
virtual ModelCheckerDefaultImpl1::~ModelCheckerDefaultImpl1 ( ) [virtual], [default]
```

6.69.3 Member Function Documentation

6.69.3.1 checkActivationCode()

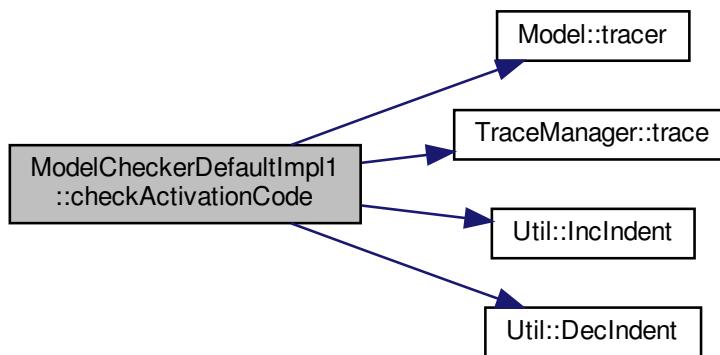
```
bool ModelCheckerDefaultImpl1::checkActivationCode ( ) [virtual]
```

Checks if user-defined strings for symbols required by components, usually expressions or functions, are valid or references existing and valid elements.

Implements [ModelChecker_if](#).

Definition at line 311 of file [ModelCheckerDefaultImpl1.cpp](#).

Here is the call graph for this function:



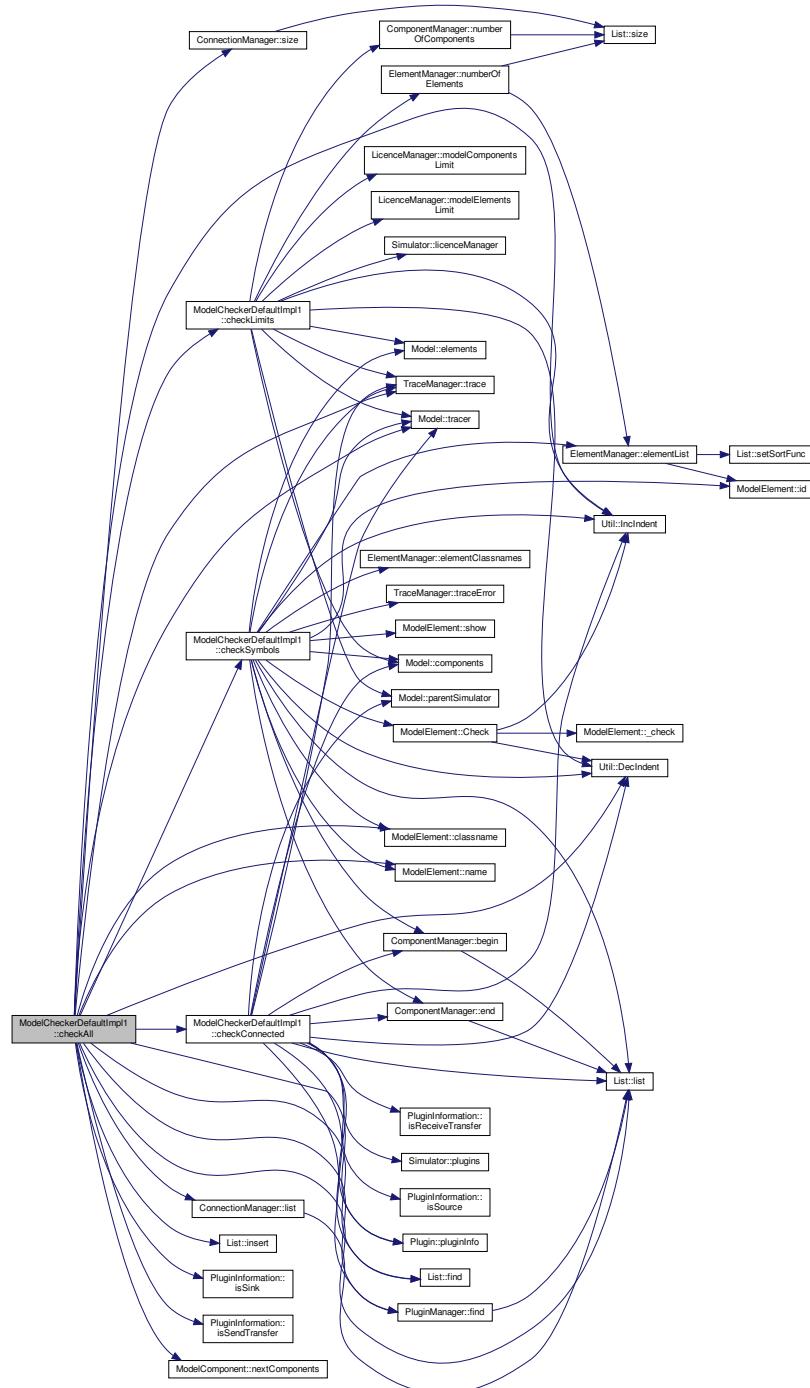
6.69.3.2 checkAll()

```
bool ModelCheckerDefaultImpl1::checkAll ( ) [virtual]
```

Implements [ModelChecker_if](#).

Definition at line 27 of file [ModelCheckerDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.69.3.3 checkConnected()

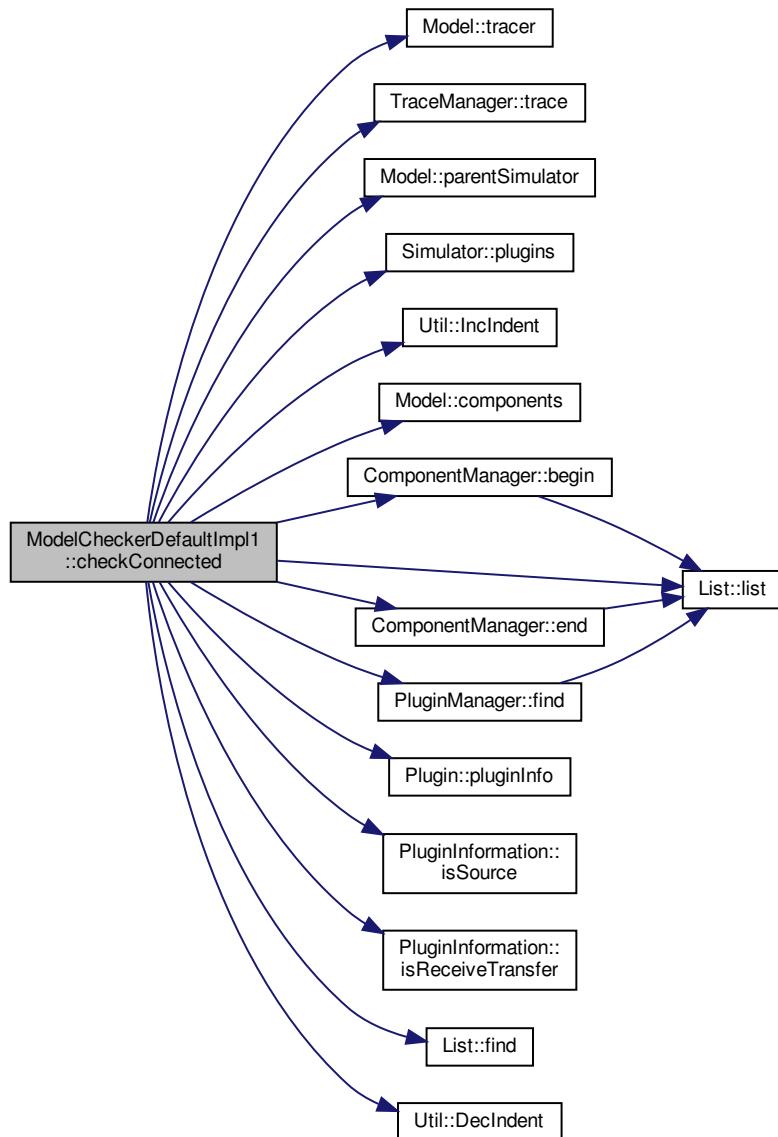
```
bool ModelCheckerDefaultImpl1::checkConnected () [virtual]
```

Invokes all other checks and returns true only if all of them returned true

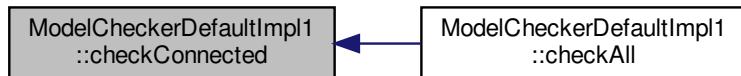
Implements [ModelChecker_if](#).

Definition at line 147 of file [ModelCheckerDefaultImpl1.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.69.3.4 checkLimits()

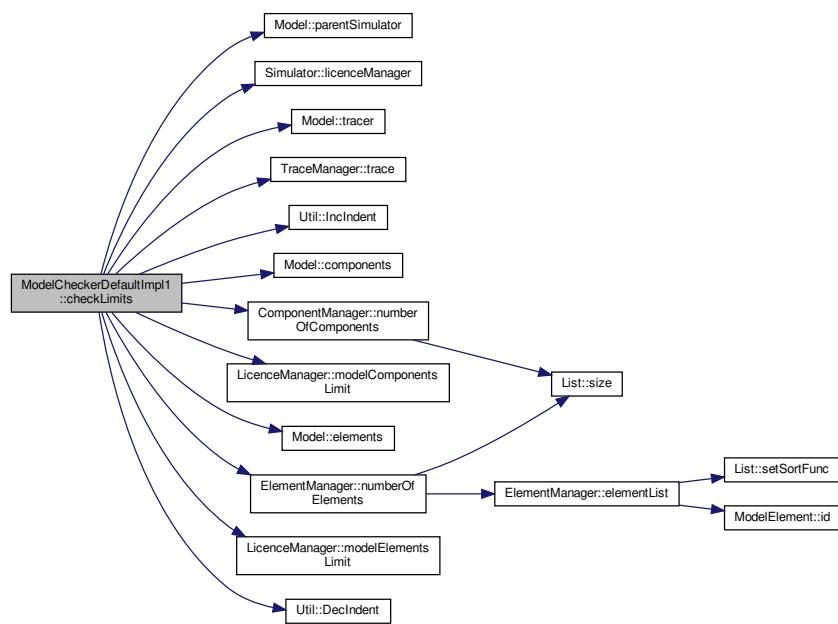
```
bool ModelCheckerDefaultImpl1::checkLimits ( ) [virtual]
```

Checks if the installed version has acquired a valid activation code for commercial use

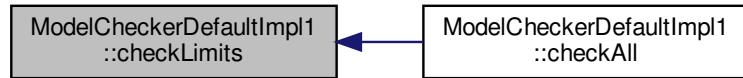
Implements [ModelChecker_if](#).

Definition at line 322 of file [ModelCheckerDefaultImpl1.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.69.3.5 checkSymbols()

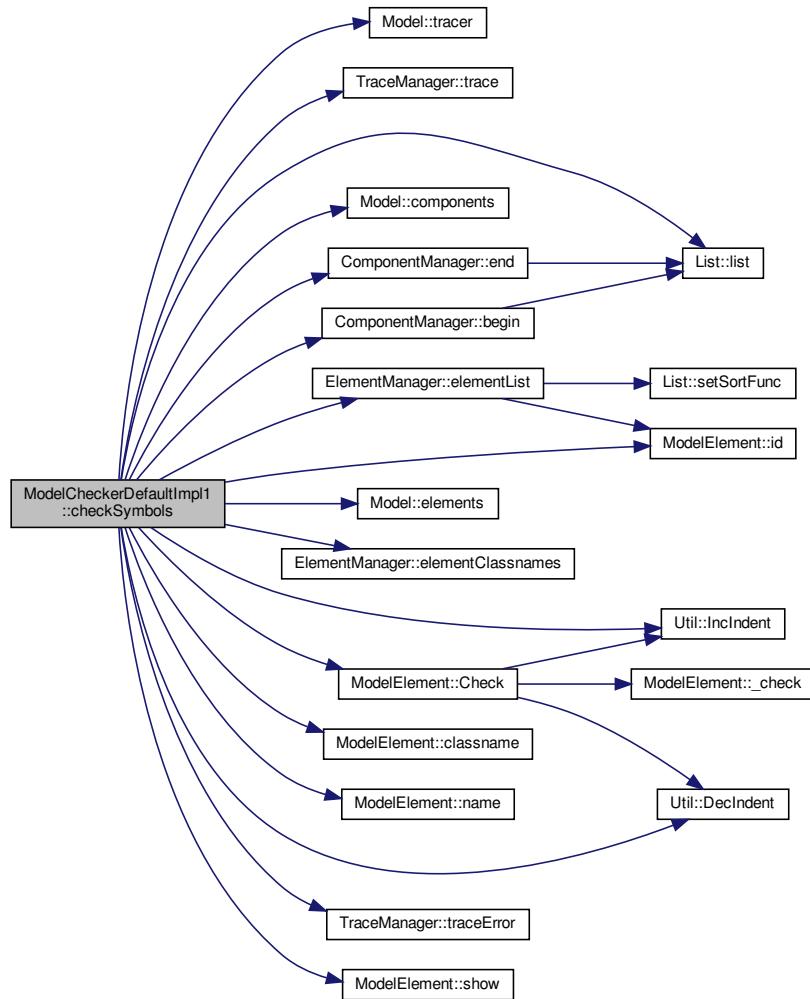
```
bool ModelCheckerDefaultImpl1::checkSymbols ( ) [virtual]
```

Checks if components are consistently connected to other to form a valid process-oriented model, describing how entities proceed to the flow

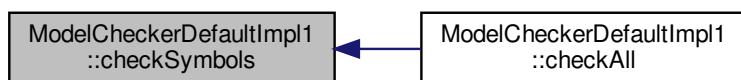
Implements [ModelChecker_if](#).

Definition at line [247](#) of file [ModelCheckerDefaultImpl1.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



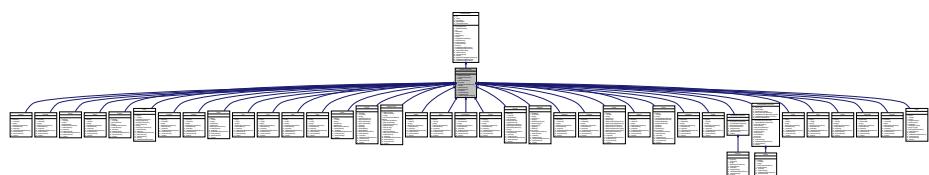
The documentation for this class was generated from the following files:

- [ModelCheckerDefaultImpl1.h](#)
- [ModelCheckerDefaultImpl1.cpp](#)

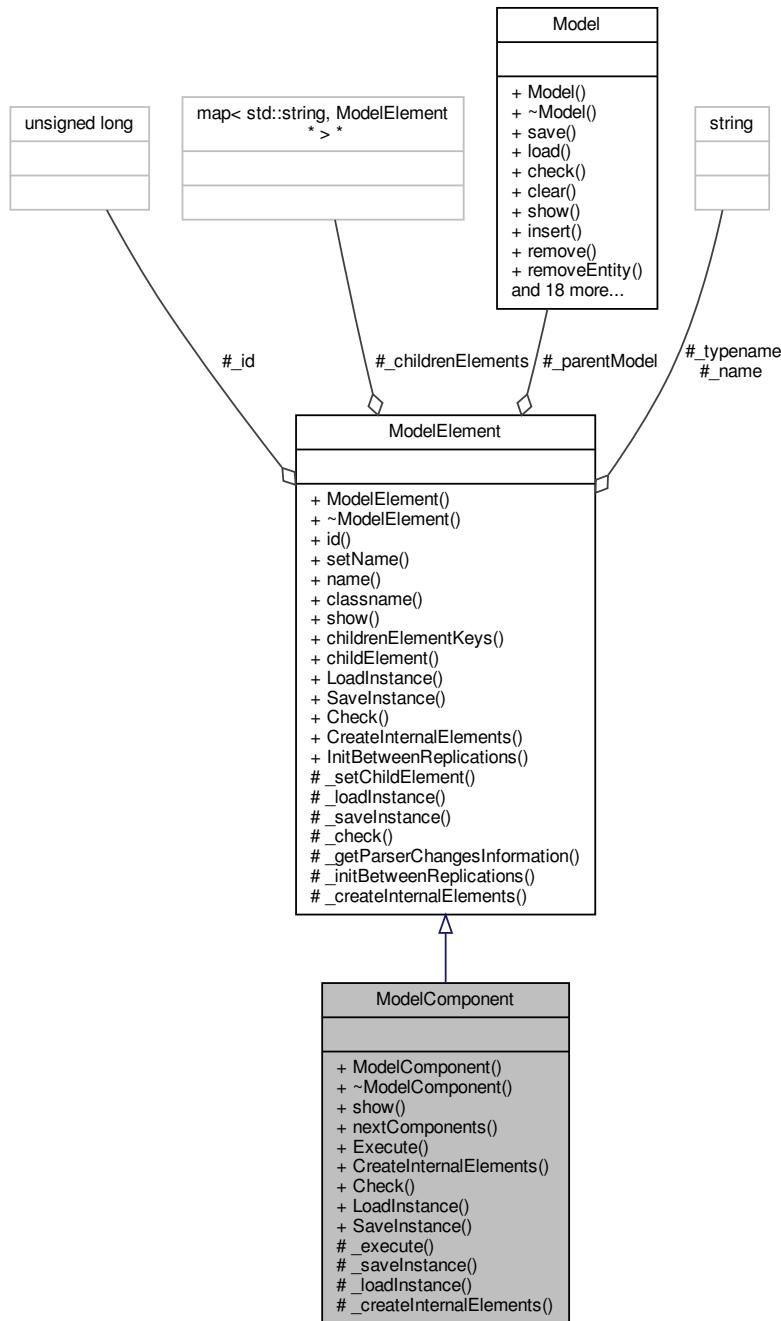
6.70 ModelComponent Class Reference

```
#include <ModelComponent.h>
```

Inheritance diagram for ModelComponent:



Collaboration diagram for ModelComponent:



Public Member Functions

- **ModelComponent** (**Model** *model, std::string componentTypename, std::string name="")
- virtual **~ModelComponent** ()
- virtual std::string **show** ()
- **ConnectionManager** * **nextComponents** () const

Returns a list of components directly connected to the output. Usually the components have a single output, but they may have none (such as [Dispose](#)) or more than one (as [Decide](#)). In addition to the component, NextComponents specifies the inputNumber of the next component where the entity will be sent to. Usually the components have a single input, but they may have none (such as [Create](#)) or more than one (as [Match](#)).

Static Public Member Functions

- static void [Execute](#) ([Entity](#) *entity, [ModelComponent](#) *component, unsigned int inputNumber)

This method triggers the simulation of the behavior of the component. It is invoked when an event (corresponding to this component) is taken from the list of future events or when an entity arrives at this component by connection.
- static void [CreateInternalElements](#) ([ModelComponent](#) *component)
- static bool [Check](#) ([ModelComponent](#) *component)
- static [ModelComponent](#) * [LoadInstance](#) ([Model](#) *model, std::map< std::string, std::string > *fields)
- static std::map< std::string, std::string > * [SaveInstance](#) ([ModelComponent](#) *component)

Protected Member Functions

- virtual void [_execute](#) ([Entity](#) *entity)=0
- virtual std::map< std::string, std::string > * [_saveinstance](#) ()
- virtual bool [_loadinstance](#) (std::map< std::string, std::string > *fields)
- virtual void [_createinternalelements](#) ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

6.70.1 Detailed Description

A component of the model is a block that represents a specific behavior to be simulated. The behavior is triggered when an entity arrives at the component, which corresponds to the occurrence of an event. A simulation model corresponds to a set of interconnected components to form the process by which the entity is submitted.

Parameters

<code>model</code>	The model this component belongs to
--------------------	-------------------------------------

Definition at line 32 of file [ModelComponent.h](#).

6.70.2 Constructor & Destructor Documentation

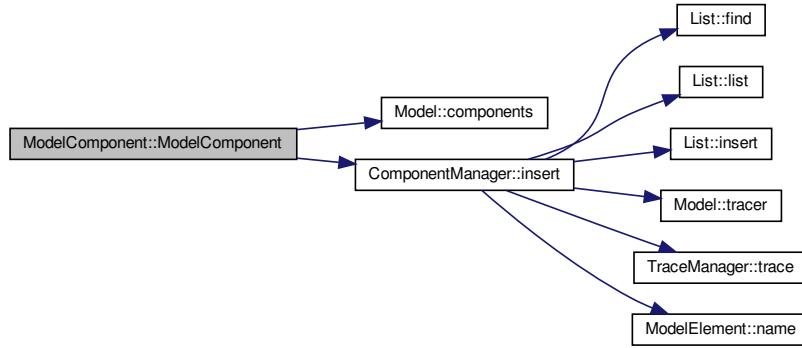
6.70.2.1 ModelComponent()

```
ModelComponent::ModelComponent (
    Model * model,
```

```
    std::string componentTypename,
    std::string name = "" )
```

Definition at line 17 of file [ModelComponent.cpp](#).

Here is the call graph for this function:

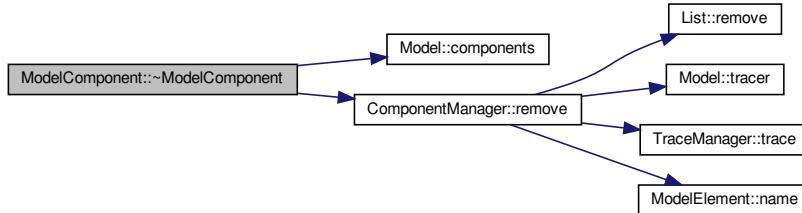


6.70.2.2 ~ModelComponent()

```
ModelComponent::~ModelComponent( ) [virtual]
```

Definition at line 21 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



6.70.3 Member Function Documentation

6.70.3.1 `_createInternalElements()`

```
void ModelComponent::_createInternalElements ( ) [protected], [virtual]
```

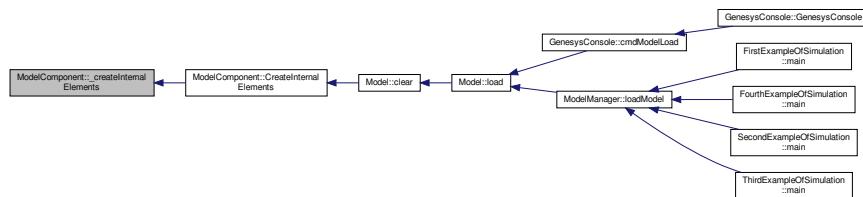
This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelElement](#).

Reimplemented in [Delay](#), and [Dispose](#).

Definition at line 125 of file [ModelComponent.cpp](#).

Here is the caller graph for this function:

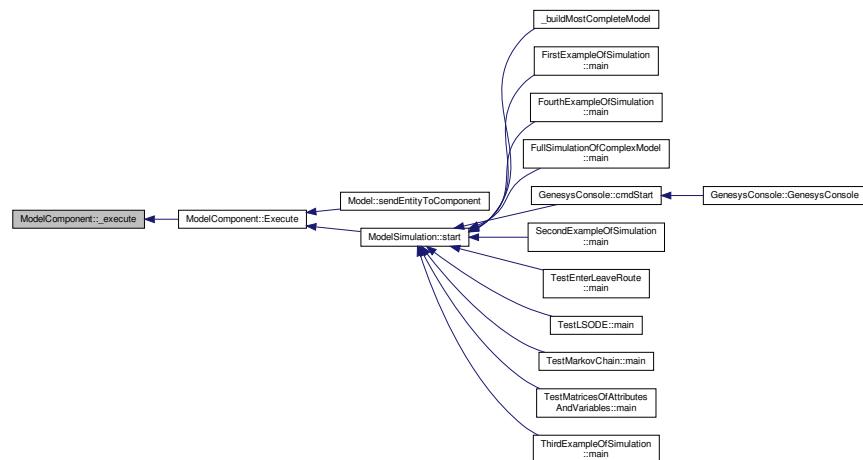


6.70.3.2 `_execute()`

```
virtual void ModelComponent::_execute (
    Entity * entity) [protected], [pure virtual]
```

Implemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Route](#), [Allocate](#), [Separate](#), [PickStation](#), [Move](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Match](#), [Delay](#), [Unstore](#), [DropOff](#), [Free](#), [Halt](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), and [Submodel](#).

Here is the caller graph for this function:



6.70.3.3 `_loadInstance()`

```
bool ModelComponent::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

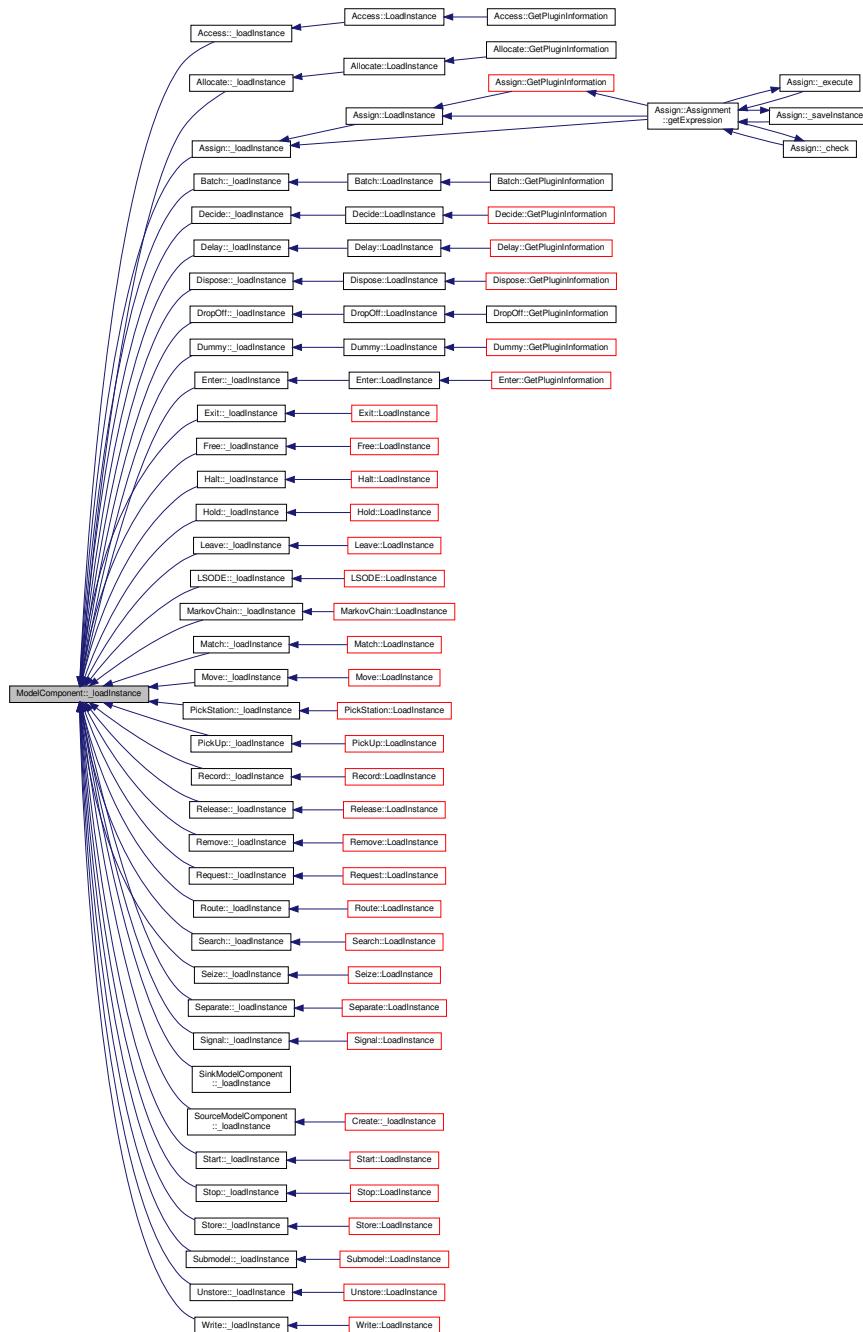
Reimplemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Route](#), [Allocate](#), [Separate](#), [PickStation](#), [Move](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Match](#), [Delay](#), [Unstore](#), [DropOff](#), [Free](#), [Halt](#), [SourceModelComponent](#), [Start](#), [PickUp](#), [Signal](#), [Dispose](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 98 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.70.3.4 `_saveInstance()`

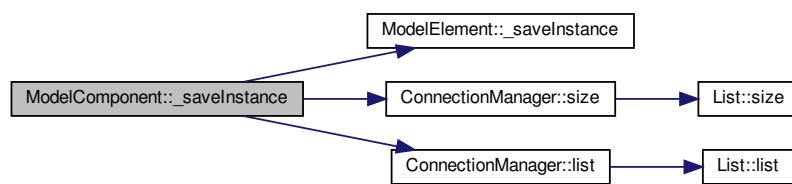
```
std::map< std::string, std::string > * ModelComponent::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

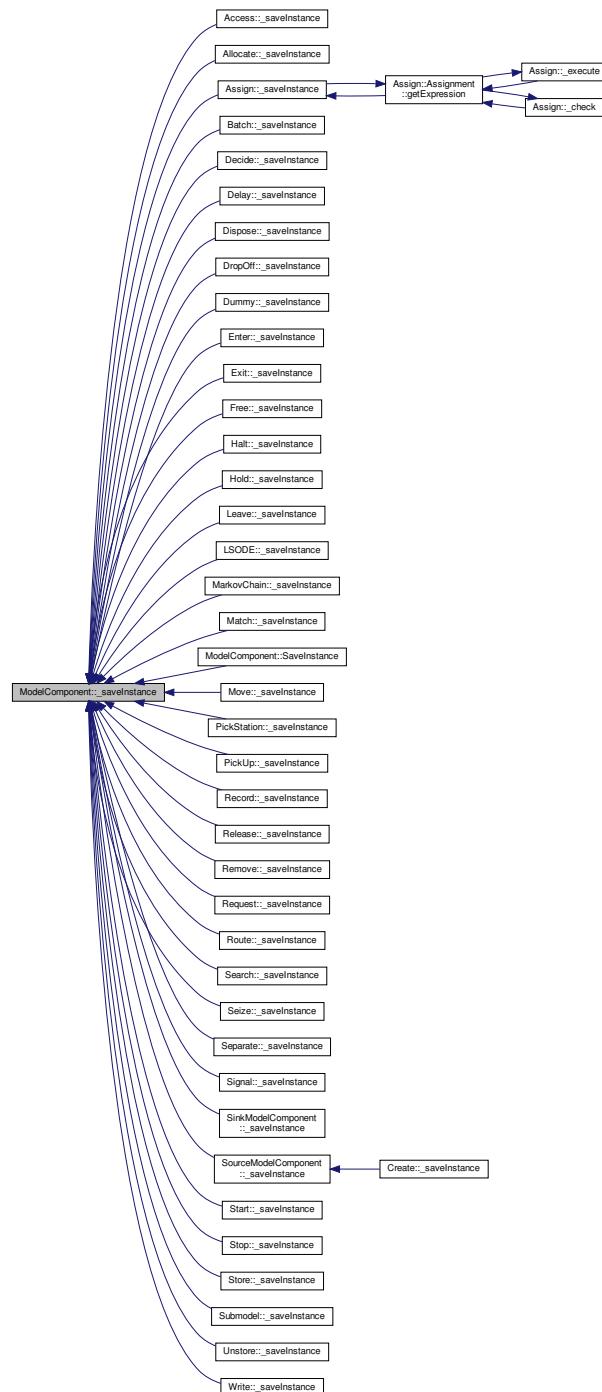
Reimplemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Route](#), [Allocate](#), [Separate](#), [PickStation](#), [Move](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Match](#), [Delay](#), [Unstore](#), [DropOff](#), [Source](#) ↔ [ModelComponent](#), [Free](#), [Halt](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 113 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

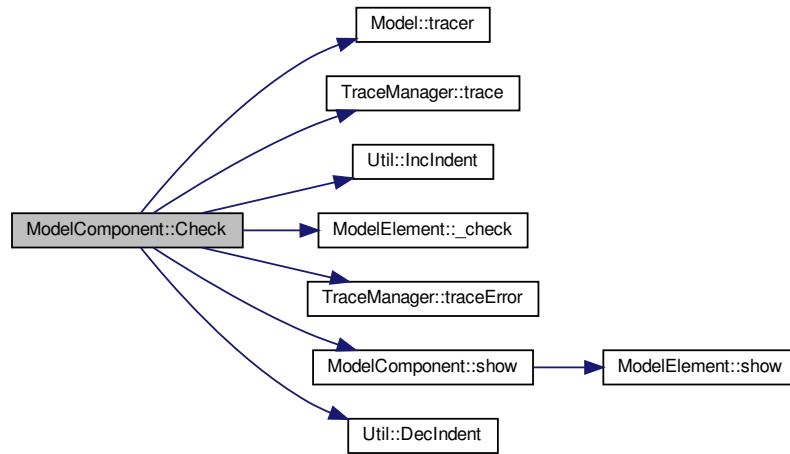


6.70.3.5 Check()

```
bool ModelComponent::Check (
    ModelComponent * component ) [static]
```

Definition at line 71 of file [ModelComponent.cpp](#).

Here is the call graph for this function:

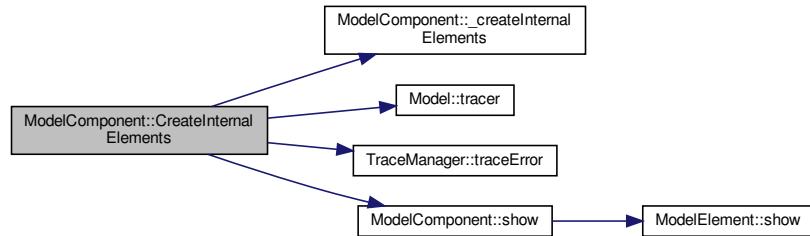


6.70.3.6 CreateInternalElements()

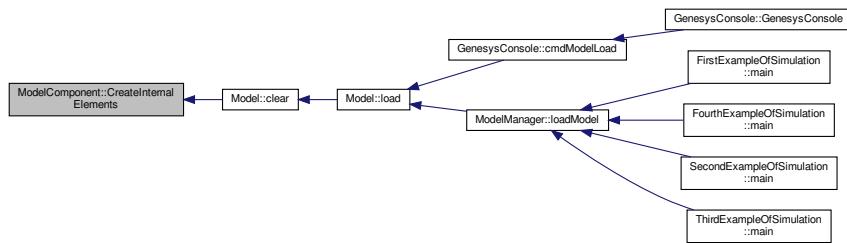
```
void ModelComponent::CreateInternalElements (
    ModelComponent * component ) [static]
```

Definition at line 51 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.70.3.7 Execute()

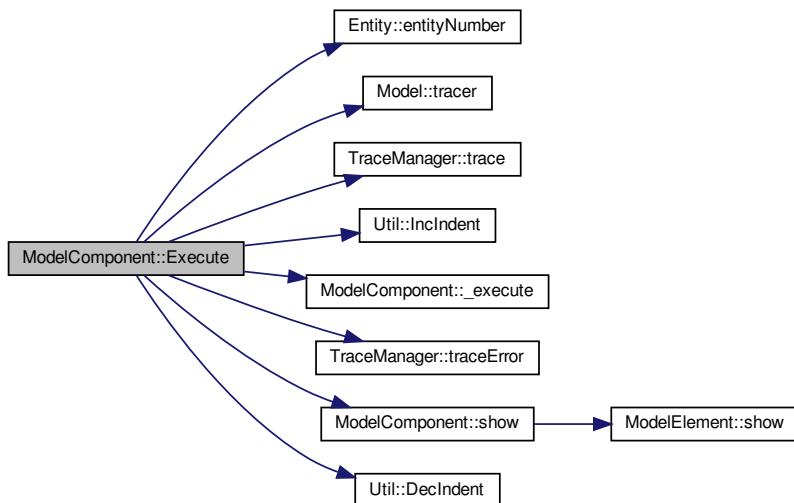
```

void ModelComponent::Execute (
    Entity * entity,
    ModelComponent * component,
    unsigned int inputNumber ) [static]
  
```

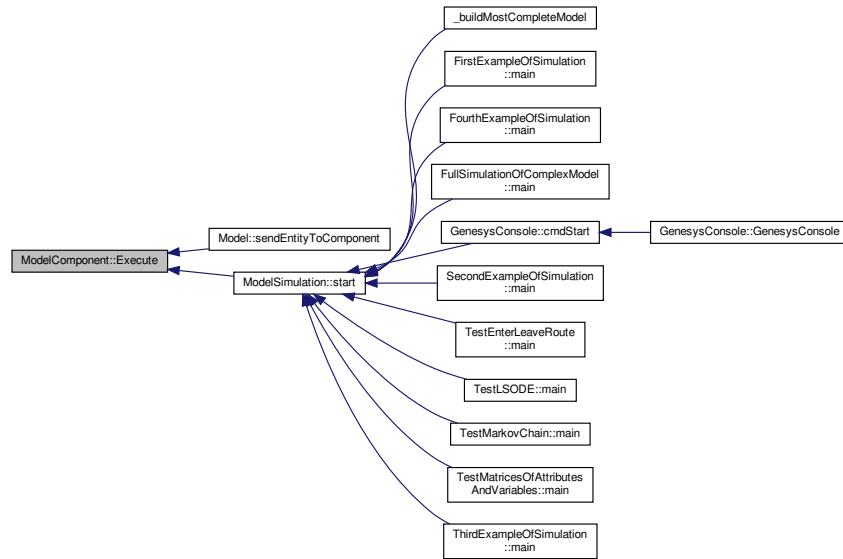
This method triggers the simulation of the behavior of the component. It is invoked when an event (corresponding to this component) is taken from the list of future events or when an entity arrives at this component by connection.

Definition at line 25 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.70.3.8 LoadInstance()

```
static ModelComponent* ModelComponent::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

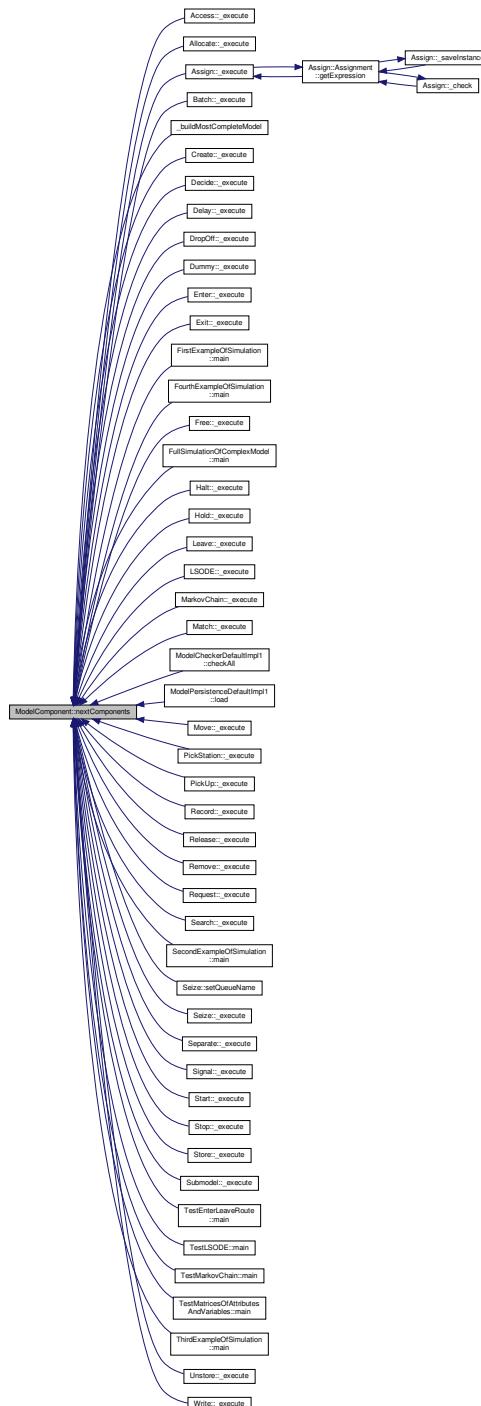
6.70.3.9 nextComponents()

```
ConnectionManager * ModelComponent::nextComponents ( ) const
```

Returns a list of components directly connected to the output. Usually the components have a single output, but they may have none (such as [Dispose](#)) or more than one (as [Decide](#)). In addition to the component, [NextComponents](#) specifies the [inputNumber](#) of the next component where the entity will be sent to. Ussually the components have a single input, but they may have none (such as [Create](#)) or more than one (as [Match](#)).

Definition at line 90 of file [ModelComponent.cpp](#).

Here is the caller graph for this function:

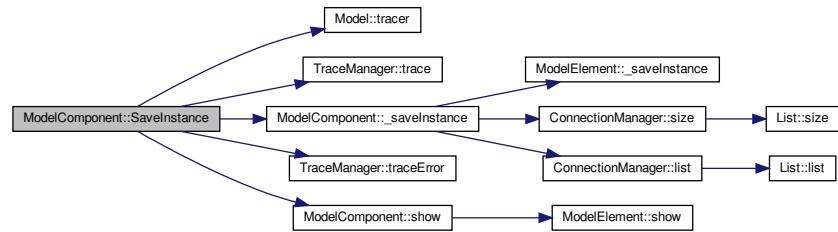


6.70.3.10 SaveInstance()

```
std::map< std::string, std::string > * ModelComponent::SaveInstance (
    ModelComponent * component ) [static]
```

Definition at line 60 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



6.70.3.11 show()

```
std::string ModelComponent::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

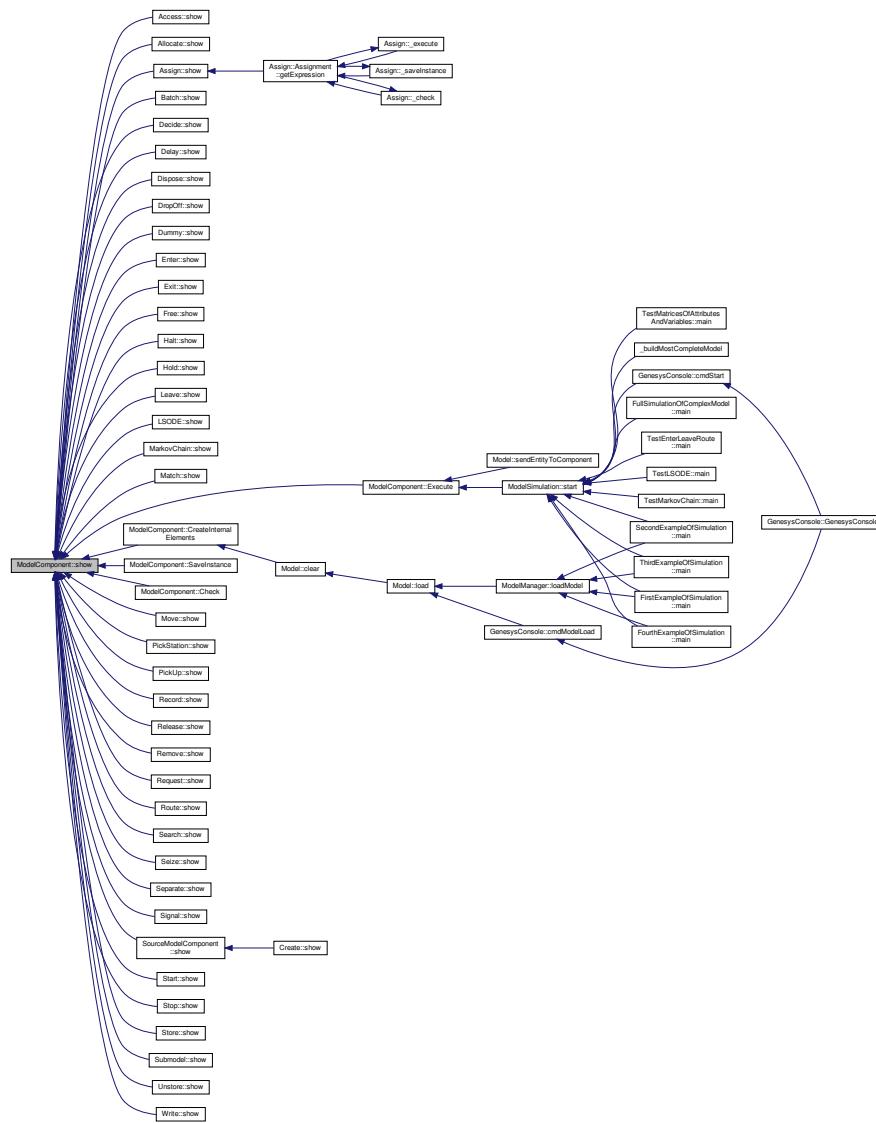
Reimplemented in [Seize](#), [Enter](#), [Leave](#), [Assign](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Allocate](#), [Separate](#), [Release](#), [PickUp](#), [Station](#), [Move](#), [Route](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Match](#), [Delay](#), [Unstore](#), [SourceModelComponent](#), [Write](#), [DropOff](#), [Free](#), [Halt](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [Dummy](#), [Submodel](#), and [MarkovChain](#).

Definition at line 94 of file [ModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



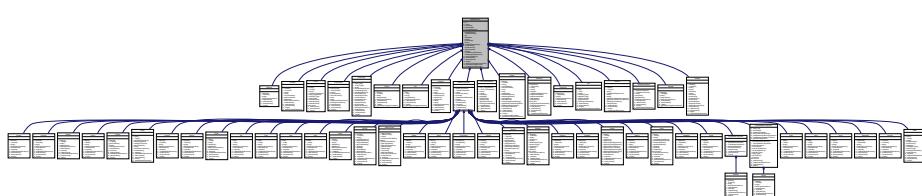
The documentation for this class was generated from the following files:

- [ModelComponent.h](#)
- [ModelComponent.cpp](#)

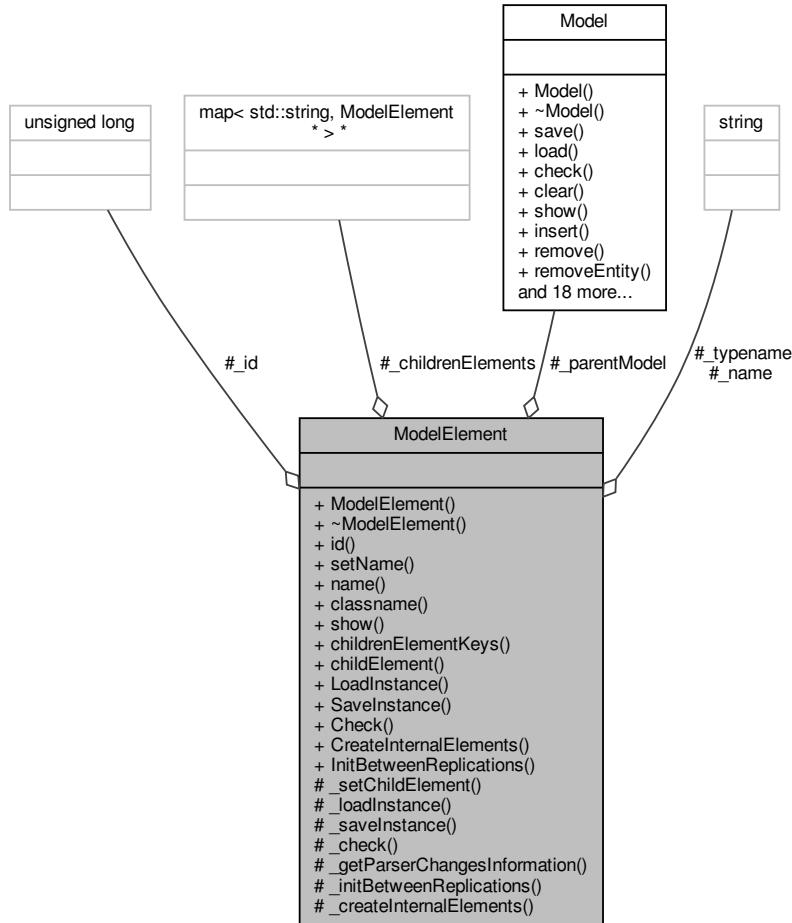
6.71 ModelElement Class Reference

```
#include <ModelElement.h>
```

Inheritance diagram for ModelElement:



Collaboration diagram for ModelElement:



Public Member Functions

- `ModelElement` (`Model` *model, `std::string` elementTypename, `std::string` name="", `bool` insertIntoModel=true)
 - `virtual ~ModelElement` ()
 - `Util::identification` id () const
 - `void setName` (`std::string` _name)
 - `std::string name` () const
 - `std::string classname` () const
 - `virtual std::string show` ()
 - `std::list< std::string > * childrenElementKeys` () const
 - `ModelElement *` childElement (`std::string` key) const

Static Public Member Functions

- static ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields, bool insertIntoModel)
 - static std::map< std::string, std::string > * SaveInstance (ModelElement *element)
 - static bool Check (ModelElement *element, std::string *errorMessage)
 - static void CreateInternalElements (ModelElement *element)
 - static void InitBetweenReplications (ModelElement *element)

Protected Member Functions

- void [_setChildElement](#) (std::string key, [ModelElement](#) *child)
- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)
- virtual [ParserChangesInformation](#) * [_getParserChangesInformation](#) ()
- virtual void [_initBetweenReplications](#) ()
- virtual void [_createInternalElements](#) ()

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Protected Attributes

- [Util::identification _id](#)
- std::string [_name](#)
- std::string [_typename](#)
- [Model](#) * [_parentModel](#)
- std::map< std::string, [ModelElement](#) * > * [_childrenElements](#) = new std::map<std::string, [ModelElement](#)*>()

6.71.1 Detailed Description

This class is the basis for any element of the model (such as [Queue](#), [Resource](#), [Variable](#), etc.) and also for any component of the model. It has the infrastructure to read and write on file and to verify symbols.

Definition at line 31 of file [ModelElement.h](#).

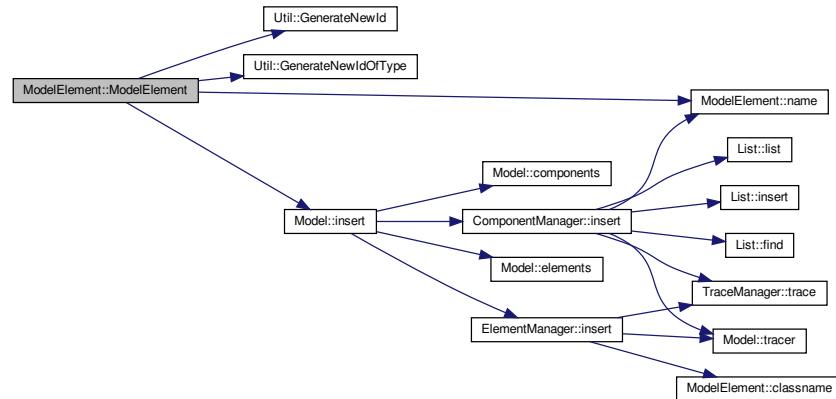
6.71.2 Constructor & Destructor Documentation

6.71.2.1 ModelElement()

```
ModelElement::ModelElement (
    Model * model,
    std::string elementTypename,
    std::string name = "",
    bool insertIntoModel = true )
```

Definition at line 19 of file [ModelElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

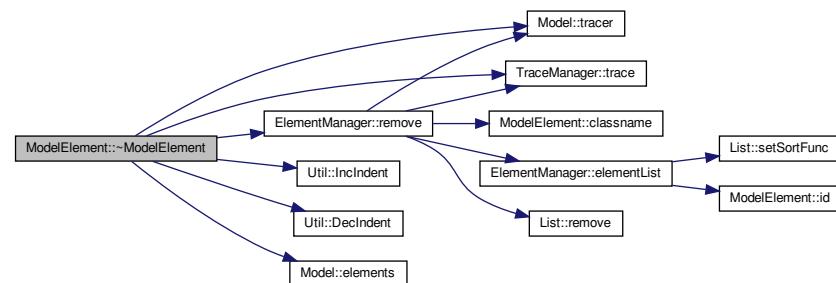


6.71.2.2 ~ModelElement()

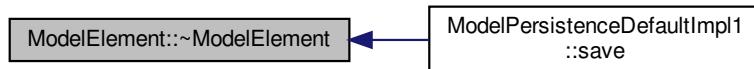
`ModelElement::~ModelElement () [virtual]`

Definition at line 37 of file [ModelElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.71.3 Member Function Documentation

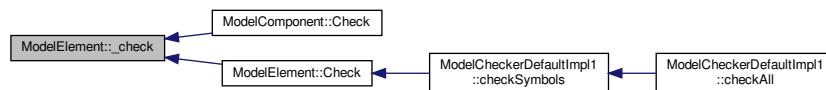
6.71.3.1 _check()

```
bool ModelElement::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Station](#), [Failure](#), [Route](#), [Allocate](#), [Separate](#), [File](#), [PickStation](#), [Attribute](#), [Move](#), [Set](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Sequence](#), [Store](#), [Write](#), [Match](#), [Delay](#), [Unstore](#), [EntityType](#), [DropOff](#), [SourceModelComponent](#), [Free](#), [Halt](#), [OLD_ODElement](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Storage](#), [Stop](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [Formula](#), [Counter](#), [StatisticsCollector](#), [Dummy](#), and [SinkModelComponent](#).

Definition at line 71 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



6.71.3.2 _createInternalElements()

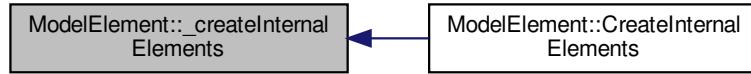
```
void ModelElement::_createInternalElements () [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented in [Resource](#), [Queue](#), [Station](#), [Delay](#), [ModelComponent](#), and [Dispose](#).

Definition at line 183 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



6.71.3.3 `_getParserChangesInformation()`

`ParserChangesInformation * ModelElement::_getParserChangesInformation () [protected], [virtual]`

Reimplemented in [Queue](#), [Failure](#), [File](#), [Set](#), and [Storage](#).

Definition at line 75 of file [ModelElement.cpp](#).

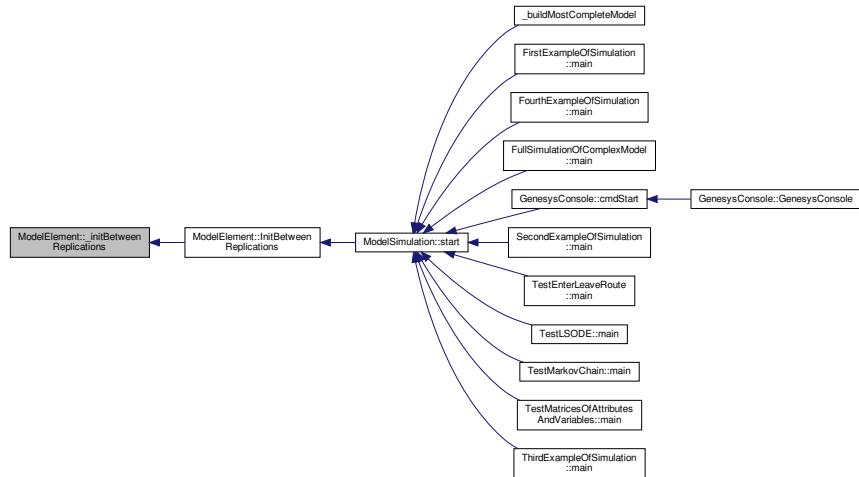
6.71.3.4 `_initBetweenReplications()`

`void ModelElement::_initBetweenReplications () [protected], [virtual]`

Reimplemented in [Seize](#), [Variable](#), [Enter](#), [Leave](#), [Assign](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Route](#), [Allocate](#), [Separate](#), [PickStation](#), [Move](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Store](#), [Write](#), [Match](#), [Delay](#), [Unstore](#), [DropOff](#), [SourceModelComponent](#), [Dispose](#), [Free](#), [Halt](#), [Start](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [LSODE](#), [MarkovChain](#), [Dummy](#), [SinkModelComponent](#), and [Submodel](#).

Definition at line 79 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



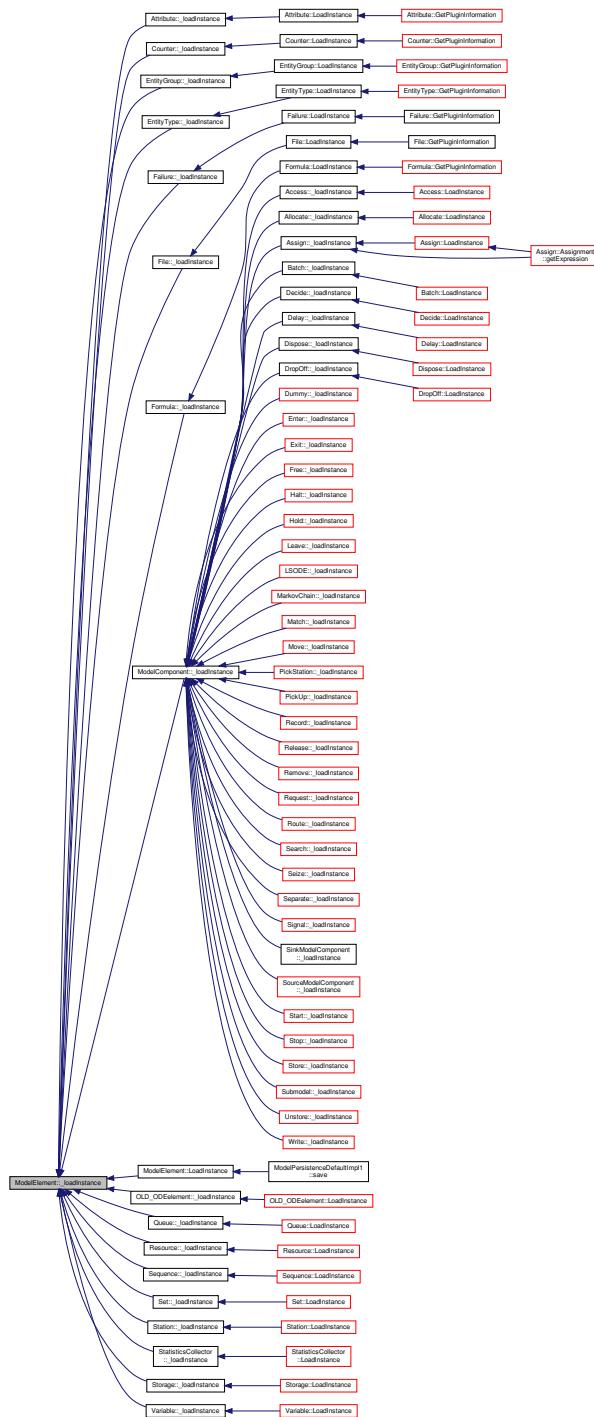
6.71.3.5 `_loadInstance()`

```
bool ModelElement::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Station](#), [Create](#), [Route](#), [Allocate](#), [Failure](#), [Separate](#), [File](#), [PickStation](#), [Attribute](#), [Move](#), [Set](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Sequence](#), [Store](#), [Write](#), [Match](#), [Delay](#), [Unstore](#), [EntityType](#), [ModelComponent](#), [DropOff](#), [Free](#), [Halt](#), [OL↔D_ODElement](#), [SourceModelComponent](#), [Start](#), [PickUp](#), [Signal](#), [Dispose](#), [Exit](#), [Remove](#), [Stop](#), [Storage](#), [LSODE](#), [EntityGroup](#), [MarkovChain](#), [Formula](#), [Counter](#), [StatisticsCollector](#), [Dummy](#), [Submodel](#), and [SinkModelComponent](#).

Definition at line 49 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



6.71.3.6 `_saveInstance()`

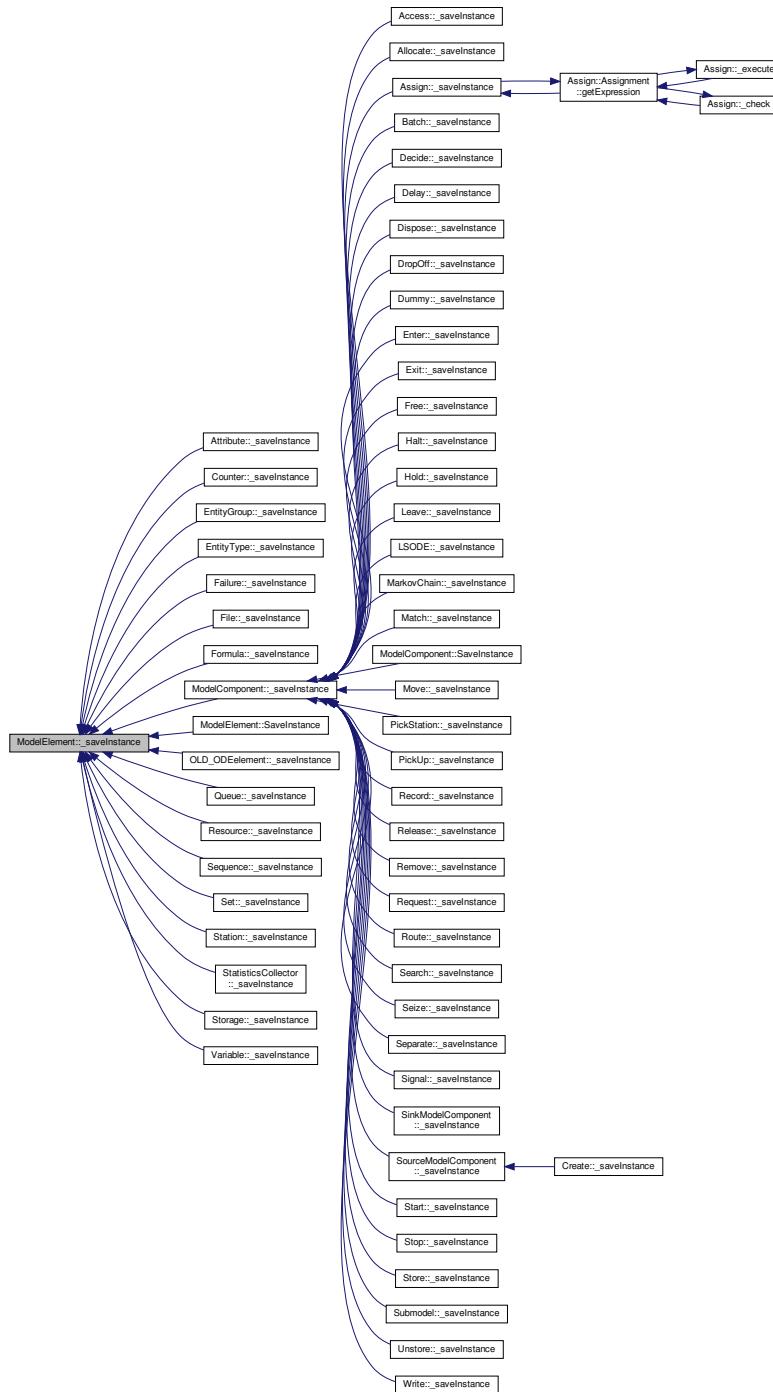
```
std::map< std::string, std::string > * ModelElement::_saveInstance ( ) [protected], [virtual]
```

Reimplemented in [Seize](#), [Resource](#), [Queue](#), [Enter](#), [Variable](#), [Leave](#), [Assign](#), [Entity](#), [Release](#), [Decide](#), [Hold](#), [Record](#), [Create](#), [Station](#), [Route](#), [Allocate](#), [Failure](#), [Separate](#), [File](#), [PickStation](#), [Attribute](#), [Move](#), [Set](#), [Request](#), [Search](#), [Access](#),

Batch, Sequence, Store, Write, Match, Delay, Unstore, EntityType, DropOff, ModelComponent, SourceModel, Component, Free, Halt, OLD_ODElement, Start, Dispose, PickUp, Signal, Exit, Remove, Stop, Storage, LSODE, EntityGroup, MarkovChain, Formula, Counter, StatisticsCollector, Dummy, Submodel, and SinkModelComponent.

Definition at line 63 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



6.71.3.7 `_setChildElement()`

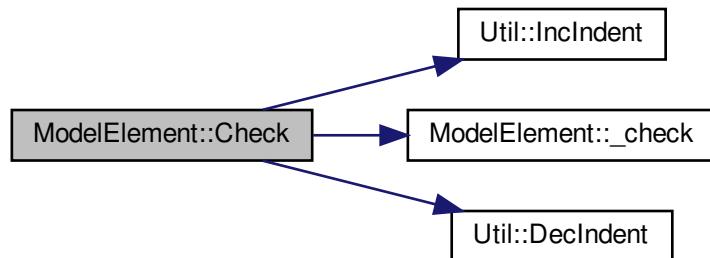
```
void ModelElement::_setChildElement (
    std::string key,
    ModelElement * child ) [protected]
```

6.71.3.8 `Check()`

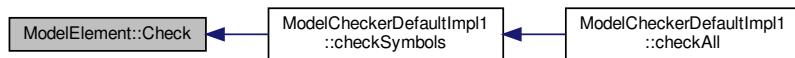
```
bool ModelElement::Check (
    ModelElement * element,
    std::string * errorMessage ) [static]
```

Definition at line 157 of file [ModelElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.71.3.9 `childElement()`

```
ModelElement* ModelElement::childElement (
    std::string key ) const
```

6.71.3.10 childrenElementKeys()

```
std::list< std::string > * ModelElement::childrenElementKeys ( ) const
```

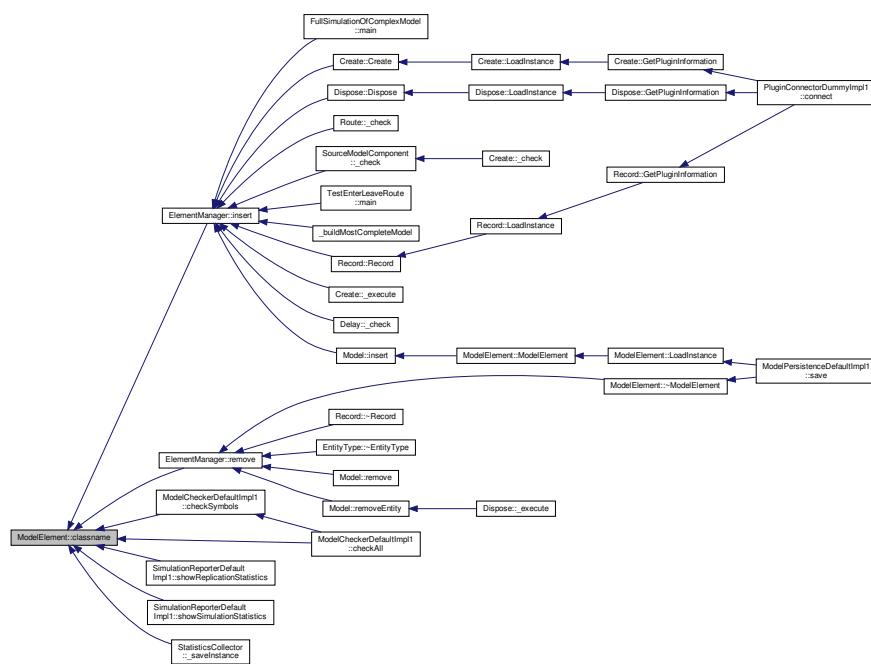
Definition at line 95 of file [ModelElement.cpp](#).

6.71.3.11 classname()

```
std::string ModelElement::classname ( ) const
```

Definition at line 112 of file [ModelElement.cpp](#).

Here is the caller graph for this function:

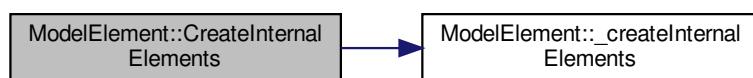


6.71.3.12 CreateInternalElements()

```
void ModelElement::CreateInternalElements (
    ModelElement * element ) [static]
```

Definition at line 175 of file [ModelElement.cpp](#).

Here is the call graph for this function:

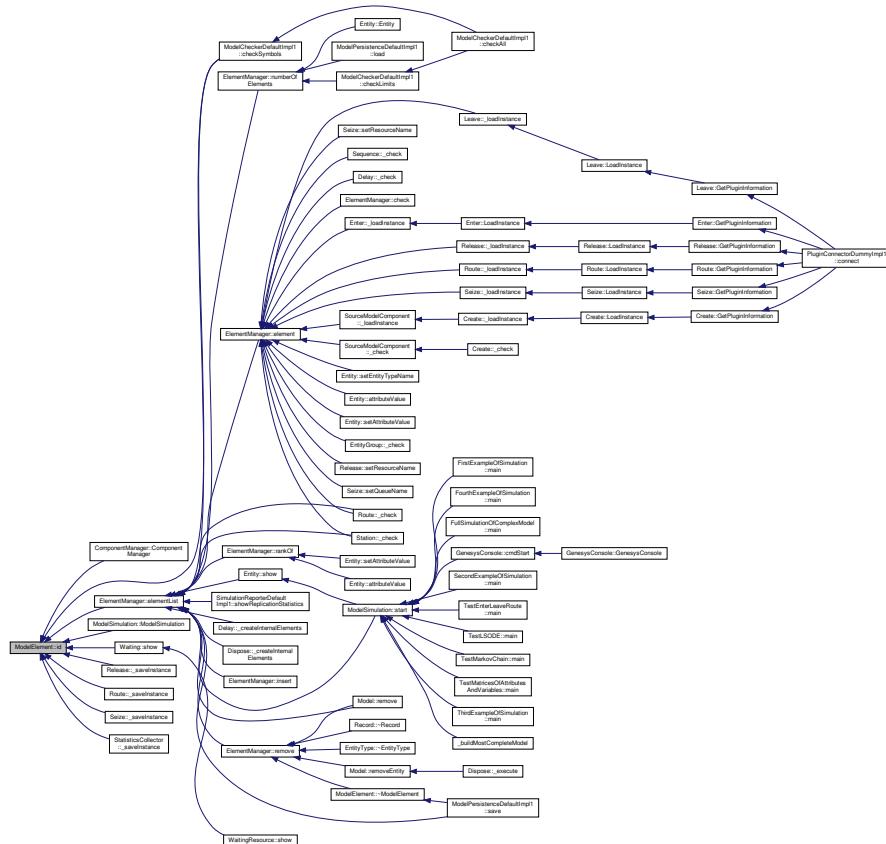


6.71.3.13 id()

```
Util::identification ModelElement::id ( ) const
```

Definition at line 100 of file [ModelElement.cpp](#).

Here is the caller graph for this function:

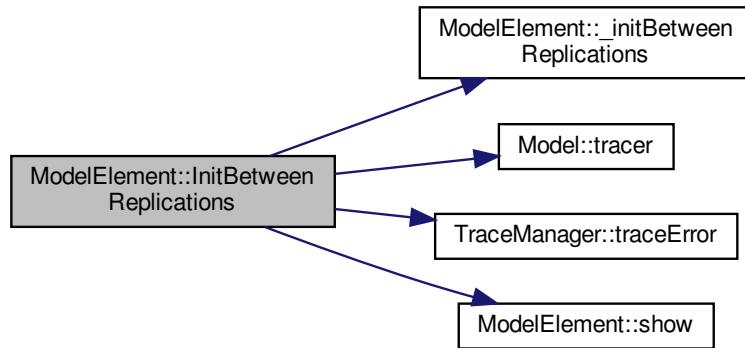


6.71.3.14 InitBetweenReplications()

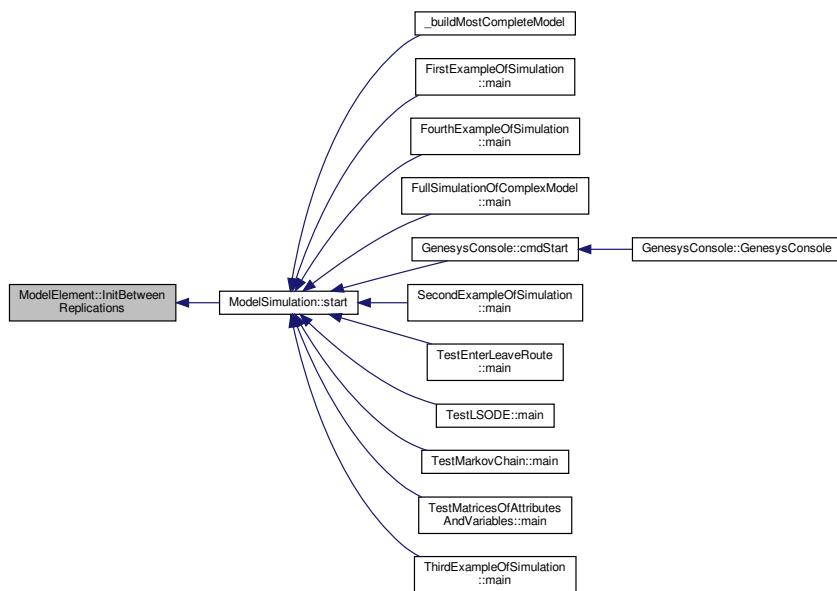
```
void ModelElement::InitBetweenReplications (
    ModelElement * element ) [static]
```

Definition at line 121 of file [ModelElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



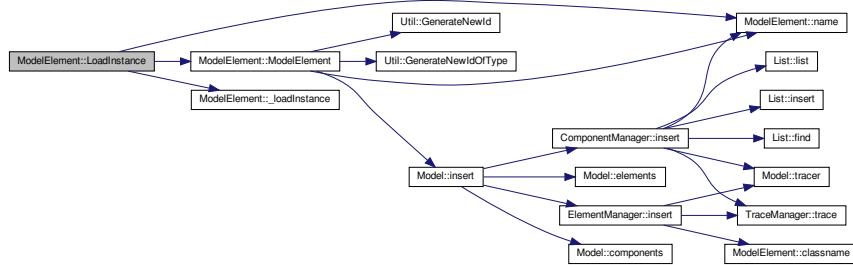
6.71.3.15 LoadInstance()

```

ModelElement * ModelElement::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields,
    bool insertIntoModel ) [static]
  
```

Definition at line 130 of file [ModelElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

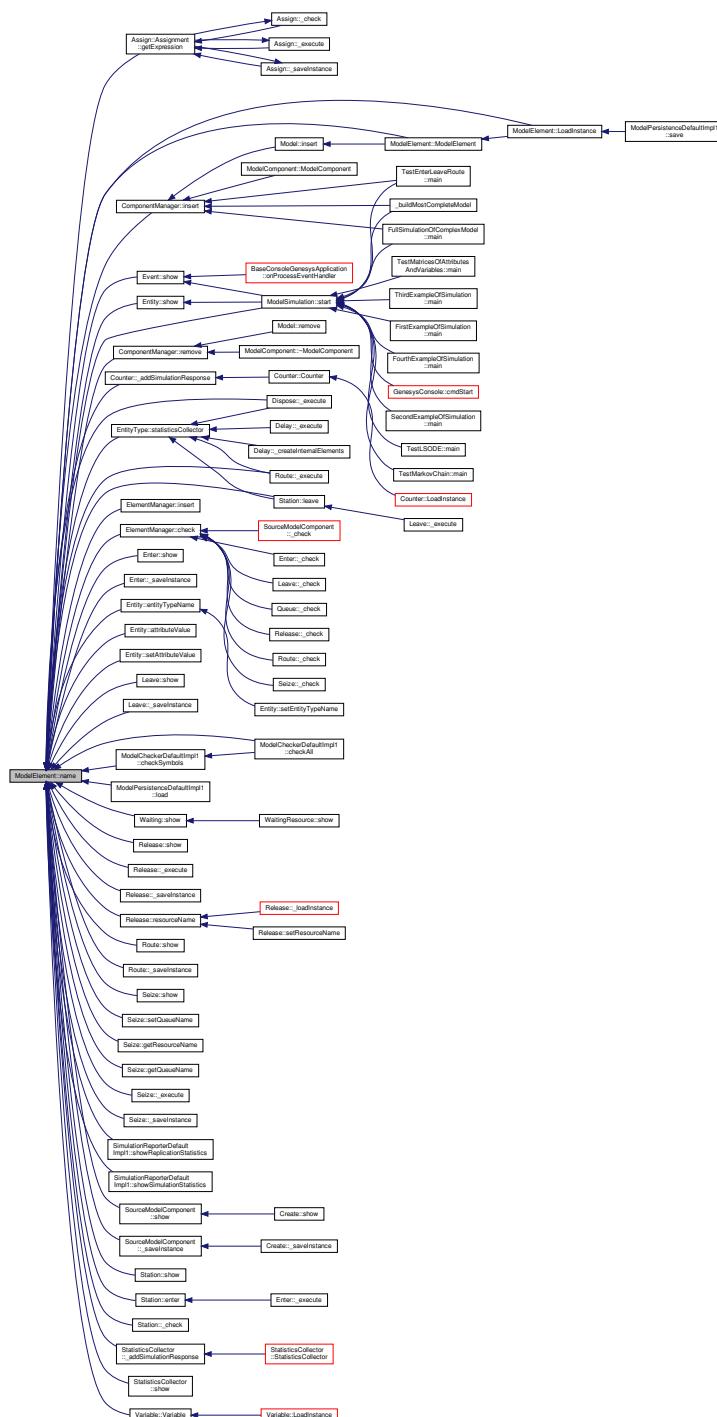


6.71.3.16 name()

```
std::string ModelElement::name ( ) const
```

Definition at line 108 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



6.71.3.17 SaveInstance()

```
std::map< std::string, std::string > * ModelElement::SaveInstance (
    ModelElement * element ) [static]
```

Definition at line 147 of file [ModelElement.cpp](#).

Here is the call graph for this function:

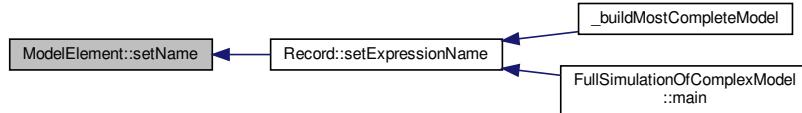


6.71.3.18 setName()

```
void ModelElement::setName (
    std::string _name )
```

Definition at line 104 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



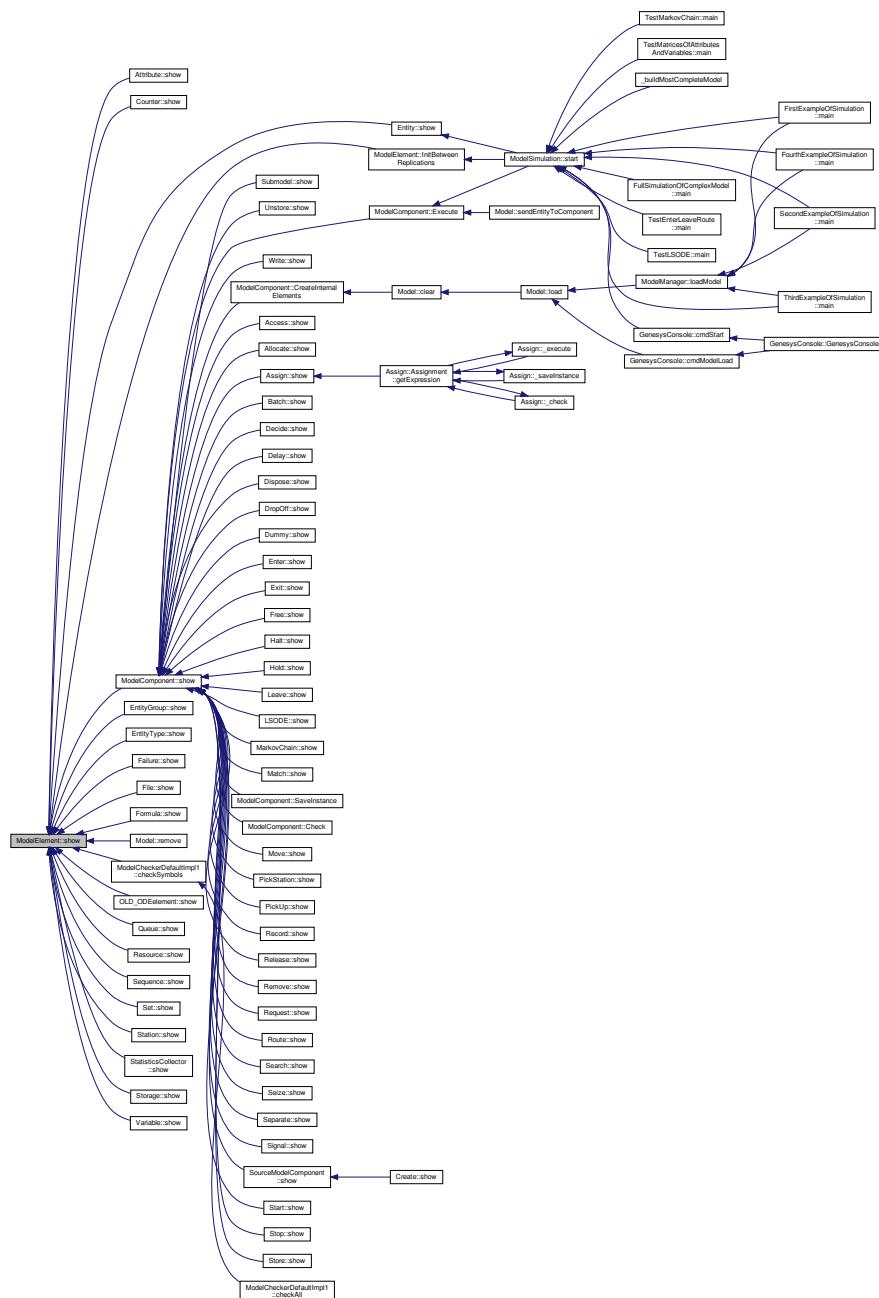
6.71.3.19 show()

```
std::string ModelElement::show ( ) [virtual]
```

Reimplemented in [Seize](#), [Resource](#), [Enter](#), [Queue](#), [Leave](#), [Assign](#), [Variable](#), [Decide](#), [Entity](#), [Hold](#), [Failure](#), [Record](#), [Create](#), [File](#), [Allocate](#), [Separate](#), [Station](#), [Release](#), [Attribute](#), [PickStation](#), [Move](#), [Set](#), [Route](#), [Request](#), [Search](#), [Access](#), [Batch](#), [Sequence](#), [Store](#), [Match](#), [Delay](#), [Unstore](#), [SourceModelComponent](#), [Write](#), [DropOff](#), [Storage](#), [Free](#), [Halt](#), [Start](#), [Dispose](#), [PickUp](#), [Signal](#), [Exit](#), [Remove](#), [Stop](#), [OLD_ODElement](#), [ModelComponent](#), [EntityType](#), [EntityGroup](#), [StatisticsCollector](#), [Counter](#), [LSODE](#), [Dummy](#), [Formula](#), [Submodel](#), and [MarkovChain](#).

Definition at line 91 of file [ModelElement.cpp](#).

Here is the caller graph for this function:



6.71.4 Member Data Documentation

6.71.4.1 childrenElements

```
std::map<std::string, ModelElement*>* ModelElement::_childrenElements = new std::map<std::string,ModelElement*>() [protected]
```

Definition at line 72 of file [ModelElement.h](#).

6.71.4.2 `_id`

`Util::identification ModelElement::_id [protected]`

Definition at line 67 of file [ModelElement.h](#).

6.71.4.3 `_name`

`std::string ModelElement::_name [protected]`

Definition at line 68 of file [ModelElement.h](#).

6.71.4.4 `_parentModel`

`Model* ModelElement::_parentModel [protected]`

Definition at line 70 of file [ModelElement.h](#).

6.71.4.5 `_typename`

`std::string ModelElement::_typename [protected]`

Definition at line 69 of file [ModelElement.h](#).

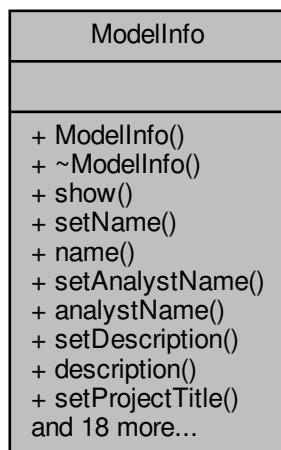
The documentation for this class was generated from the following files:

- [ModelElement.h](#)
- [ModelElement.cpp](#)

6.72 ModellInfo Class Reference

#include <ModellInfo.h>

Collaboration diagram for ModellInfo:



Public Member Functions

- `ModelInfo ()`
- `virtual ~ModelInfo ()=default`
- `std::string show ()`
- `void setName (std::string _name)`
- `std::string name () const`
- `void setAnalystName (std::string _analystName)`
- `std::string analystName () const`
- `void setDescription (std::string _description)`
- `std::string description () const`
- `void setProjectTitle (std::string _projectTitle)`
- `std::string projectTitle () const`
- `void setVersion (std::string _version)`
- `std::string version () const`
- `void setNumberOfReplications (unsigned int _numberOfReplications)`
- `unsigned int numberOfReplications () const`
- `void setReplicationLength (double _replicationLength)`
- `double replicationLength () const`
- `void setReplicationLengthTimeUnit (Util::TimeUnit _replicationLengthTimeUnit)`
- `Util::TimeUnit replicationLengthTimeUnit () const`
- `void setWarmUpPeriod (double _warmUpPeriod)`
- `double warmUpPeriod () const`
- `void setWarmUpPeriodTimeUnit (Util::TimeUnit _warmUpPeriodTimeUnit)`
- `Util::TimeUnit warmUpPeriodTimeUnit () const`
- `void setTerminatingCondition (std::string _terminatingCondition)`
- `std::string terminatingCondition () const`
- `void loadInstance (std::map< std::string, std::string > *fields)`
- `std::map< std::string, std::string > * saveInstance ()`
- `bool hasChanged () const`

6.72.1 Detailed Description

`ModelInfo` stores basic model project information.

Definition at line 23 of file `ModelInfo.h`.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 ModelInfo()

```
ModelInfo::ModelInfo ( )
```

Definition at line 18 of file `ModelInfo.cpp`.

6.72.2.2 ~ModelInfo()

```
virtual ModelInfo::~ModelInfo ( ) [virtual], [default]
```

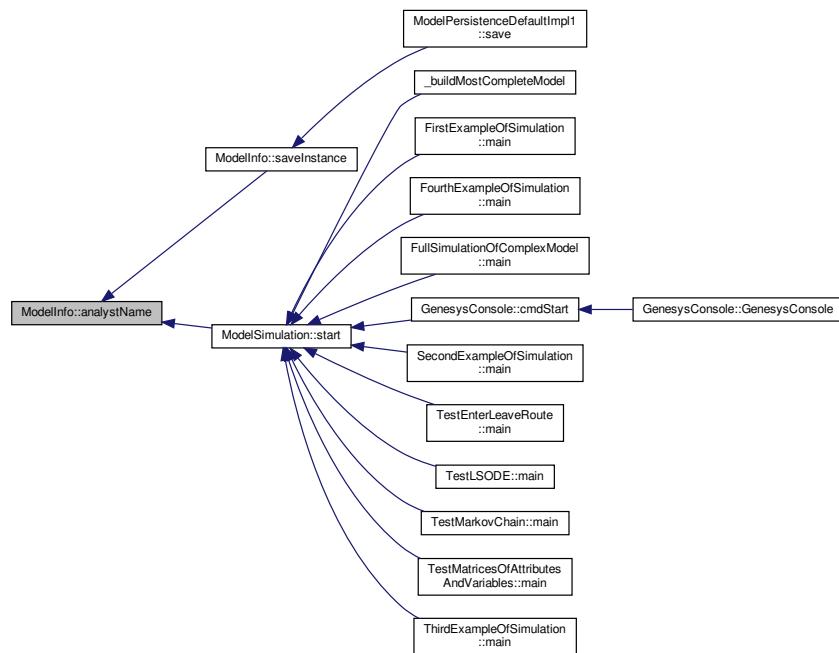
6.72.3 Member Function Documentation

6.72.3.1 analystName()

```
std::string ModelInfo::analystName() const
```

Definition at line 46 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

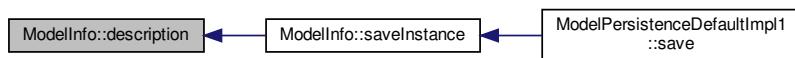


6.72.3.2 description()

```
std::string ModelInfo::description() const
```

Definition at line 55 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

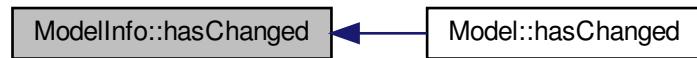


6.72.3.3 hasChanged()

```
bool ModelInfo::hasChanged ( ) const
```

Definition at line 167 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:



6.72.3.4 loadInstance()

```
void ModelInfo::loadInstance (   
    std::map< std::string, std::string > * fields )
```

Definition at line 132 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

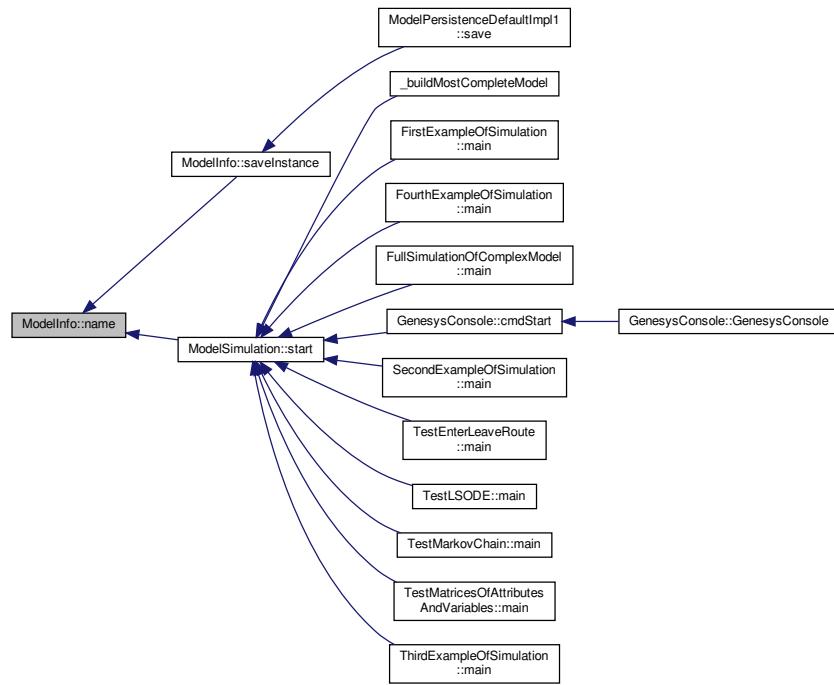


6.72.3.5 name()

```
std::string ModelInfo::name ( ) const
```

Definition at line 37 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

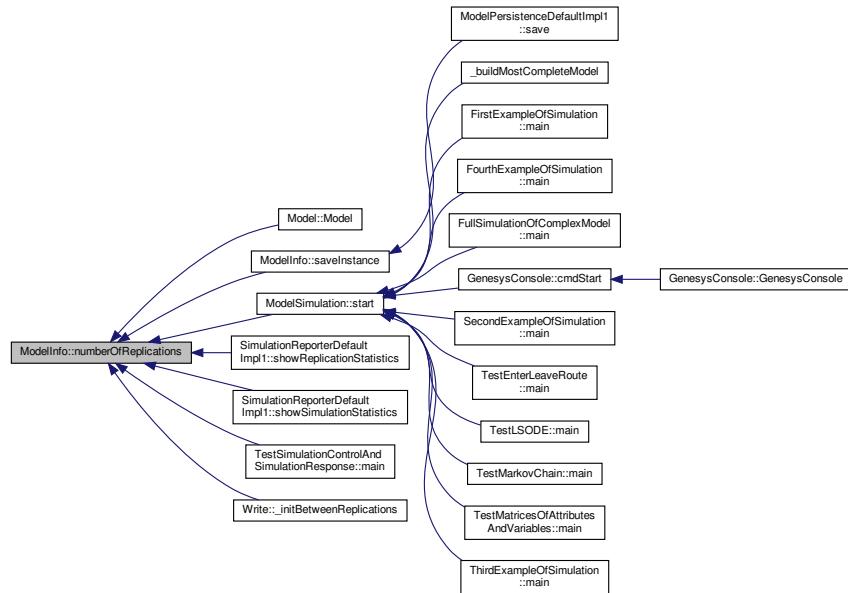


6.72.3.6 `numberOfReplications()`

```
unsigned int ModelInfo::numberOfReplications ( ) const
```

Definition at line 82 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

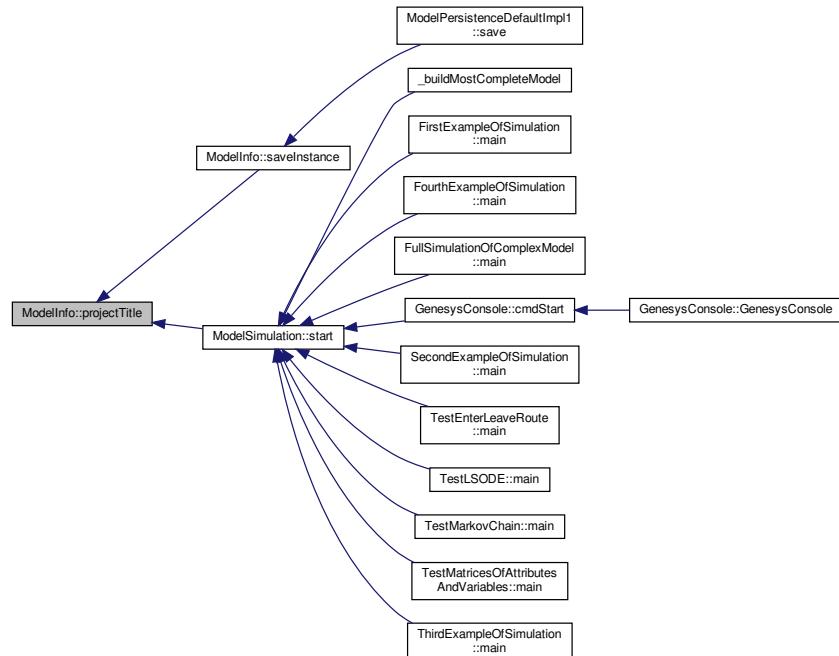


6.72.3.7 projectTitle()

```
std::string ModelInfo::projectTitle ( ) const
```

Definition at line 64 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

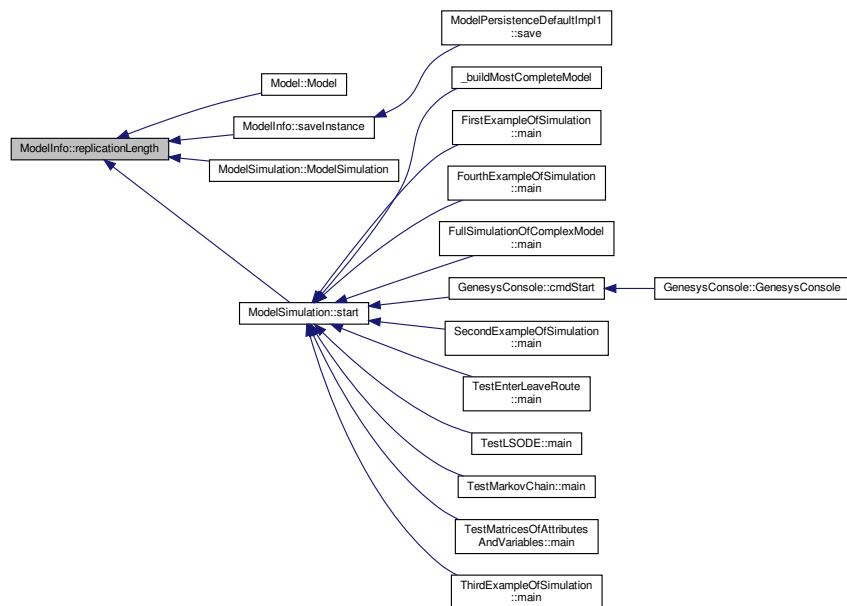


6.72.3.8 replicationLength()

```
double ModelInfo::replicationLength ( ) const
```

Definition at line 91 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:

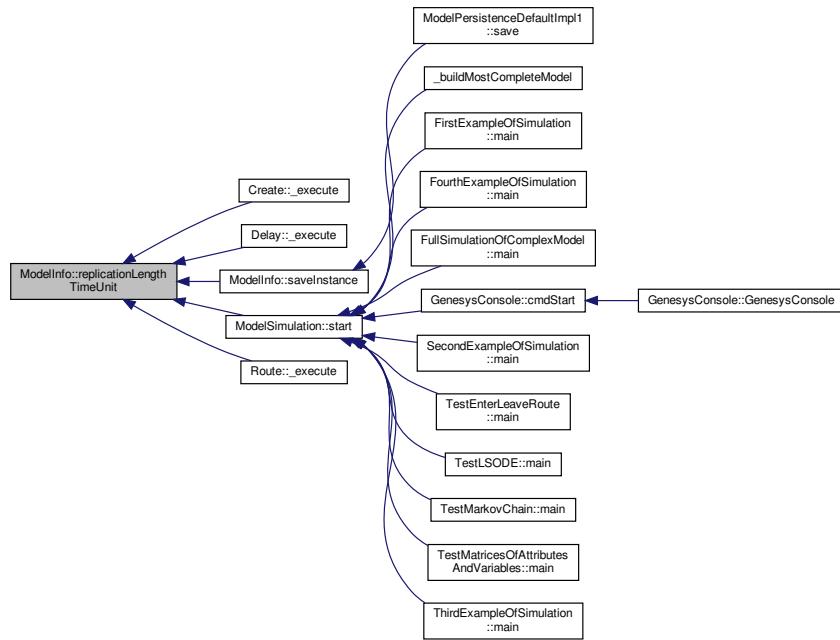


6.72.3.9 replicationLengthTimeUnit()

```
Util::TimeUnit ModelInfo::replicationLengthTimeUnit ( ) const
```

Definition at line 100 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:

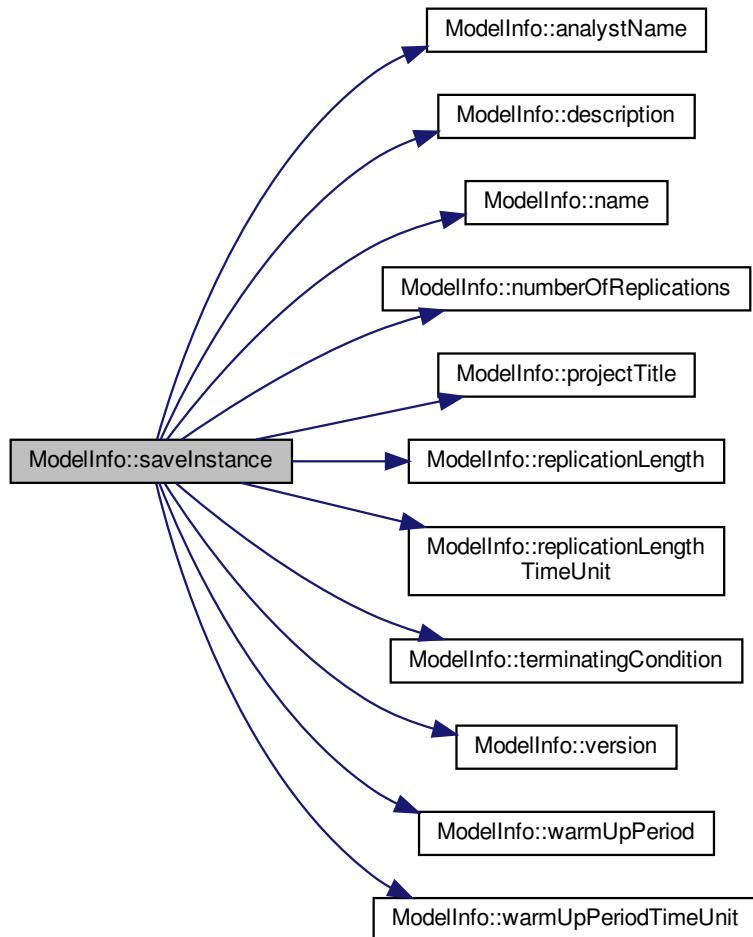


6.72.3.10 saveInstance()

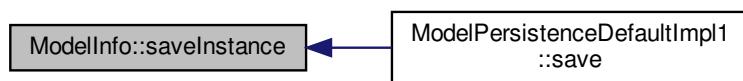
```
std::map< std::string, std::string > * ModelInfo::saveInstance ( )
```

Definition at line 149 of file [ModelInfo.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



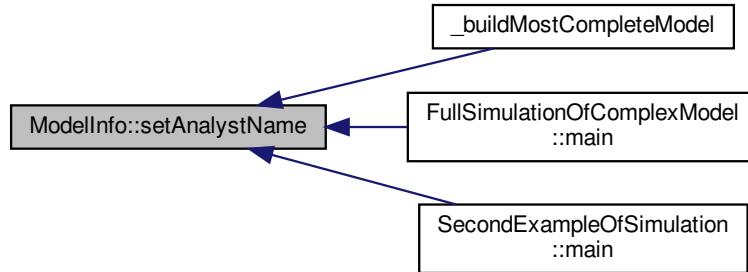
6.72.3.11 setAnalystName()

```

void ModelInfo::setAnalystName (
    std::string _analystName )
  
```

Definition at line 41 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

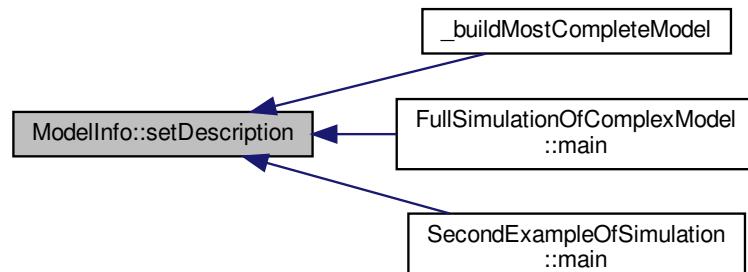


6.72.3.12 setDescription()

```
void ModelInfo::setDescription ( std::string _description )
```

Definition at line 50 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:



6.72.3.13 setName()

```
void ModelInfo::setName ( std::string _name )
```

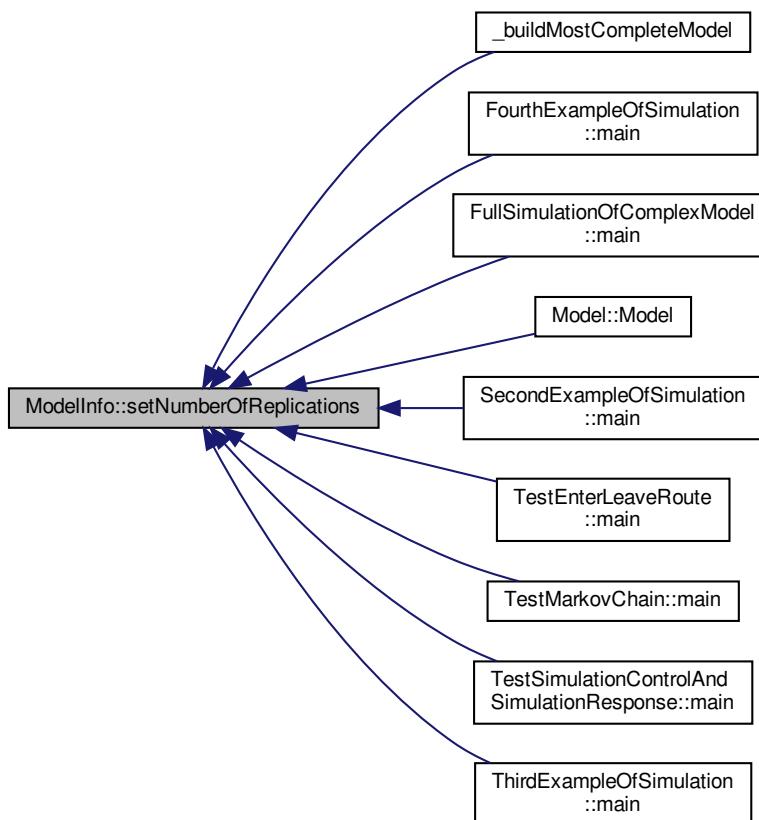
Definition at line 32 of file [ModelInfo.cpp](#).

6.72.3.14 setNumberOfReplications()

```
void ModelInfo::setNumberOfReplications (  
    unsigned int _numberOfReplications )
```

Definition at line 77 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

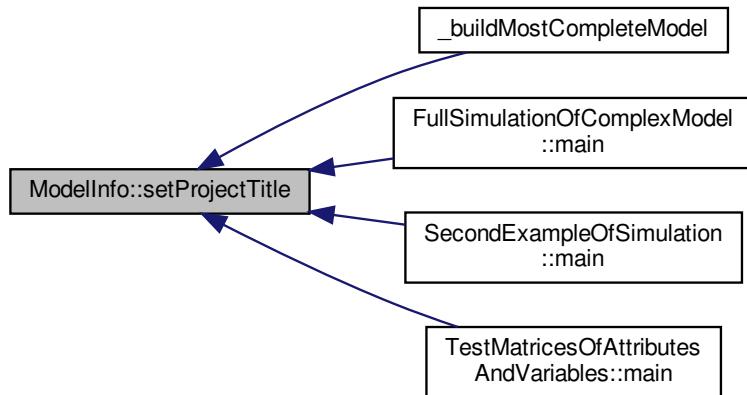


6.72.3.15 setProjectTitle()

```
void ModelInfo::setProjectTitle (  
    std::string _projectTitle )
```

Definition at line 59 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

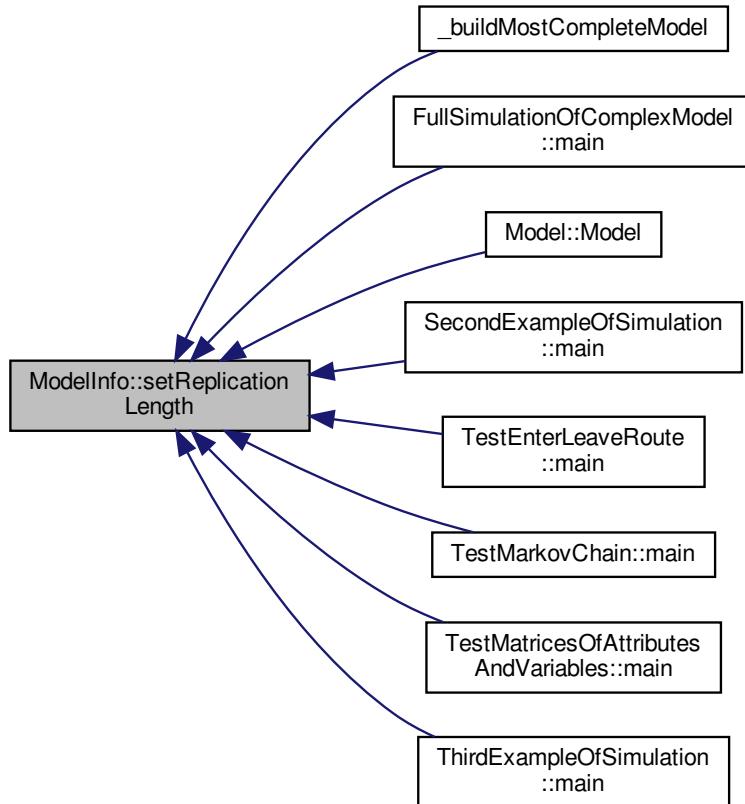


6.72.3.16 setReplicationLength()

```
void ModelInfo::setReplicationLength ( double _replicationLength )
```

Definition at line 86 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

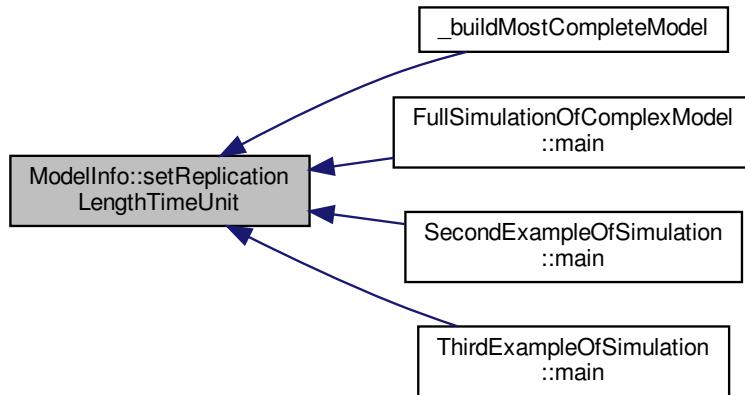


6.72.3.17 `setReplicationLengthTimeUnit()`

```
void ModelInfo::setReplicationLengthTimeUnit (
    Util::TimeUnit _replicationLengthTimeUnit )
```

Definition at line 95 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

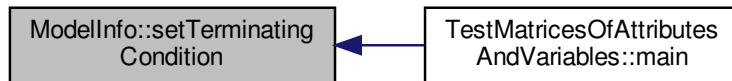


6.72.3.18 setTerminatingCondition()

```
void ModelInfo::setTerminatingCondition ( std::string _terminatingCondition )
```

Definition at line 122 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:



6.72.3.19 setVersion()

```
void ModelInfo::setVersion ( std::string _version )
```

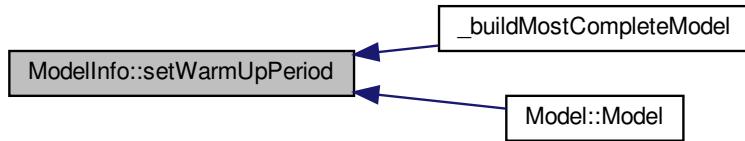
Definition at line 68 of file [ModellInfo.cpp](#).

6.72.3.20 setWarmUpPeriod()

```
void ModelInfo::setWarmUpPeriod (  
    double _warmUpPeriod )
```

Definition at line 104 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:

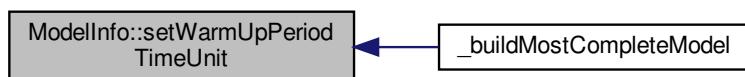


6.72.3.21 setWarmUpPeriodTimeUnit()

```
void ModelInfo::setWarmUpPeriodTimeUnit (   
    Util::TimeUnit _warmUpPeriodTimeUnit )
```

Definition at line 113 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:



6.72.3.22 show()

```
std::string ModelInfo::show ( )
```

Definition at line 21 of file [ModellInfo.cpp](#).

Here is the call graph for this function:

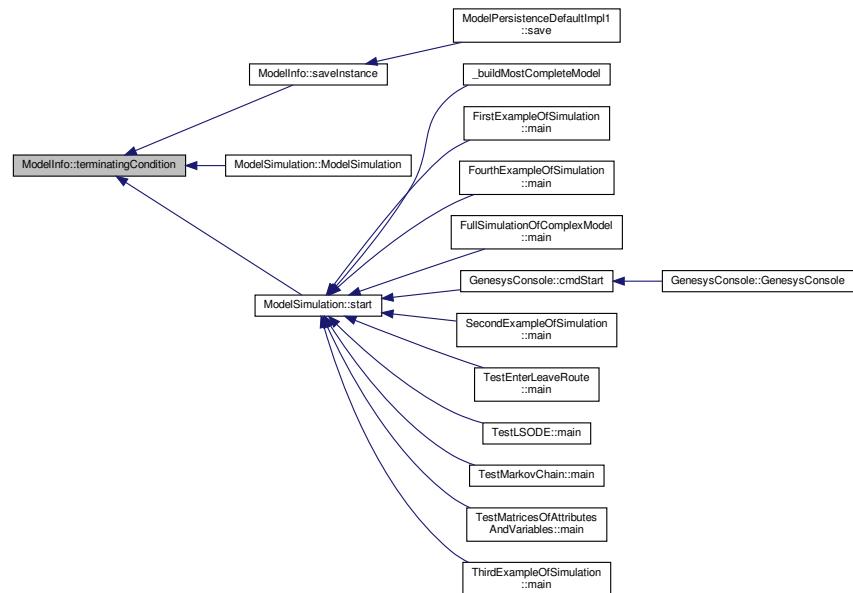


6.72.3.23 terminatingCondition()

```
std::string ModelInfo::terminatingCondition() const
```

Definition at line 127 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:



6.72.3.24 version()

```
std::string ModelInfo::version() const
```

Definition at line 73 of file [ModelInfo.cpp](#).

Here is the caller graph for this function:

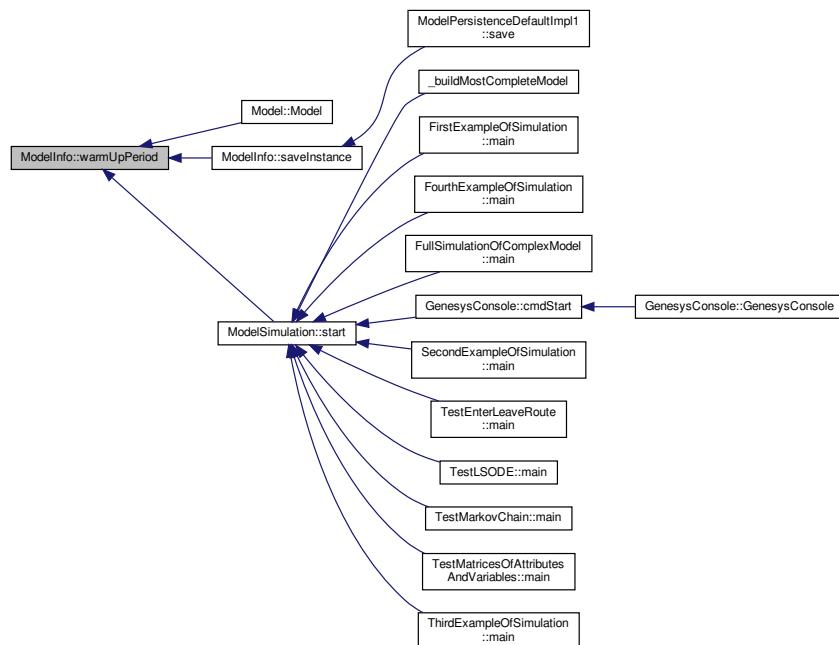


6.72.3.25 warmUpPeriod()

```
double ModelInfo::warmUpPeriod() const
```

Definition at line 109 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:

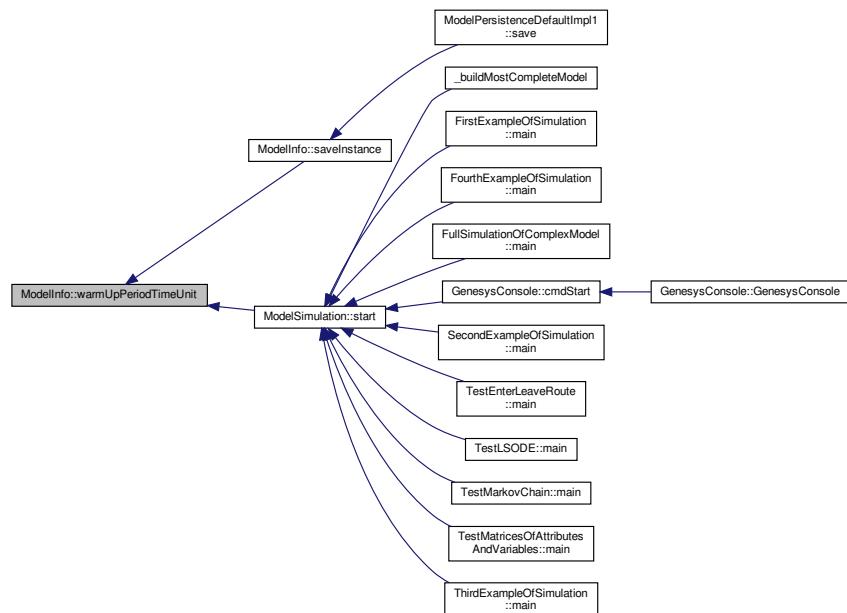


6.72.3.26 warmUpPeriodTimeUnit()

```
Util::TimeUnit ModelInfo::warmUpPeriodTimeUnit() const
```

Definition at line 118 of file [ModellInfo.cpp](#).

Here is the caller graph for this function:



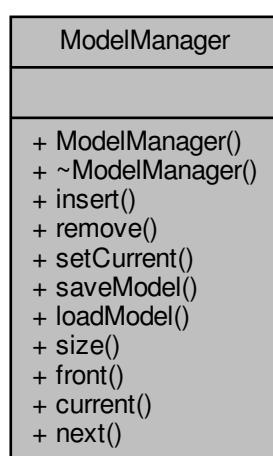
The documentation for this class was generated from the following files:

- [ModellInfo.h](#)
- [ModellInfo.cpp](#)

6.73 ModelManager Class Reference

```
#include <ModelManager.h>
```

Collaboration diagram for ModelManager:



Public Member Functions

- `ModelManager (Simulator *simulator)`
- `virtual ~ModelManager ()=default`
- `void insert (Model *model)`
- `void remove (Model *model)`
- `void setCurrent (Model *model)`
- `bool saveModel (std::string filename)`
- `bool loadModel (std::string filename)`
- `unsigned int size ()`
- `Model * front ()`
- `Model * current ()`
- `Model * next ()`

6.73.1 Detailed Description

Definition at line 20 of file [ModelManager.h](#).

6.73.2 Constructor & Destructor Documentation

6.73.2.1 ModelManager()

```
ModelManager::ModelManager (
    Simulator * simulator )
```

Definition at line 18 of file [ModelManager.cpp](#).

6.73.2.2 ~ModelManager()

```
virtual ModelManager::~ModelManager ( ) [virtual], [default]
```

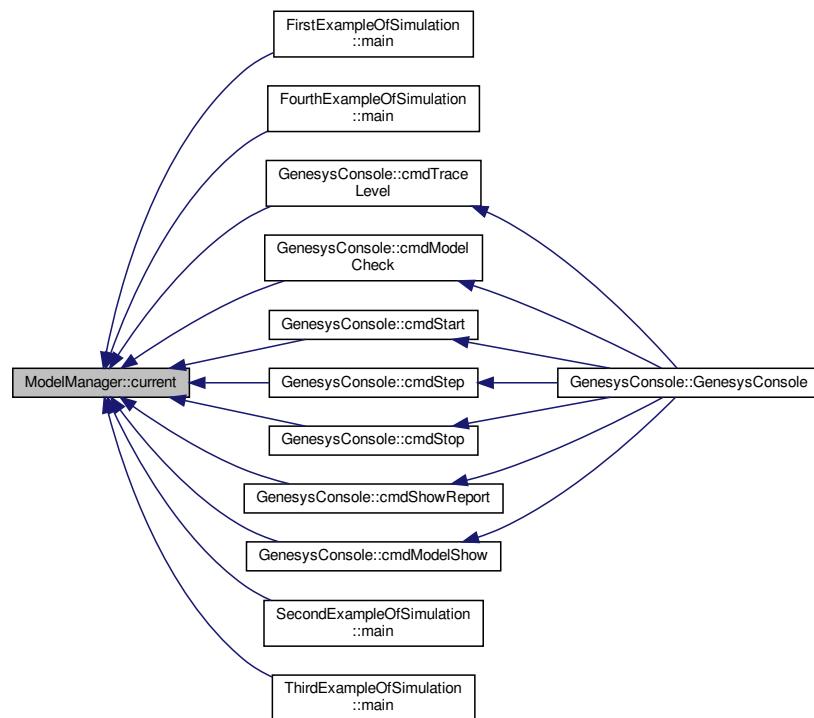
6.73.3 Member Function Documentation

6.73.3.1 current()

```
Model * ModelManager::current ( )
```

Definition at line 65 of file [ModelManager.cpp](#).

Here is the caller graph for this function:



6.73.3.2 front()

```
Model * ModelManager::front ( )
```

Definition at line 69 of file [ModelManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

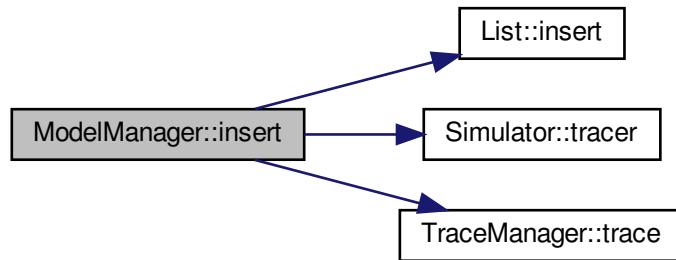


6.73.3.3 insert()

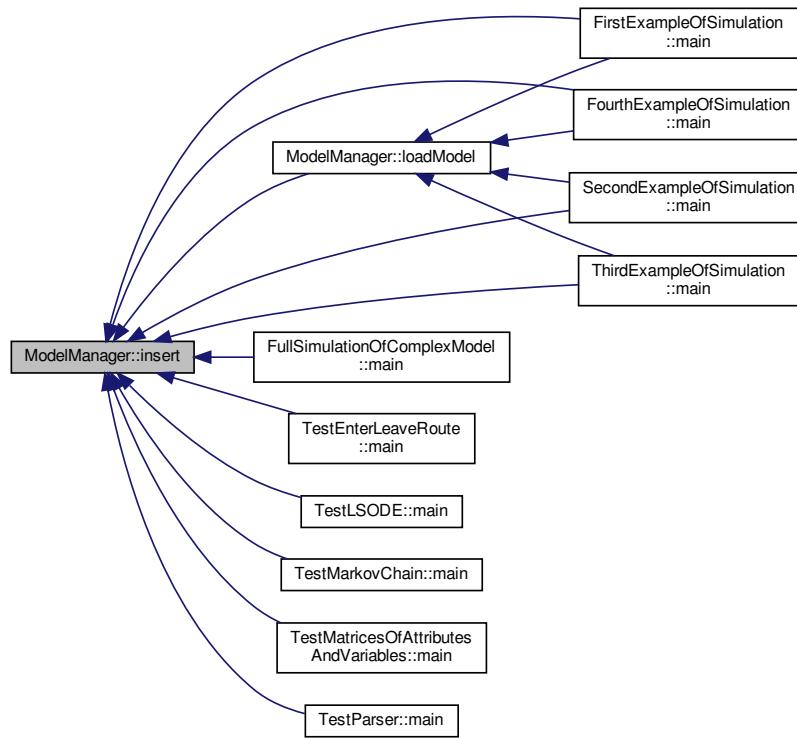
```
void ModelManager::insert ( Model * model )
```

Definition at line 23 of file [ModelManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

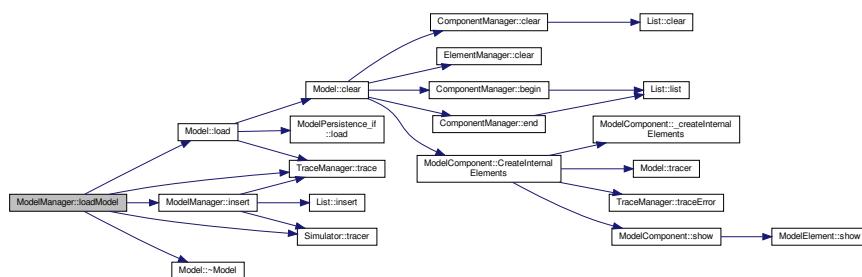


6.73.3.4 loadModel()

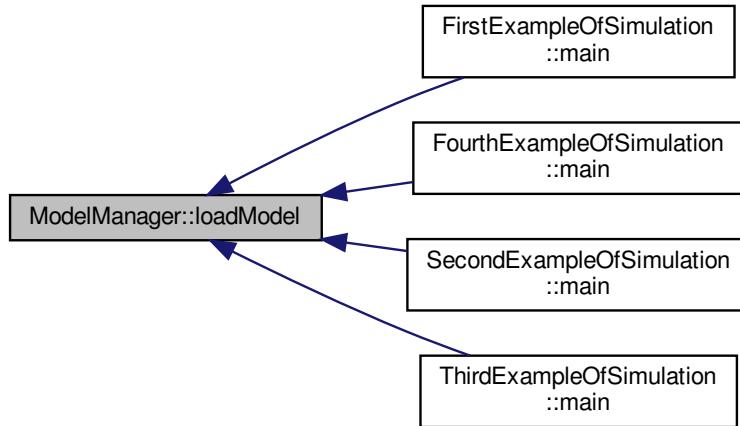
```
bool ModelManager::loadModel (
    std::string filename )
```

Definition at line 48 of file [ModelManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.73.3.5 next()

```
Model * ModelManager::next ( )
```

Definition at line 73 of file [ModelManager.cpp](#).

Here is the call graph for this function:

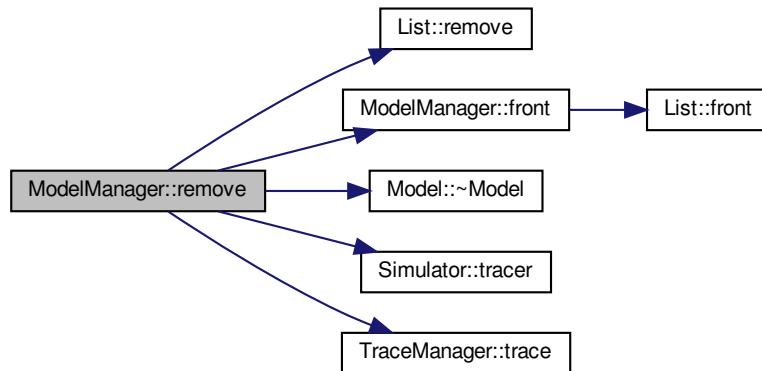


6.73.3.6 remove()

```
void ModelManager::remove ( Model * model )
```

Definition at line 29 of file [ModelManager.cpp](#).

Here is the call graph for this function:

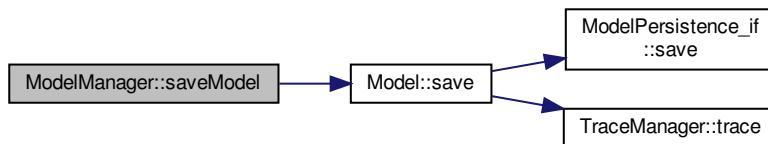


6.73.3.7 saveModel()

```
bool ModelManager::saveModel ( std::string filename )
```

Definition at line 42 of file [ModelManager.cpp](#).

Here is the call graph for this function:



6.73.3.8 setCurrent()

```
void ModelManager::setCurrent ( Model * model )
```

Definition at line 61 of file [ModelManager.cpp](#).

6.73.3.9 size()

```
unsigned int ModelManager::size ( )
```

Definition at line 38 of file [ModelManager.cpp](#).

Here is the call graph for this function:



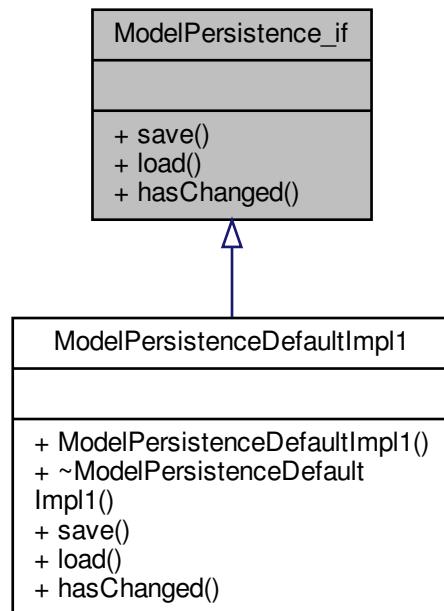
The documentation for this class was generated from the following files:

- [ModelManager.h](#)
- [ModelManager.cpp](#)

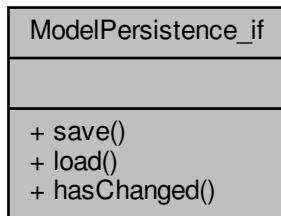
6.74 ModelPersistence_if Class Reference

```
#include <ModelPersistence_if.h>
```

Inheritance diagram for ModelPersistence_if:



Collaboration diagram for ModelPersistence_if:



Public Member Functions

- virtual bool `save` (std::string filename)=0
- virtual bool `load` (std::string filename)=0
- virtual bool `hasChanged` ()=0

6.74.1 Detailed Description

First and inadequate interface for model persistence. It should use the best pattern for the DAO approach

Definition at line 22 of file [ModelPersistence_if.h](#).

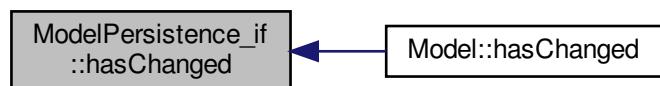
6.74.2 Member Function Documentation

6.74.2.1 hasChanged()

```
virtual bool ModelPersistence_if::hasChanged ( ) [pure virtual]
```

Implemented in [ModelPersistenceDefaultImpl1](#).

Here is the caller graph for this function:

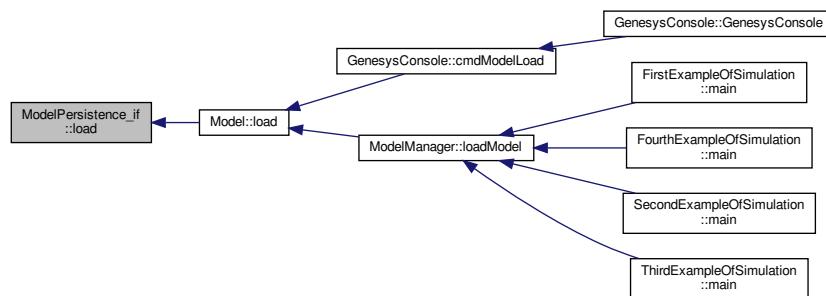


6.74.2.2 load()

```
virtual bool ModelPersistence_if::load (
    std::string filename ) [pure virtual]
```

Implemented in [ModelPersistenceDefaultImpl1](#).

Here is the caller graph for this function:

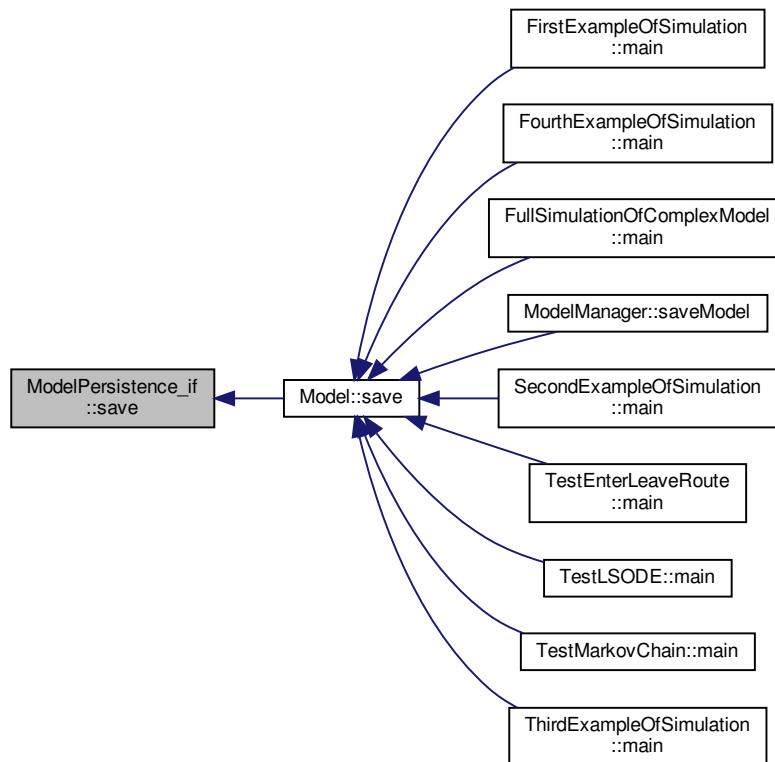


6.74.2.3 save()

```
virtual bool ModelPersistence_if::save (
    std::string filename ) [pure virtual]
```

Implemented in [ModelPersistenceDefaultImpl1](#).

Here is the caller graph for this function:



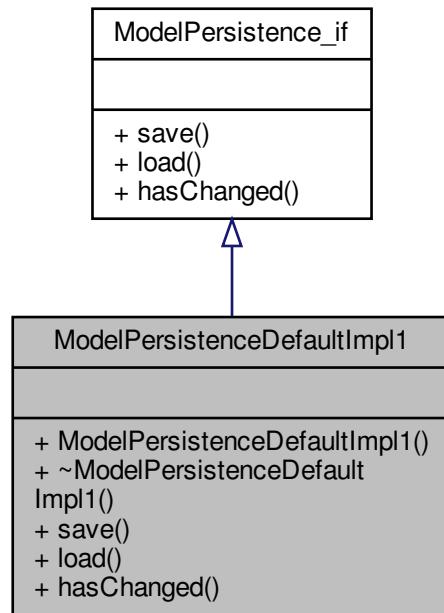
The documentation for this class was generated from the following file:

- [ModelPersistence_if.h](#)

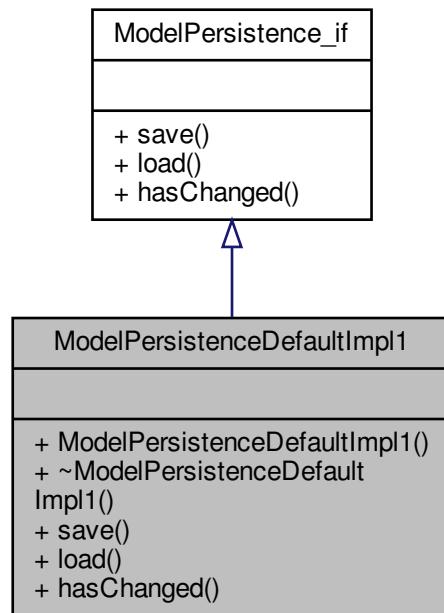
6.75 ModelPersistenceDefaultImpl1 Class Reference

```
#include <ModelPersistenceDefaultImpl1.h>
```

Inheritance diagram for ModelPersistenceDefaultImpl1:



Collaboration diagram for ModelPersistenceDefaultImpl1:



Public Member Functions

- `ModelPersistenceDefaultImpl1 (Model *model)`
- `virtual ~ModelPersistenceDefaultImpl1 ()=default`
- `virtual bool save (std::string filename)`
- `virtual bool load (std::string filename)`
- `virtual bool hasChanged ()`

6.75.1 Detailed Description

Definition at line 20 of file [ModelPersistenceDefaultImpl1.h](#).

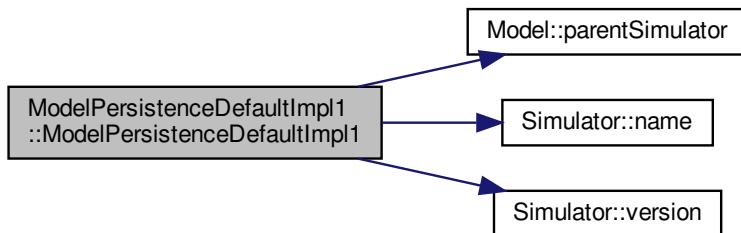
6.75.2 Constructor & Destructor Documentation

6.75.2.1 ModelPersistenceDefaultImpl1()

```
ModelPersistenceDefaultImpl1::ModelPersistenceDefaultImpl1 (
    Model * model )
```

Definition at line 22 of file [ModelPersistenceDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.75.2.2 ~ModelPersistenceDefaultImpl1()

```
virtual ModelPersistenceDefaultImpl1::~ModelPersistenceDefaultImpl1 ( ) [virtual], [default]
```

6.75.3 Member Function Documentation

6.75.3.1 hasChanged()

```
bool ModelPersistenceDefaultImpl1::hasChanged ( ) [virtual]
```

Implements [ModelPersistence_if](#).

Definition at line 273 of file [ModelPersistenceDefaultImpl1.cpp](#).

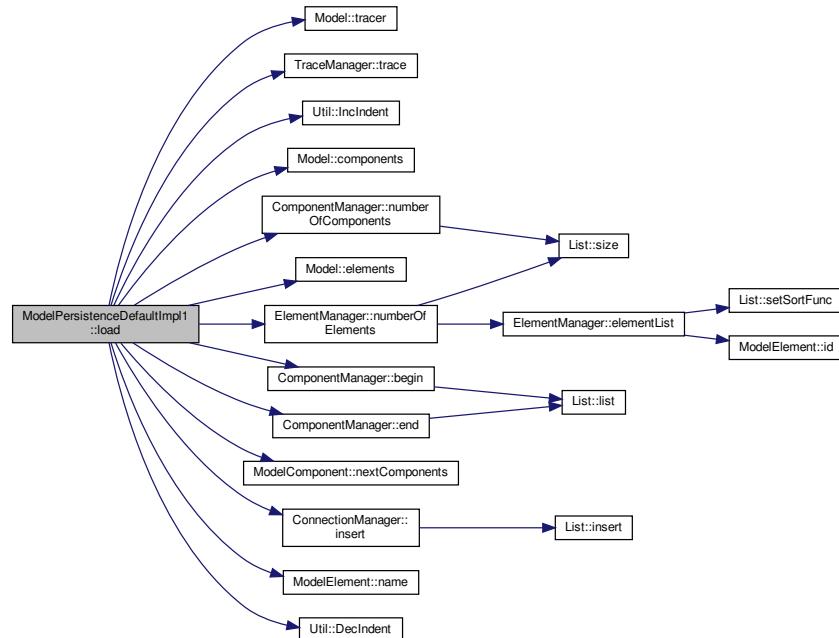
6.75.3.2 load()

```
bool ModelPersistenceDefaultImpl1::load (
    std::string filename ) [virtual]
```

Implements [ModelPersistence_if](#).

Definition at line 184 of file [ModelPersistenceDefaultImpl1.cpp](#).

Here is the call graph for this function:



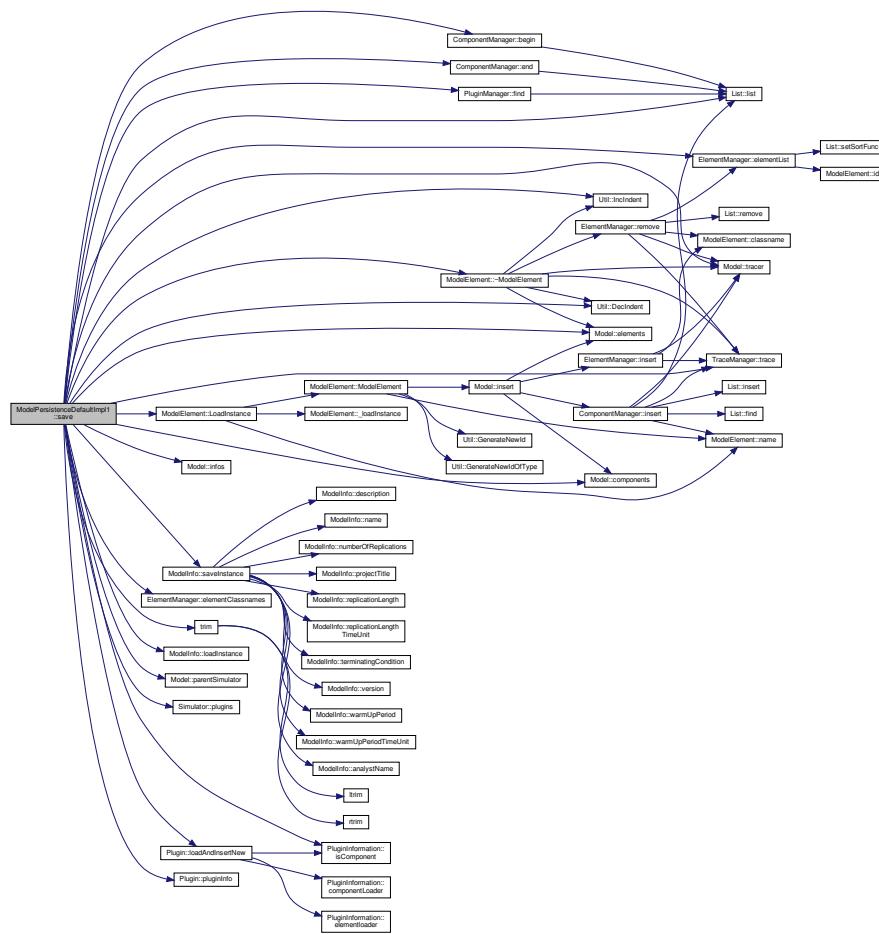
6.75.3.3 save()

```
bool ModelPersistenceDefaultImpl1::save (
    std::string filename ) [virtual]
```

Implements [ModelPersistence_if](#).

Definition at line 34 of file [ModelPersistenceDefaultImpl1.cpp](#).

Here is the call graph for this function:



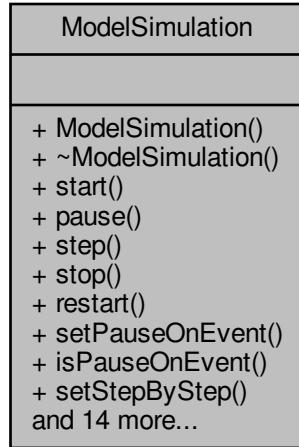
The documentation for this class was generated from the following files:

- [ModelPersistenceDefaultImpl1.h](#)
- [ModelPersistenceDefaultImpl1.cpp](#)

6.76 ModelSimulation Class Reference

```
#include <ModelSimulation.h>
```

Collaboration diagram for ModelSimulation:



Public Member Functions

- `ModelSimulation (Model *model)`
- virtual `~ModelSimulation ()=default`
- void `start ()`

Starts a sequential execution of a simulation, ie, a set of replications of this model.
- void `pause ()`
- void `step ()`

Executes the processing of a single event, the next one in the future events list.
- void `stop ()`
- void `restart ()`
- void `setPauseOnEvent (bool _pauseOnEvent)`
- bool `isPauseOnEvent () const`
- void `setStepByStep (bool _stepByStep)`
- bool `isStepByStep () const`
- void `setInitializeStatistics (bool _initializeStatistics)`
- bool `isInitializeStatistics () const`
- void `setInitializeSystem (bool _initializeSystem)`
- bool `isInitializeSystem () const`
- void `setPauseOnReplication (bool _pauseBetweenReplications)`
- bool `isPauseOnReplication () const`
- double `simulatedTime () const`
- bool `isRunning () const`
- unsigned int `currentReplicationNumber () const`
- `ModelComponent * currentComponent () const`
- `Entity * currentEntity () const`
- unsigned int `currentInputNumber () const`
- `SimulationReporter_if * reporter () const`

6.76.1 Detailed Description

The [ModelSimulation](#) controls the simulation of a model, allowing to start, pause, resume or stop a simulation, composed by a set of replications.

Definition at line 29 of file [ModelSimulation.h](#).

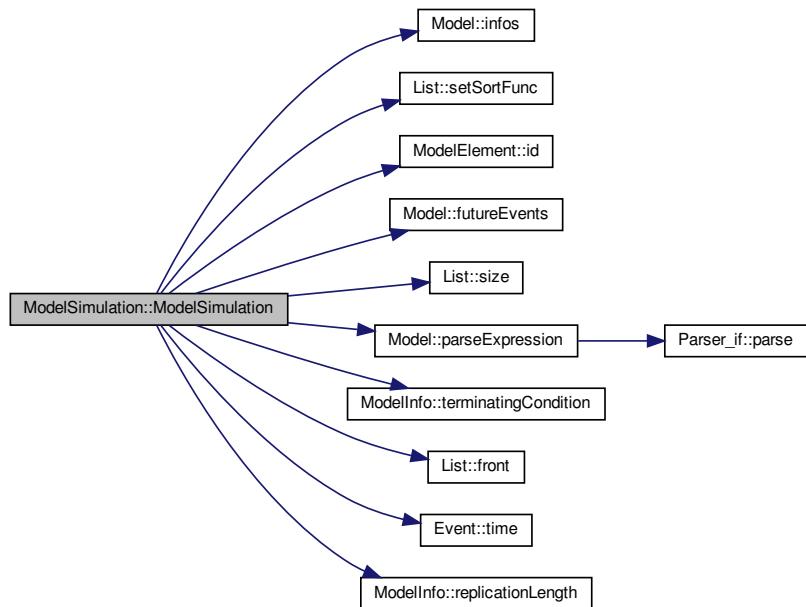
6.76.2 Constructor & Destructor Documentation

6.76.2.1 ModelSimulation()

```
ModelSimulation::ModelSimulation ( Model * model )
```

Definition at line 25 of file [ModelSimulation.cpp](#).

Here is the call graph for this function:



6.76.2.2 ~ModelSimulation()

```
virtual ModelSimulation::~ModelSimulation ( ) [virtual], [default]
```

6.76.3 Member Function Documentation

6.76.3.1 currentComponent()

```
ModelComponent * ModelSimulation::currentComponent ( ) const
```

Definition at line 412 of file [ModelSimulation.cpp](#).

6.76.3.2 currentEntity()

```
Entity * ModelSimulation::currentEntity ( ) const
```

Definition at line 416 of file [ModelSimulation.cpp](#).

6.76.3.3 currentInputNumber()

```
unsigned int ModelSimulation::currentInputNumber ( ) const
```

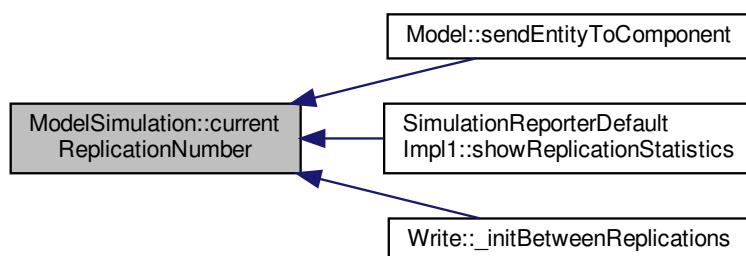
Definition at line 424 of file [ModelSimulation.cpp](#).

6.76.3.4 currentReplicationNumber()

```
unsigned int ModelSimulation::currentReplicationNumber ( ) const
```

Definition at line 408 of file [ModelSimulation.cpp](#).

Here is the caller graph for this function:



6.76.3.5 isInitializeStatistics()

```
bool ModelSimulation::isInitializeStatistics () const
```

Definition at line 372 of file [ModelSimulation.cpp](#).

6.76.3.6 isInitializeSystem()

```
bool ModelSimulation::isInitializeSystem () const
```

Definition at line 380 of file [ModelSimulation.cpp](#).

6.76.3.7 isPauseOnEvent()

```
bool ModelSimulation::isPauseOnEvent () const
```

Definition at line 364 of file [ModelSimulation.cpp](#).

6.76.3.8 isPauseOnReplication()

```
bool ModelSimulation::isPauseOnReplication () const
```

Definition at line 396 of file [ModelSimulation.cpp](#).

6.76.3.9 isRunning()

```
bool ModelSimulation::isRunning () const
```

The current time in the model being simulated, i.e., the instant when the current event was triggered

Definition at line 404 of file [ModelSimulation.cpp](#).

6.76.3.10 isStepByStep()

```
bool ModelSimulation::isStepByStep () const
```

Definition at line 388 of file [ModelSimulation.cpp](#).

6.76.3.11 pause()

```
void ModelSimulation::pause ( )
```

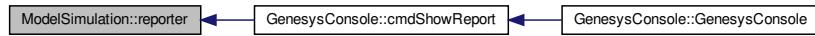
Definition at line 338 of file [ModelSimulation.cpp](#).

6.76.3.12 reporter()

```
SimulationReporter_if * ModelSimulation::reporter ( ) const
```

Definition at line 420 of file [ModelSimulation.cpp](#).

Here is the caller graph for this function:



6.76.3.13 restart()

```
void ModelSimulation::restart ( )
```

Definition at line 357 of file [ModelSimulation.cpp](#).

6.76.3.14 setInitializeStatistics()

```
void ModelSimulation::setInitializeStatistics (
    bool _initializeStatistics )
```

Definition at line 368 of file [ModelSimulation.cpp](#).

6.76.3.15 setInitializeSystem()

```
void ModelSimulation::setInitializeSystem (
    bool _initializeSystem )
```

Definition at line 376 of file [ModelSimulation.cpp](#).

6.76.3.16 setPauseOnEvent()

```
void ModelSimulation::setPauseOnEvent (
    bool _pauseOnEvent )
```

Definition at line 360 of file [ModelSimulation.cpp](#).

6.76.3.17 setPauseOnReplication()

```
void ModelSimulation::setPauseOnReplication (
    bool _pauseBetweenReplications )
```

Definition at line 392 of file [ModelSimulation.cpp](#).

6.76.3.18 setStepByStep()

```
void ModelSimulation::setStepByStep (
    bool _stepByStep )
```

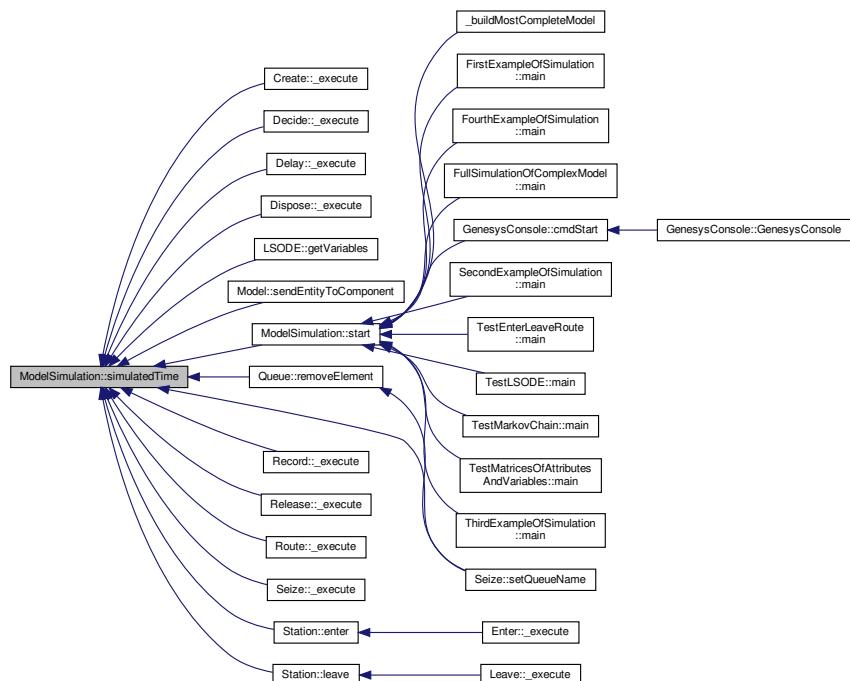
Definition at line 384 of file [ModelSimulation.cpp](#).

6.76.3.19 simulatedTime()

```
double ModelSimulation::simulatedTime ( ) const
```

Definition at line 400 of file [ModelSimulation.cpp](#).

Here is the caller graph for this function:



6.76.3.20 start()

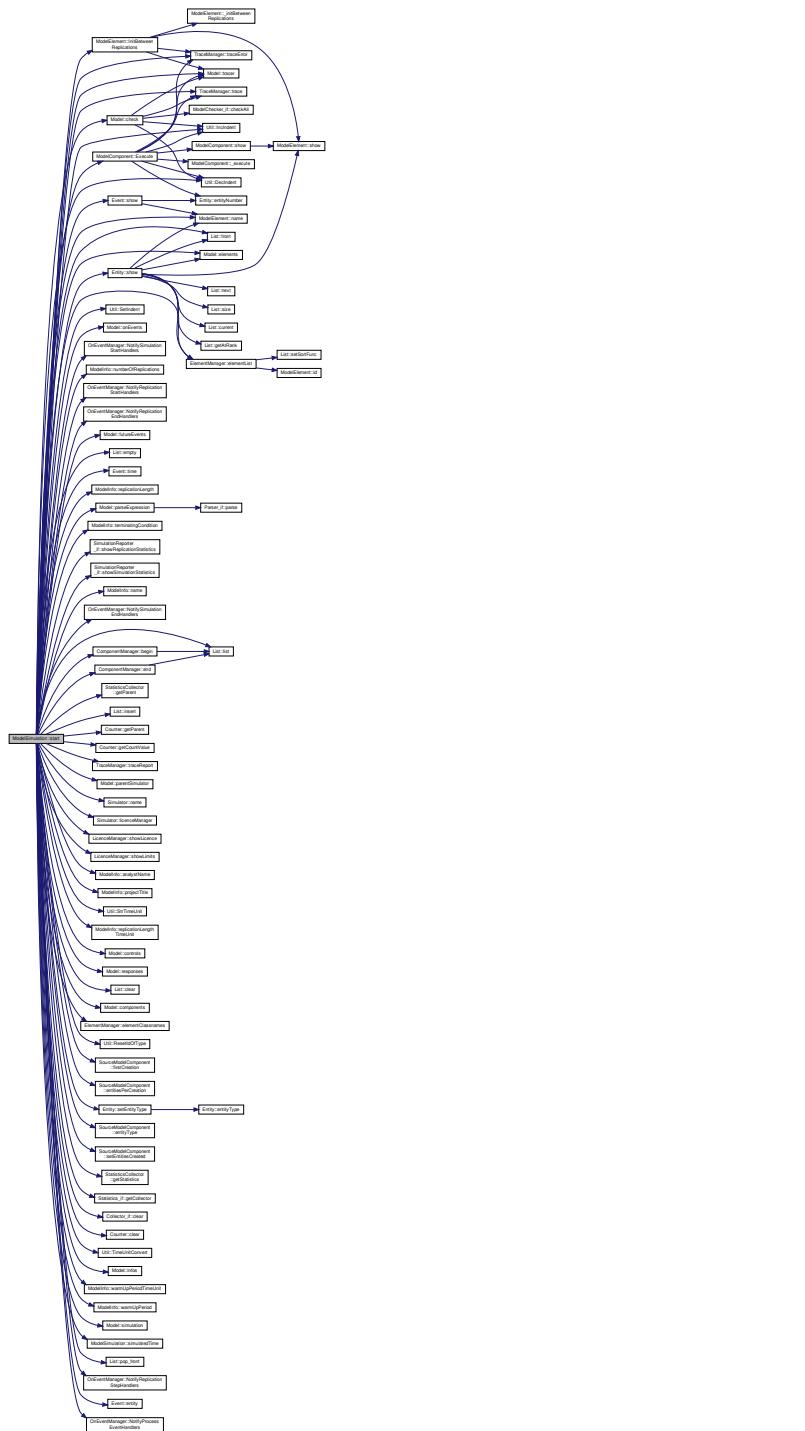
```
void ModelSimulation::start ()
```

Starts a sequential execution of a simulation, ie, a set of replications of this model.

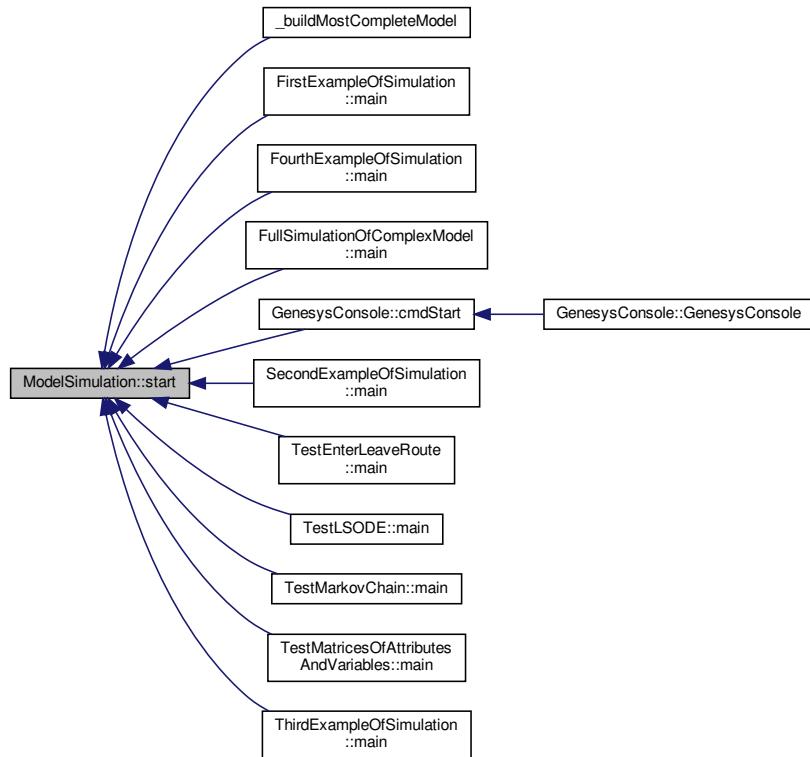
Checks the model and if ok then initialize the simulation, execute repeatedly each replication and then show simulation statistics

Definition at line 47 of file [ModelSimulation.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



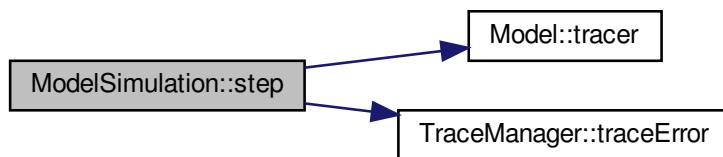
6.76.3.21 step()

```
void ModelSimulation::step( )
```

Executes the processing of a single event, the next one in the future events list.

Definition at line 341 of file [ModelSimulation.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

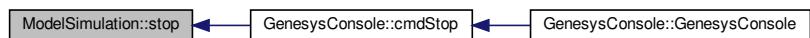


6.76.3.22 stop()

```
void ModelSimulation::stop ( )
```

Definition at line 354 of file [ModelSimulation.cpp](#).

Here is the caller graph for this function:



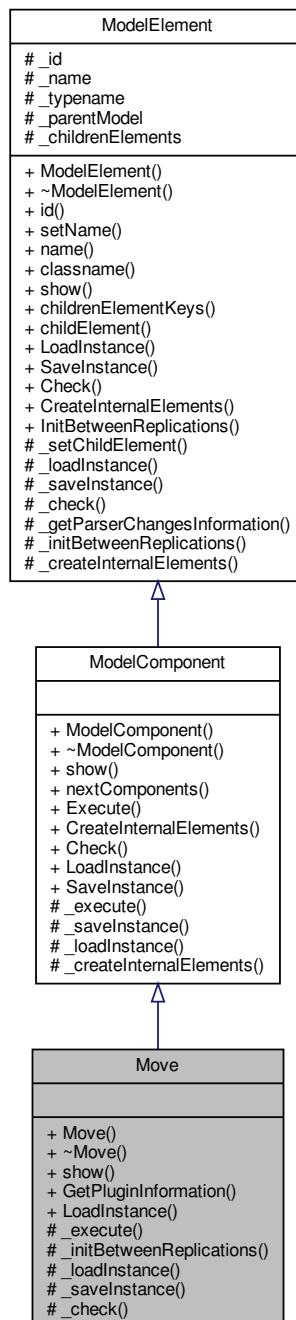
The documentation for this class was generated from the following files:

- [ModelSimulation.h](#)
- [ModelSimulation.cpp](#)

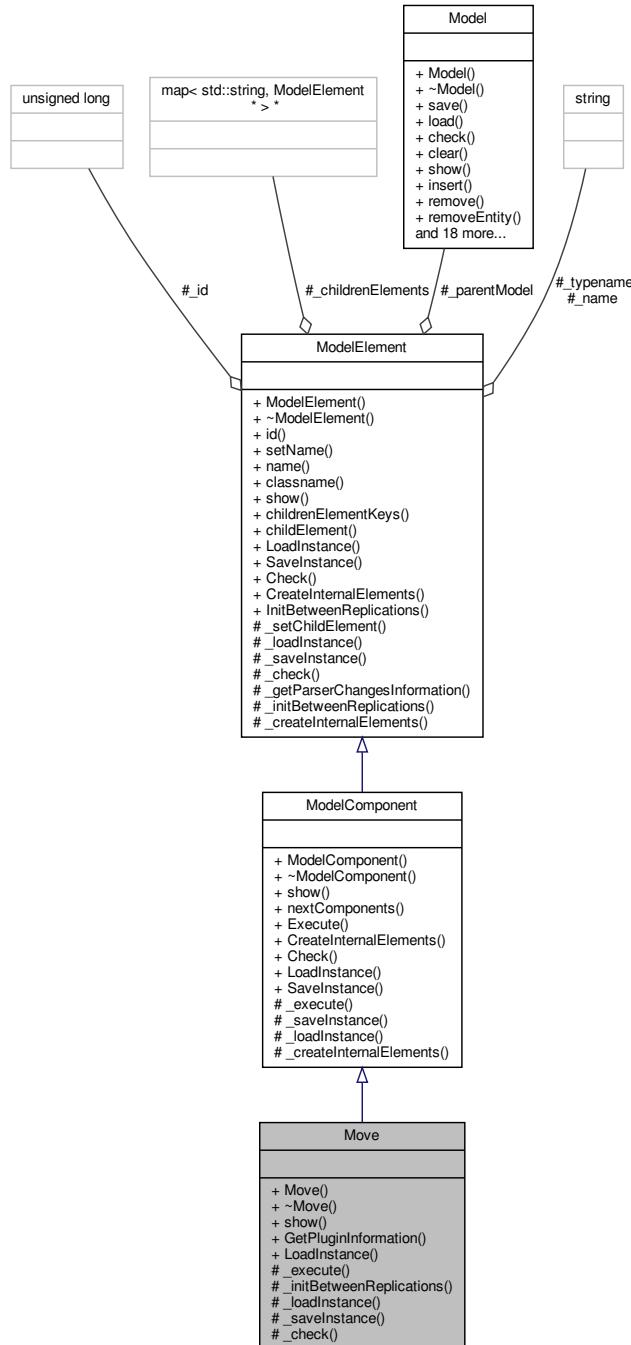
6.77 Move Class Reference

```
#include <Move.h>
```

Inheritance diagram for Move:



Collaboration diagram for Move:



Public Member Functions

- `Move (Model *model, std::string name="")`
- virtual `~Move ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static [PluginInformation * GetPluginInformation \(\)](#)
- static [ModelComponent * LoadInstance \(Model *model, std::map< std::string, std::string > *fields\)](#)

Protected Member Functions

- virtual void [_execute \(Entity *entity\)](#)
- virtual void [_initBetweenReplications \(\)](#)
- virtual bool [_loadInstance \(std::map< std::string, std::string > *fields\)](#)
- virtual std::map< std::string, std::string > * [_saveInstance \(\)](#)
- virtual bool [_check \(std::string *errorMessage\)](#)

Additional Inherited Members

6.77.1 Detailed Description

Move module DESCRIPTION The **Move** module advances a transporter from one location to another without moving the controlling entity to the destination station. The controlling entity remains at its current module location until the transporter arrives at its destination. At that time, the entity will be able to move to another module or task in the model. If the transporter being moved is a free-path transporter, then the time delay to move the transporter from one station to the next is based on the velocity of the transporter and the distance between the stations (specified in the Distance module). If the transporter being moved is a guided transporter, then the time delay to move the transporter depends not only on the velocity and distance to the destination, but also the transporter's acceleration, deceleration, and any traffic it encounters on the way TYPICAL USES **Move** a broken forklift to a service station **Move** a worker to a break room **Move** a waiter to the kitchen PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Transporter Name Name of the transporter to move. Unit Number Determines which of the transporter units in the transporter set to move. Destination Type Determines the method for specifying the transporter destination. The Intersection and Network Link options apply only to guided transporters. Station Name Name of the individual destination station. Attribute Name Name of the attribute that stores the destination station name to which entities will route. Expression Expression that is evaluated to the destination station name where entities will route. Intersection Name Defines the name of the intersection to which the guided transporter will move. Network Link Name Defines the name of the network link to which the guided transporter will move. Zone The specific zone number of the Network Link Name. Velocity Specifies the temporary velocity at which the transporter is moved to the destination station. Units Velocity time units.

Definition at line 60 of file [Move.h](#).

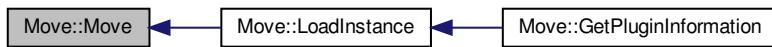
6.77.2 Constructor & Destructor Documentation

6.77.2.1 Move()

```
Move::Move (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Move.cpp](#).

Here is the caller graph for this function:



6.77.2.2 ~Move()

```
virtual Move::~Move ( ) [virtual], [default]
```

6.77.3 Member Function Documentation

6.77.3.1 _check()

```
bool Move::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Move.cpp](#).

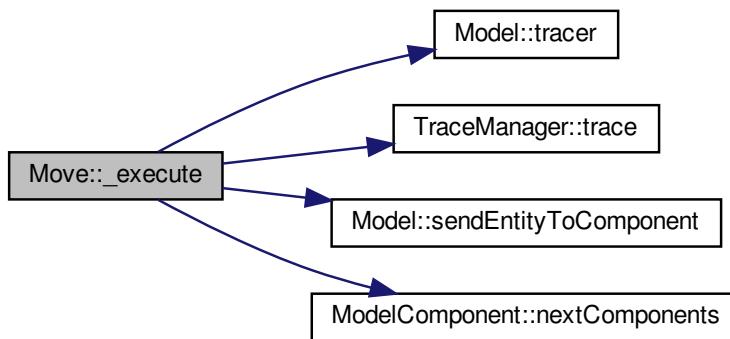
6.77.3.2 `_execute()`

```
void Move::_execute (
    Entity * entity) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Move.cpp](#).

Here is the call graph for this function:



6.77.3.3 `_initBetweenReplications()`

```
void Move::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Move.cpp](#).

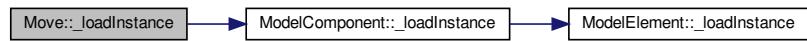
6.77.3.4 `_loadInstance()`

```
bool Move::_loadInstance (
    std::map< std::string, std::string > * fields) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Move.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



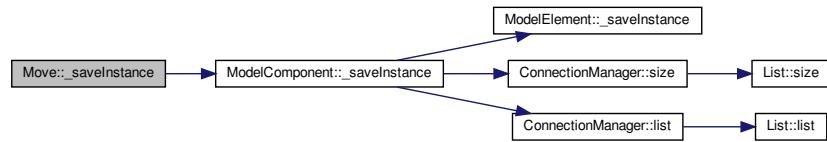
6.77.3.5 _saveInstance()

```
std::map< std::string, std::string > * Move::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [52](#) of file [Move.cpp](#).

Here is the call graph for this function:



6.77.3.6 GetPluginInformation()

```
PluginInformation * Move::GetPluginInformation ( ) [static]
```

Definition at line [64](#) of file [Move.cpp](#).

Here is the call graph for this function:

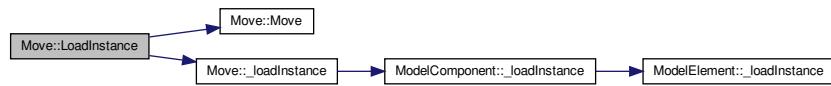


6.77.3.7 LoadInstance()

```
ModelComponent * Move::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 26 of file [Move.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



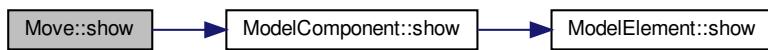
6.77.3.8 show()

```
std::string Move::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Move.cpp](#).

Here is the call graph for this function:



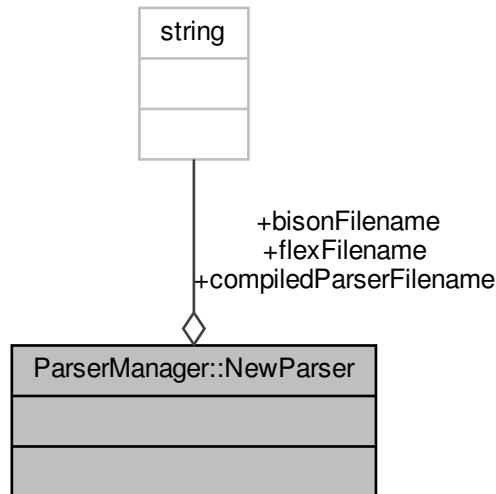
The documentation for this class was generated from the following files:

- [Move.h](#)
- [Move.cpp](#)

6.78 ParserManager::NewParser Struct Reference

```
#include <ParserManager.h>
```

Collaboration diagram for ParserManager::NewParser:



Public Attributes

- `std::string bisonFilename`
- `std::string flexFilename`
- `std::string compiledParserFilename`

6.78.1 Detailed Description

Definition at line [22](#) of file [ParserManager.h](#).

6.78.2 Member Data Documentation

6.78.2.1 `bisonFilename`

```
std::string ParserManager::NewParser::bisonFilename
```

Definition at line [23](#) of file [ParserManager.h](#).

6.78.2.2 compiledParserFilename

```
std::string ParserManager::NewParser::compiledParserFilename
```

Definition at line 25 of file [ParserManager.h](#).

6.78.2.3 flexFilename

```
std::string ParserManager::NewParser::flexFilename
```

Definition at line 24 of file [ParserManager.h](#).

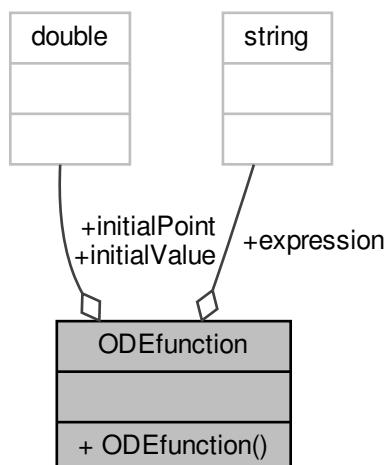
The documentation for this struct was generated from the following file:

- [ParserManager.h](#)

6.79 ODEfunction Class Reference

```
#include <OLD_ODElement.h>
```

Collaboration diagram for ODEfunction:



Public Member Functions

- `ODEfunction` (`std::string expression`, `double initialPoint`, `double initialValue`)

Public Attributes

- std::string [expression](#)
- double [initialPoint](#)
- double [initialValue](#)

6.79.1 Detailed Description

Definition at line [21](#) of file [OLD_ODElement.h](#).

6.79.2 Constructor & Destructor Documentation

6.79.2.1 ODEfunction()

```
ODEfunction::ODEfunction (
    std::string expression,
    double initialPoint,
    double initialValue ) [inline]
```

Definition at line [24](#) of file [OLD_ODElement.h](#).

6.79.3 Member Data Documentation

6.79.3.1 expression

```
std::string ODEfunction::expression
```

Definition at line [29](#) of file [OLD_ODElement.h](#).

6.79.3.2 initialPoint

```
double ODEfunction::initialPoint
```

Definition at line [30](#) of file [OLD_ODElement.h](#).

6.79.3.3 initialValue

```
double ODEfunction::initialValue
```

Definition at line [31](#) of file [OLD_ODElement.h](#).

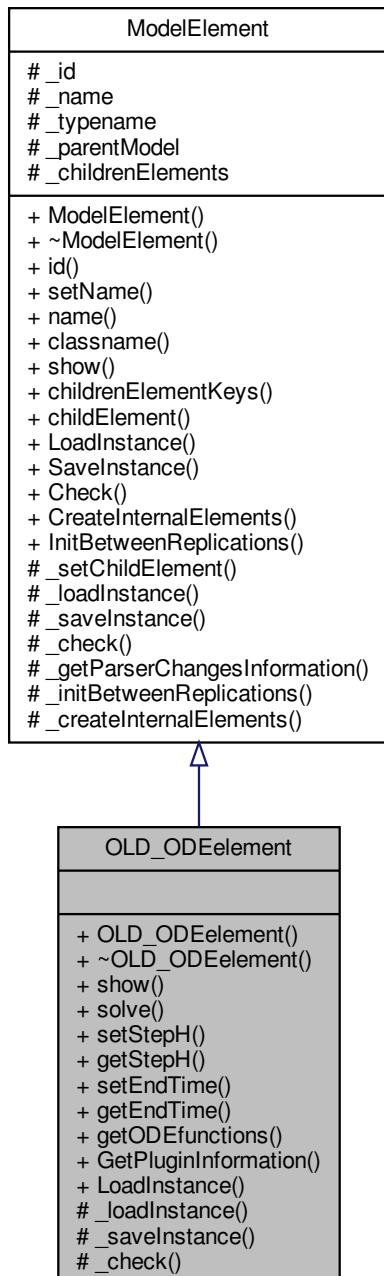
The documentation for this class was generated from the following file:

- [OLD_ODElement.h](#)

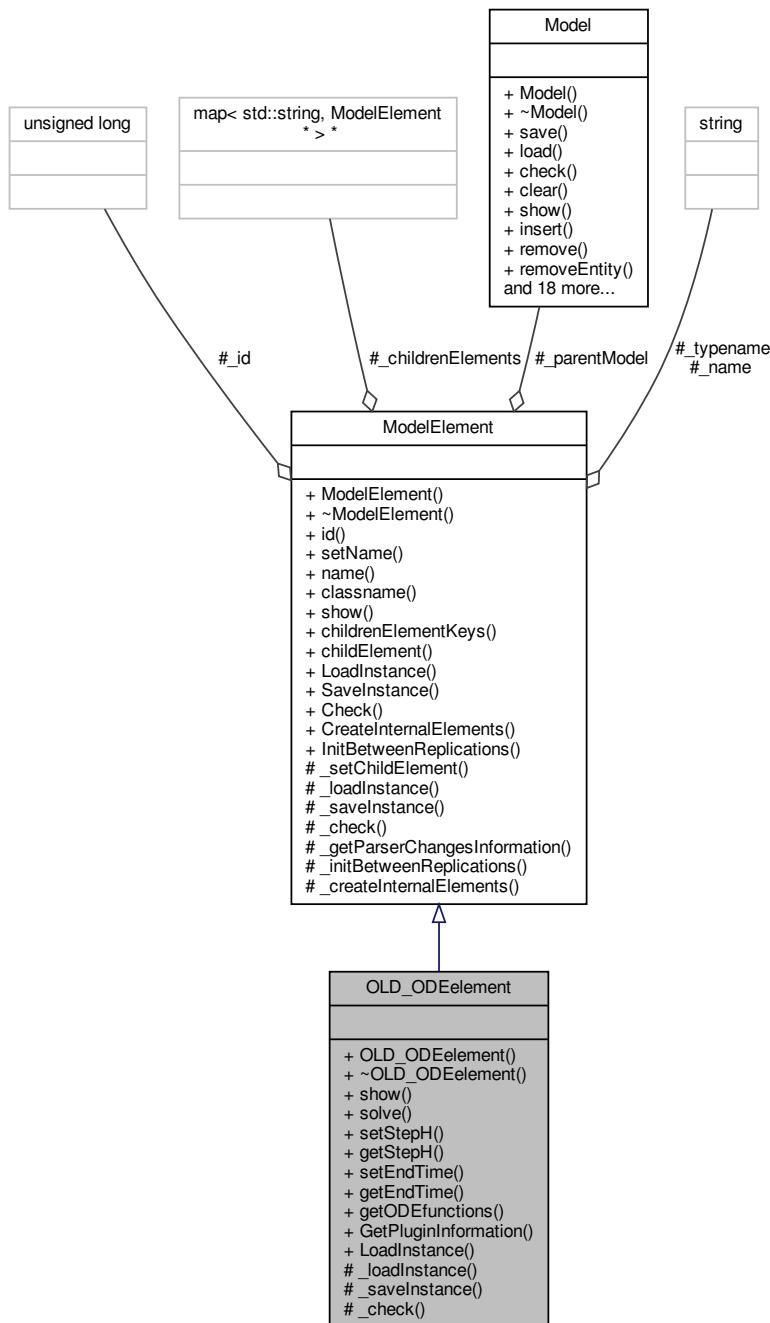
6.80 OLD_ODElement Class Reference

```
#include <OLD_ODElement.h>
```

Inheritance diagram for OLD_ODElement:



Collaboration diagram for OLD_ODElement:



Public Member Functions

- `OLD_ODElement (Model *model, std::string name="")`
- `virtual ~OLD_ODElement ()=default`
- `virtual std::string show ()`
- `double solve ()`
- `void setStepH (double _h)`

- double `getStepH () const`
- void `setEndTime (double _endTime)`
- double `getEndTime () const`
- `List< ODEfunction * > * getODEfunctions () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.80.1 Detailed Description

Definition at line 34 of file `OLD_ODElement.h`.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 OLD_ODElement()

```
OLD_ODElement::OLD_ODElement (
    Model * model,
    std::string name = "")
```

Definition at line 17 of file `OLD_ODElement.cpp`.

Here is the caller graph for this function:



6.80.2.2 ~OLD_ODElement()

```
virtual OLD_ODElement::~OLD_ODElement ( ) [virtual], [default]
```

6.80.3 Member Function Documentation

6.80.3.1 `_check()`

```
bool OLD_ODElement::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 80 of file [OLD_ODElement.cpp](#).

Here is the call graph for this function:



6.80.3.2 `_loadInstance()`

```
bool OLD_ODElement::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 72 of file [OLD_ODElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



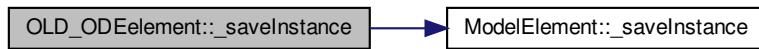
6.80.3.3 `_saveInstance()`

```
std::map< std::string, std::string > * OLD_ODElement::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 76 of file [OLD_ODElement.cpp](#).

Here is the call graph for this function:



6.80.3.4 `getEndTime()`

```
double OLD_ODElement::getEndTime ( ) const
```

Definition at line 49 of file [OLD_ODElement.cpp](#).

6.80.3.5 `getODEfunctions()`

```
List< ODEfunction * > * OLD_ODElement::getODEfunctions ( ) const
```

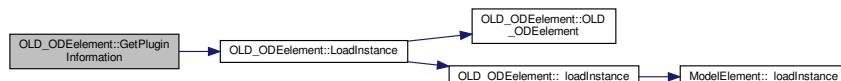
Definition at line 53 of file [OLD_ODElement.cpp](#).

6.80.3.6 `GetPluginInformation()`

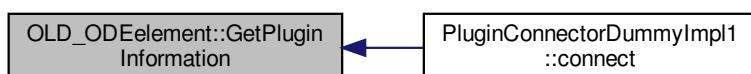
```
PluginInformation * OLD_ODElement::GetPluginInformation ( ) [static]
```

Definition at line 57 of file [OLD_ODElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.80.3.7 getStepH()

```
double OLD_ODElement::getStepH ( ) const
```

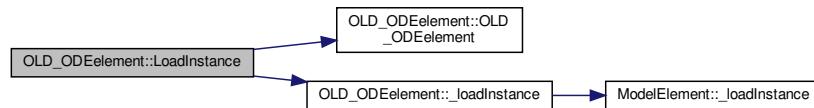
Definition at line 41 of file [OLD_ODElement.cpp](#).

6.80.3.8 LoadInstance()

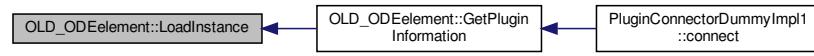
```
ModelElement * OLD_ODElement::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 62 of file [OLD_ODElement.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.80.3.9 setEndTime()

```
void OLD_ODElement::setEndTime (
    double _endTime )
```

Definition at line 45 of file [OLD_ODElement.cpp](#).

6.80.3.10 setStepH()

```
void OLD_ODElement::setStepH (
    double _h )
```

Definition at line 37 of file [OLD_ODElement.cpp](#).

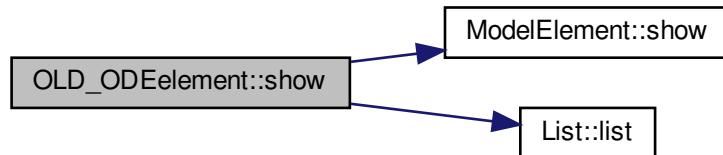
6.80.3.11 show()

```
std::string OLD_ODElement::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 22 of file [OLD_ODElement.cpp](#).

Here is the call graph for this function:



6.80.3.12 solve()

```
double OLD_ODElement::solve ( )
```

Definition at line 31 of file [OLD_ODElement.cpp](#).

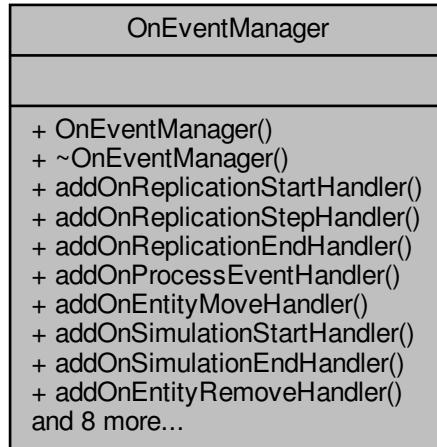
The documentation for this class was generated from the following files:

- [OLD_ODElement.h](#)
- [OLD_ODElement.cpp](#)

6.81 OnEventManager Class Reference

```
#include <OnEventManager.h>
```

Collaboration diagram for OnEventManager:



Public Member Functions

- `OnEventManager ()`
- virtual `~OnEventManager ()=default`
- void `addOnReplicationStartHandler (simulationEventHandler EventHandler)`
- void `addOnReplicationStepHandler (simulationEventHandler EventHandler)`
- void `addOnReplicationEndHandler (simulationEventHandler EventHandler)`
- void `addOnProcessEventHandler (simulationEventHandler EventHandler)`
- void `addOnEntityMoveHandler (simulationEventHandler EventHandler)`
- void `addOnSimulationStartHandler (simulationEventHandler EventHandler)`
- void `addOnSimulationEndHandler (simulationEventHandler EventHandler)`
- void `addOnEntityRemoveHandler (simulationEventHandler EventHandler)`
- template<typename Class >
 `void addOnProcessEventHandler (Class *object, void(Class::*function)(SimulationEvent *))`
- void `NotifyReplicationStartHandlers (SimulationEvent *se)`
- void `NotifyReplicationStepHandlers (SimulationEvent *se)`
- void `NotifyReplicationEndHandlers (SimulationEvent *se)`
- void `NotifyProcessEventHandlers (SimulationEvent *se)`
- void `NotifyEntityMoveHandlers (SimulationEvent *se)`
- void `NotifySimulationStartHandlers (SimulationEvent *se)`
- void `NotifySimulationEndHandlers (SimulationEvent *se)`

6.81.1 Detailed Description

`OnEventManager` allows external methods to hook interval simulation events as listeners (or observers) of specific events. All methods added as listeners of an event will be invoked when that event is triggered.

Definition at line 56 of file `OnEventManager.h`.

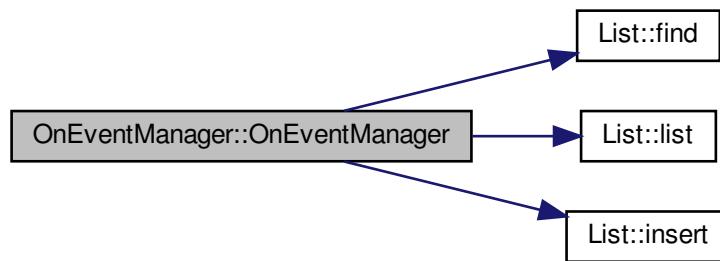
6.81.2 Constructor & Destructor Documentation

6.81.2.1 OnEventManager()

```
OnEventManager::OnEventManager ( )
```

Definition at line 16 of file [OnEventManager.cpp](#).

Here is the call graph for this function:



6.81.2.2 ~OnEventManager()

```
virtual OnEventManager::~OnEventManager ( ) [virtual], [default]
```

6.81.3 Member Function Documentation

6.81.3.1 addOnEntityMoveHandler()

```
void OnEventManager::addOnEntityMoveHandler ( simulationEventHandler EventHandler )
```

Definition at line 36 of file [OnEventManager.cpp](#).

6.81.3.2 addOnEntityRemoveHandler()

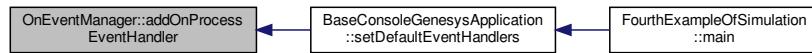
```
void OnEventManager::addOnEntityRemoveHandler ( simulationEventHandler EventHandler )
```

6.81.3.3 addOnProcessEventHandler() [1/2]

```
void OnEventManager::addOnProcessEventHandler (
    simulationEventHandler EventHandler )
```

Definition at line 32 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:



6.81.3.4 addOnProcessEventHandler() [2/2]

```
template<typename Class >
void OnEventManager::addOnProcessEventHandler (
    Class * object,
    void(Class::*)(SimulationEvent *) function )
```

Definition at line 99 of file [OnEventManager.h](#).

6.81.3.5 addOnReplicationEndHandler()

```
void OnEventManager::addOnReplicationEndHandler (
    simulationEventHandler EventHandler )
```

Definition at line 40 of file [OnEventManager.cpp](#).

6.81.3.6 addOnReplicationStartHandler()

```
void OnEventManager::addOnReplicationStartHandler (
    simulationEventHandler EventHandler )
```

Definition at line 24 of file [OnEventManager.cpp](#).

6.81.3.7 addOnReplicationStepHandler()

```
void OnEventManager::addOnReplicationStepHandler (
    simulationEventHandler EventHandler )
```

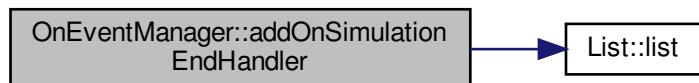
Definition at line 28 of file [OnEventManager.cpp](#).

6.81.3.8 addOnSimulationEndHandler()

```
void OnEventManager::addOnSimulationEndHandler (  
    simulationEventHandler EventHandler )
```

Definition at line 48 of file [OnEventManager.cpp](#).

Here is the call graph for this function:



6.81.3.9 addOnSimulationStartHandler()

```
void OnEventManager::addOnSimulationStartHandler (  
    simulationEventHandler EventHandler )
```

Definition at line 44 of file [OnEventManager.cpp](#).

6.81.3.10 NotifyEntityMoveHandlers()

```
void OnEventManager::NotifyEntityMoveHandlers (  
    SimulationEvent * se )
```

Definition at line 77 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:

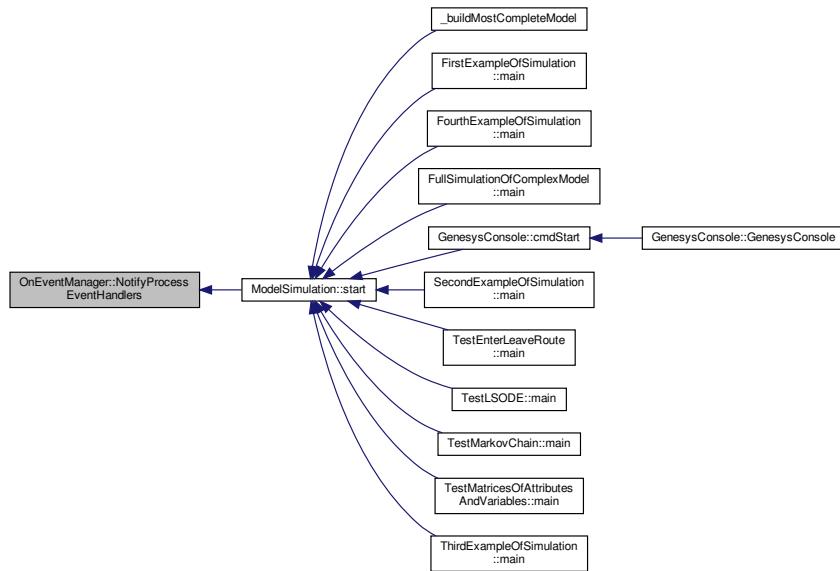


6.81.3.11 NotifyProcessEventHandlers()

```
void OnEventManager::NotifyProcessEventHandlers (
    SimulationEvent * se )
```

Definition at line 81 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:

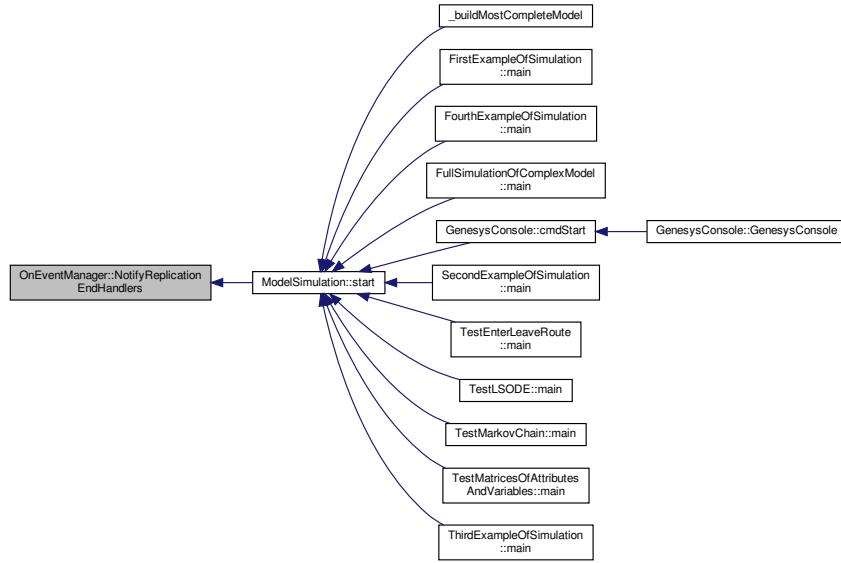


6.81.3.12 NotifyReplicationEndHandlers()

```
void OnEventManager::NotifyReplicationEndHandlers (
    SimulationEvent * se )
```

Definition at line 73 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:

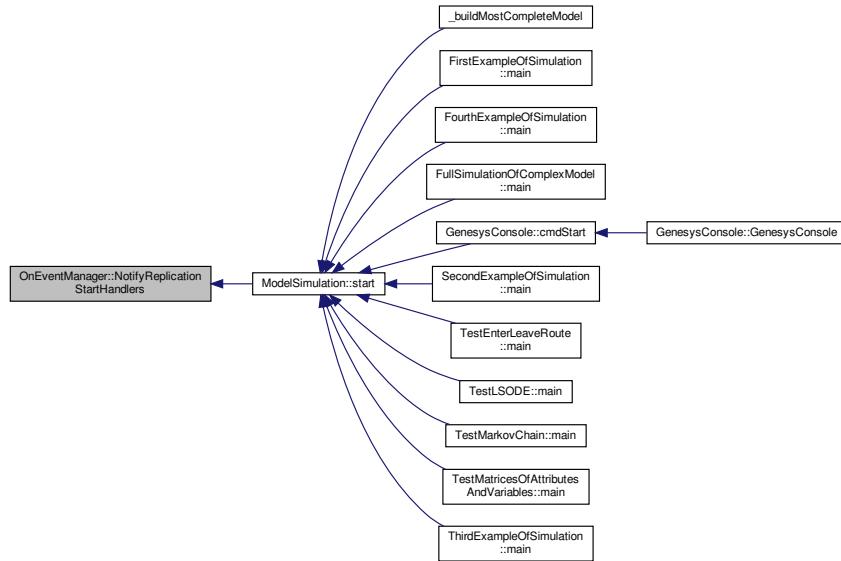


6.81.3.13 NotifyReplicationStartHandlers()

```
void OnEventManager::NotifyReplicationStartHandlers (
    SimulationEvent * se )
```

Definition at line 64 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:

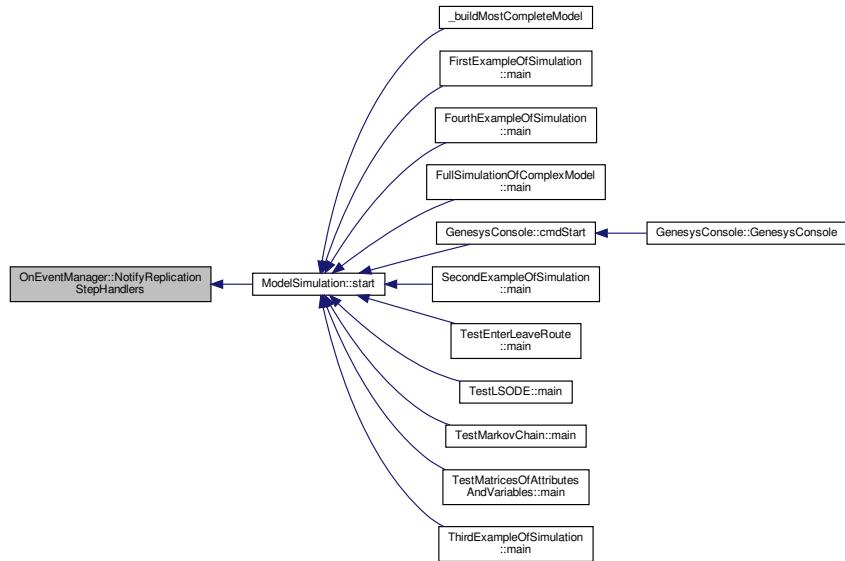


6.81.3.14 NotifyReplicationStepHandlers()

```
void OnEventManager::NotifyReplicationStepHandlers (  
    SimulationEvent * se )
```

Definition at line 69 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:

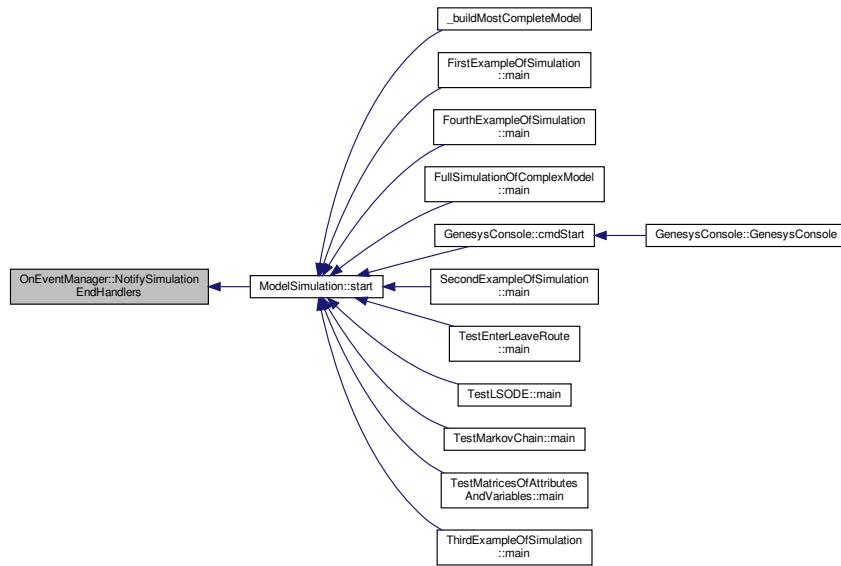


6.81.3.15 NotifySimulationEndHandlers()

```
void OnEventManager::NotifySimulationEndHandlers (   
    SimulationEvent * se )
```

Definition at line 90 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:

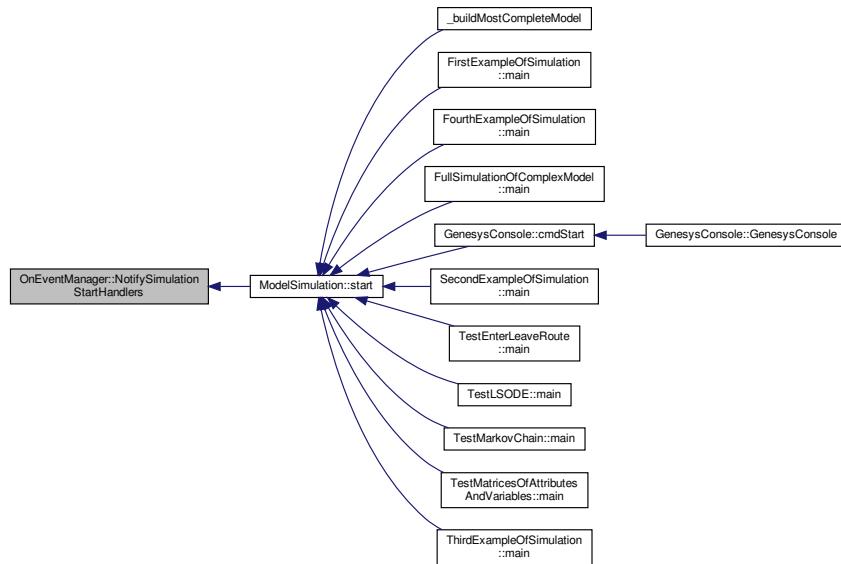


6.81.3.16 NotifySimulationStartHandlers()

```
void OnEventManager::NotifySimulationStartHandlers (
    SimulationEvent * se )
```

Definition at line 86 of file [OnEventManager.cpp](#).

Here is the caller graph for this function:



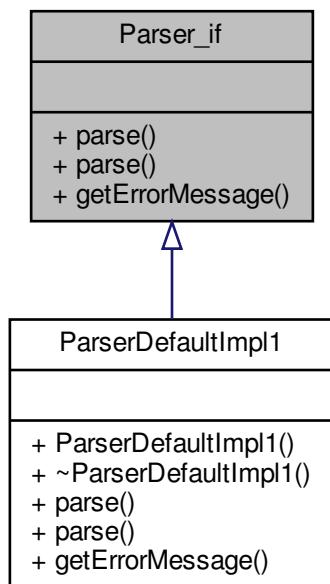
The documentation for this class was generated from the following files:

- [OnEventManager.h](#)
- [OnEventManager.cpp](#)

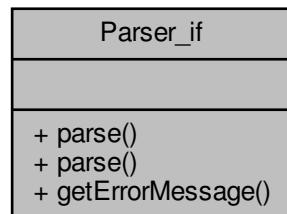
6.82 Parser_if Class Reference

```
#include <Parser_if.h>
```

Inheritance diagram for Parser_if:



Collaboration diagram for Parser_if:



Public Member Functions

- virtual double [parse](#) (const std::string &expression)=0
- virtual double [parse](#) (const std::string &expression, bool *success, std::string *errorMessage)=0
- virtual std::string * [getErrorMessage](#) ()=0

6.82.1 Detailed Description

Definition at line 19 of file [Parser_if.h](#).

6.82.2 Member Function Documentation

6.82.2.1 [getErrorMessage\(\)](#)

```
virtual std::string* Parser_if::getErrorMessage ( ) [pure virtual]
```

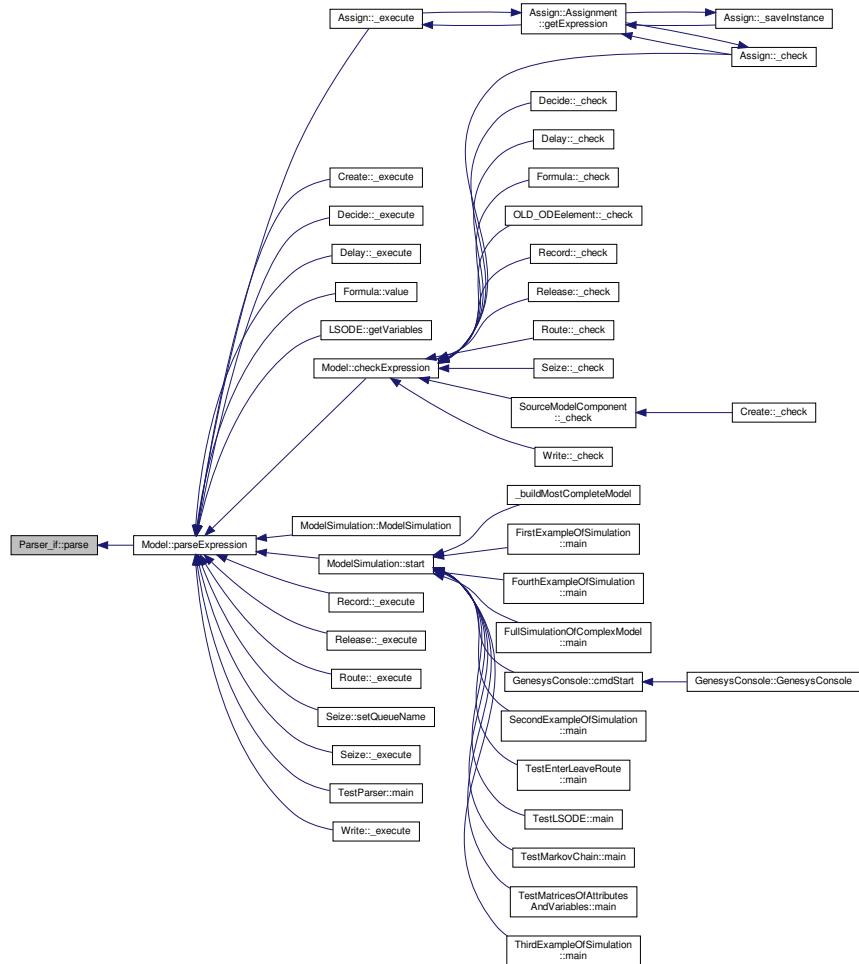
Implemented in [ParserDefaultImpl1](#).

6.82.2.2 [parse\(\)](#) [1/2]

```
virtual double Parser_if::parse (
    const std::string &expression ) [pure virtual]
```

Implemented in [ParserDefaultImpl1](#).

Here is the caller graph for this function:



6.82.2.3 parse() [2/2]

```
virtual double Parser_if::parse (
        const std::string expression,
        bool * success,
        std::string * errorMessage ) [pure virtual]
```

Implemented in ParserDefaultImpl1.

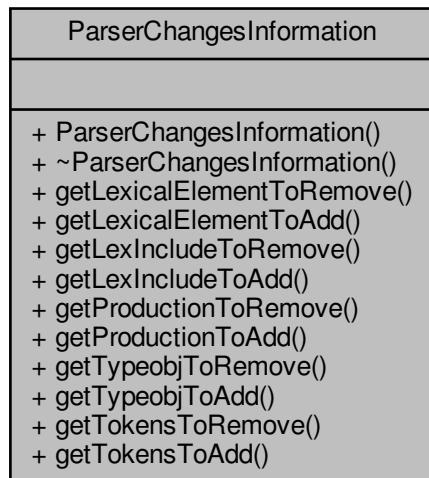
The documentation for this class was generated from the following file:

- Parser_if.h

6.83 ParserChangesInformation Class Reference

```
#include <ParserChangesInformation.h>
```

Collaboration diagram for ParserChangesInformation:



Public Types

- `typedef std::string token`
- `typedef std::string typeobj`
- `typedef std::string lexinclude`
- `typedef std::pair< std::string, std::string > production`
- `typedef std::pair< std::string, std::string > lexicalement`

Public Member Functions

- `ParserChangesInformation ()`
- `virtual ~ParserChangesInformation ()=default`
- `std::list< ParserChangesInformation::lexicalement * > * getLexicalElementToRemove () const`
- `std::list< ParserChangesInformation::lexicalement * > * getLexicalElementToAdd () const`
- `std::list< ParserChangesInformation::lexinclude * > * getLexIncludeToRemove () const`
- `std::list< ParserChangesInformation::lexinclude * > * getLexIncludeToAdd () const`
- `std::list< ParserChangesInformation::production * > * getProductionToRemove () const`
- `std::list< ParserChangesInformation::production * > * getProductionToAdd () const`
- `std::list< ParserChangesInformation::typeobj * > * getTypeobjToRemove () const`
- `std::list< ParserChangesInformation::typeobj * > * getTypeobjToAdd () const`
- `std::list< ParserChangesInformation::token * > * getTokensToRemove () const`
- `std::list< ParserChangesInformation::token * > * getTokensToAdd () const`

6.83.1 Detailed Description

Definition at line 20 of file [ParserChangesInformation.h](#).

6.83.2 Member Typedef Documentation

6.83.2.1 `lexicalelement`

```
typedef std::pair<std::string, std::string> ParserChangesInformation::lexicalelement
```

Definition at line 26 of file [ParserChangesInformation.h](#).

6.83.2.2 `lexinclude`

```
typedef std::string ParserChangesInformation::lexinclude
```

Definition at line 24 of file [ParserChangesInformation.h](#).

6.83.2.3 `production`

```
typedef std::pair<std::string, std::string> ParserChangesInformation::production
```

Definition at line 25 of file [ParserChangesInformation.h](#).

6.83.2.4 `token`

```
typedef std::string ParserChangesInformation::token
```

Definition at line 22 of file [ParserChangesInformation.h](#).

6.83.2.5 `typeobj`

```
typedef std::string ParserChangesInformation::typeobj
```

Definition at line 23 of file [ParserChangesInformation.h](#).

6.83.3 Constructor & Destructor Documentation

6.83.3.1 ParserChangesInformation()

```
ParserChangesInformation::ParserChangesInformation ( )
```

Definition at line 16 of file [ParserChangesInformation.cpp](#).

6.83.3.2 ~ParserChangesInformation()

```
virtual ParserChangesInformation::~ParserChangesInformation ( ) [virtual], [default]
```

6.83.4 Member Function Documentation

6.83.4.1 getLexicalElementToAdd()

```
std::list< ParserChangesInformation::lexicalelement * > * ParserChangesInformation::getLexicalElementToAdd ( ) const
```

Definition at line 24 of file [ParserChangesInformation.cpp](#).

6.83.4.2 getLexicalElementToRemove()

```
std::list< ParserChangesInformation::ParserChangesInformation::lexicalelement * > * ParserChangesInformation::getLexicalElementToRemove ( ) const
```

Definition at line 20 of file [ParserChangesInformation.cpp](#).

6.83.4.3 getLexIncludeToAdd()

```
std::list< ParserChangesInformation::lexinclude * > * ParserChangesInformation::getLexIncludeToAdd ( ) const
```

Definition at line 32 of file [ParserChangesInformation.cpp](#).

6.83.4.4 getLexIncludeToRemove()

```
std::list< ParserChangesInformation::lexinclude * > * ParserChangesInformation::getLexIncludeToRemove ( ) const
```

Definition at line 28 of file [ParserChangesInformation.cpp](#).

6.83.4.5 `getProductionToAdd()`

```
std::list< ParserChangesInformation::production * > * ParserChangesInformation::getProductionToAdd ( ) const
```

Definition at line [40](#) of file [ParserChangesInformation.cpp](#).

6.83.4.6 `getProductionToRemove()`

```
std::list< ParserChangesInformation::production * > * ParserChangesInformation::getProductionToRemove ( ) const
```

Definition at line [36](#) of file [ParserChangesInformation.cpp](#).

6.83.4.7 `getTokensToAdd()`

```
std::list< ParserChangesInformation::token * > * ParserChangesInformation::getTokensToAdd ( ) const
```

Definition at line [56](#) of file [ParserChangesInformation.cpp](#).

6.83.4.8 `getTokensToRemove()`

```
std::list< ParserChangesInformation::token * > * ParserChangesInformation::getTokensToRemove ( ) const
```

Definition at line [52](#) of file [ParserChangesInformation.cpp](#).

6.83.4.9 `getTypeobjToAdd()`

```
std::list< ParserChangesInformation::typeobj * > * ParserChangesInformation::getTypeobjToAdd ( ) const
```

Definition at line [48](#) of file [ParserChangesInformation.cpp](#).

6.83.4.10 `getTypeobjToRemove()`

```
std::list< ParserChangesInformation::typeobj * > * ParserChangesInformation::getTypeobjToRemove ( ) const
```

Definition at line [44](#) of file [ParserChangesInformation.cpp](#).

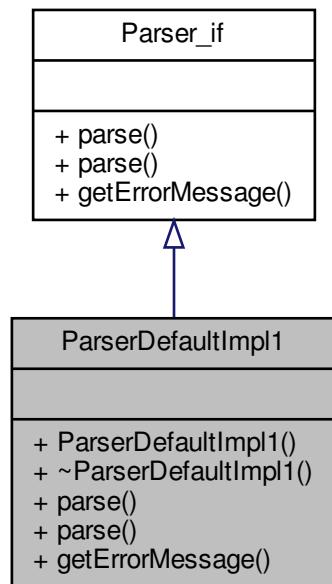
The documentation for this class was generated from the following files:

- [ParserChangesInformation.h](#)
- [ParserChangesInformation.cpp](#)

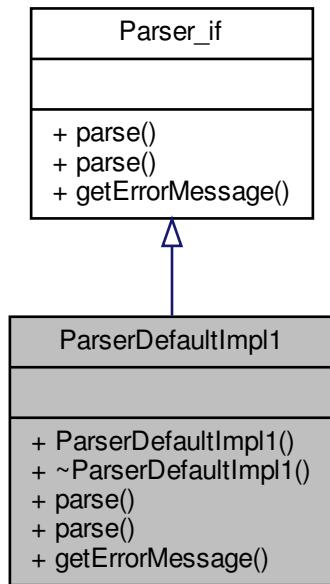
6.84 ParserDefaultImpl1 Class Reference

```
#include <ParserDefaultImpl1.h>
```

Inheritance diagram for ParserDefaultImpl1:



Collaboration diagram for ParserDefaultImpl1:



Public Member Functions

- `ParserDefaultImpl1 (Model *model)`
- virtual `~ParserDefaultImpl1 ()=default`
- `double parse (const std::string expression)`
- `double parse (const std::string expression, bool *success, std::string *errorMessage)`
- `std::string * getErrorMessage ()`

6.84.1 Detailed Description

Definition at line 21 of file [ParserDefaultImpl1.h](#).

6.84.2 Constructor & Destructor Documentation

6.84.2.1 ParserDefaultImpl1()

```
ParserDefaultImpl1::ParserDefaultImpl1 (
    Model * model )
```

Definition at line 16 of file [ParserDefaultImpl1.cpp](#).

6.84.2.2 ~ParserDefaultImpl1()

```
virtual ParserDefaultImpl1::~ParserDefaultImpl1 ( ) [virtual], [default]
```

6.84.3 Member Function Documentation

6.84.3.1 getErrorMessage()

```
std::string * ParserDefaultImpl1::getErrorMessage ( ) [virtual]
```

Implements [Parser_if](#).

Definition at line [26](#) of file [ParserDefaultImpl1.cpp](#).

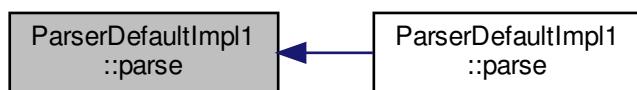
6.84.3.2 parse() [1/2]

```
double ParserDefaultImpl1::parse (
    const std::string expression ) [virtual]
```

Implements [Parser_if](#).

Definition at line [21](#) of file [ParserDefaultImpl1.cpp](#).

Here is the caller graph for this function:



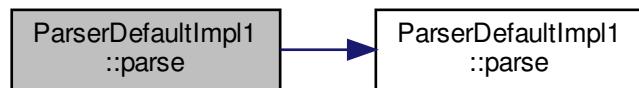
6.84.3.3 `parse()` [2/2]

```
double ParserDefaultImpl1::parse (
    const std::string expression,
    bool * success,
    std::string * errorMessage ) [virtual]
```

Implements [Parser_if](#).

Definition at line 31 of file [ParserDefaultImpl1.cpp](#).

Here is the call graph for this function:



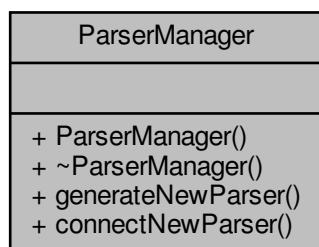
The documentation for this class was generated from the following files:

- [ParserDefaultImpl1.h](#)
- [ParserDefaultImpl1.cpp](#)

6.85 ParserManager Class Reference

```
#include <ParserManager.h>
```

Collaboration diagram for ParserManager:



Classes

- struct [GenerateNewParserResult](#)
- struct [NewParser](#)

Public Member Functions

- `ParserManager ()`
- `virtual ~ParserManager ()=default`
- `ParserManager::GenerateNewParserResult generateNewParser (ParserChangesInformation *changes)`
- `bool connectNewParser (ParserManager::NewParser newParser)`

6.85.1 Detailed Description

Definition at line 19 of file [ParserManager.h](#).

6.85.2 Constructor & Destructor Documentation

6.85.2.1 ParserManager()

```
ParserManager::ParserManager ( )
```

Definition at line 16 of file [ParserManager.cpp](#).

6.85.2.2 ~ParserManager()

```
virtual ParserManager::~ParserManager ( ) [virtual], [default]
```

6.85.3 Member Function Documentation

6.85.3.1 connectNewParser()

```
bool ParserManager::connectNewParser (
    ParserManager::NewParser newParser )
```

6.85.3.2 generateNewParser()

```
ParserManager::GenerateNewParserResult ParserManager::generateNewParser (
    ParserChangesInformation * changes )
```

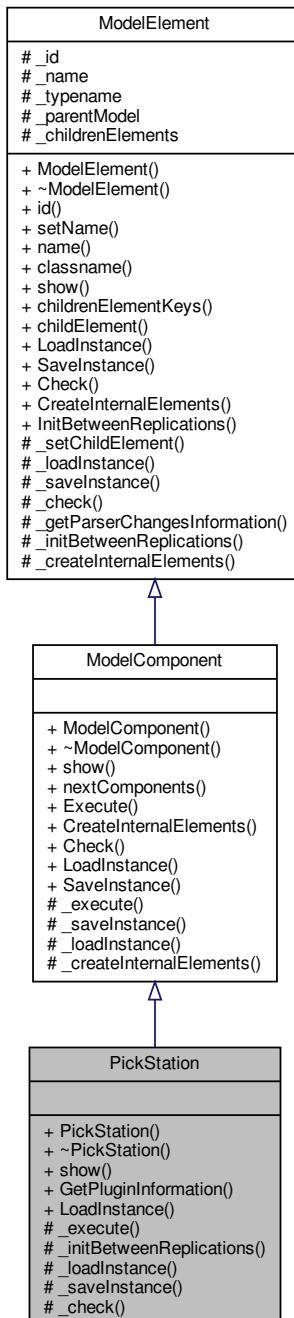
The documentation for this class was generated from the following files:

- [ParserManager.h](#)
- [ParserManager.cpp](#)

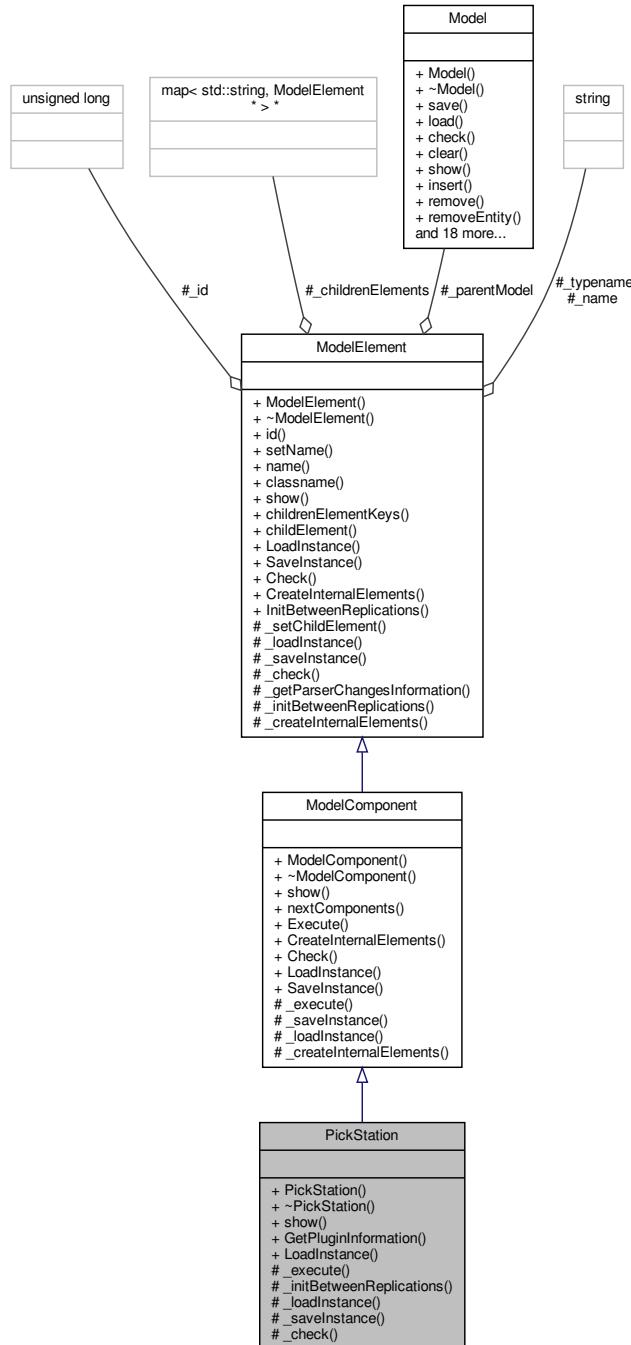
6.86 PickStation Class Reference

```
#include <PickStation.h>
```

Inheritance diagram for PickStation:



Collaboration diagram for PickStation:



Public Member Functions

- `PickStation (Model *model, std::string name=""")`
- virtual `~PickStation ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.86.1 Detailed Description

PickStation module DESCRIPTION The `PickStation` module allows an entity to select a particular station from the multiple stations specified. This module picks among the group of stations based on the selection logic defined with the module. The entity may then route, transport, convey, or connect to the station specified. If the method chosen is connect, the selected station is assigned to an entity attribute. The station selection process is based on the minimum or maximum value of a variety of system variables and expressions. TYPICAL USES A part sent to a processing station based on machine's availability at each station. A loan application sent to a set of loan officers based on the number sent to each officer. A customer selecting among cashier lines based on the least number waiting in each line. PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Test Condition Test condition to use for the station selection process, either Minimum or Maximum. Number En Route to Station The number of entities transferring to the station is considered in the station selection process. Number in Queue The number of entities in the queue at the station is considered in the station selection process. Number of Resources Busy The number of busy resources at the station is considered in the station selection process. Expression Determines if an additional user-defined expression is considered in the station selection process. Transfer Type Determines how an entity will be transferred out of this module to its next destination station—either `Route`, `Convey`, `Transport`, or `Connect`. Save Attribute Defines the name of the attribute that will store the station name that is selected, visible when the transfer method is Connect. Route Time Move time of the entity from its current station to the station determined through this module. Units Time units for route-time parameters.

Definition at line 62 of file `PickStation.h`.

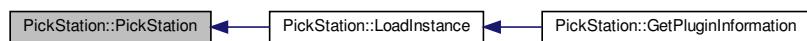
6.86.2 Constructor & Destructor Documentation

6.86.2.1 PickStation()

```
PickStation::PickStation (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file `PickStation.cpp`.

Here is the caller graph for this function:



6.86.2.2 ~PickStation()

```
virtual PickStation::~PickStation ( ) [virtual], [default]
```

6.86.3 Member Function Documentation

6.86.3.1 _check()

```
bool PickStation::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [PickStation.cpp](#).

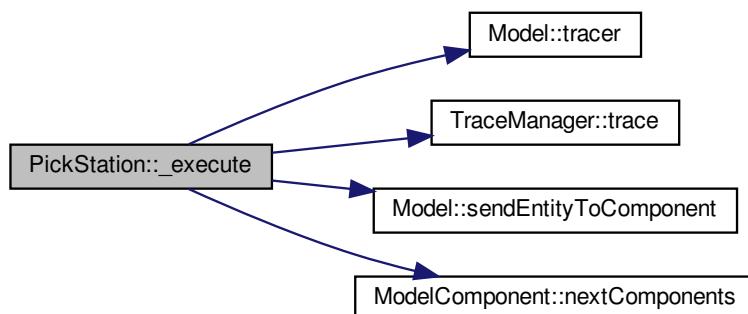
6.86.3.2 _execute()

```
void PickStation::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [PickStation.cpp](#).

Here is the call graph for this function:



6.86.3.3 `_initBetweenReplications()`

```
void PickStation::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [PickStation.cpp](#).

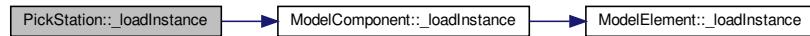
6.86.3.4 `_loadInstance()`

```
bool PickStation::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

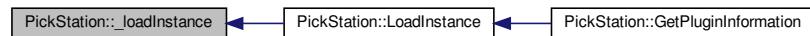
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [PickStation.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



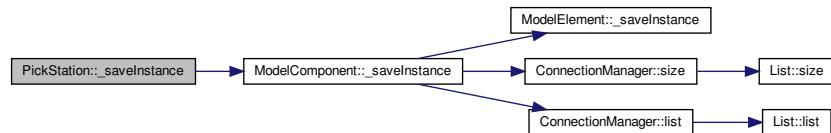
6.86.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * PickStation::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [PickStation.cpp](#).

Here is the call graph for this function:

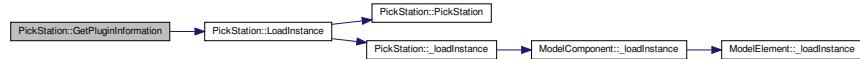


6.86.3.6 GetPluginInformation()

```
PluginInformation * PickStation::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [PickStation.cpp](#).

Here is the call graph for this function:



6.86.3.7 LoadInstance()

```
ModelComponent * PickStation::LoadInstance (   
    Model * model,  
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [PickStation.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



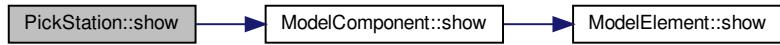
6.86.3.8 show()

```
std::string PickStation::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [PickStation.cpp](#).

Here is the call graph for this function:



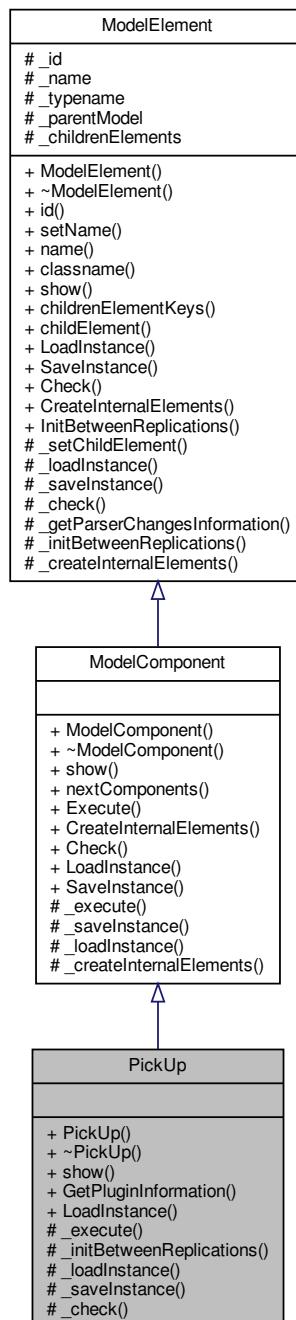
The documentation for this class was generated from the following files:

- [PickStation.h](#)
- [PickStation.cpp](#)

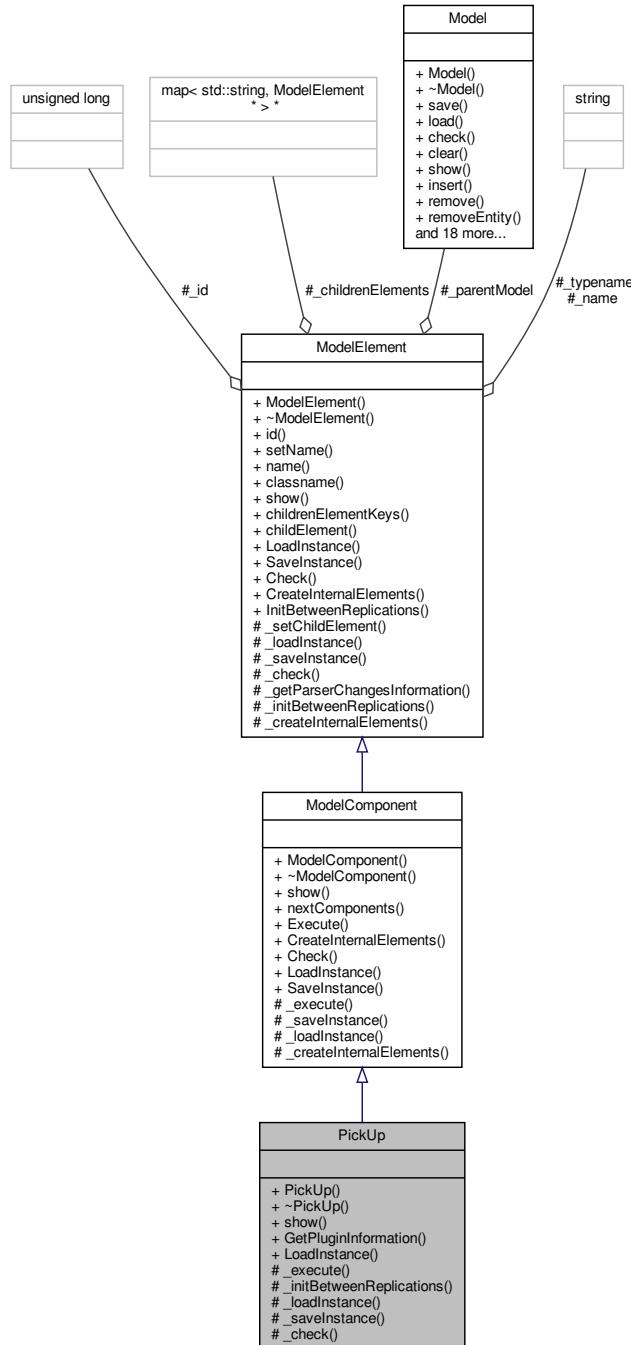
6.87 PickUp Class Reference

```
#include <PickUp.h>
```

Inheritance diagram for PickUp:



Collaboration diagram for PickUp:



Public Member Functions

- `PickUp (Model *model, std::string name="")`
- virtual `~PickUp ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.87.1 Detailed Description

Pickup module DESCRIPTION The Pickup module removes a number of consecutive entities from a given queue starting at a specified rank in the queue. The entities that are picked up are added to the end of the incoming entity's group. TYPICAL USES Gathering an order from various queue locations Gathering completed forms for an office order Picking up students at a bus stop for school PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Quantity Number of entities to pick up. Queue Name Name of the queue from which the entities will be picked up, starting at the specified rank. Starting Rank Starting rank of the entities to pick up from the queue, Queue Name.

Definition at line 38 of file `PickUp.h`.

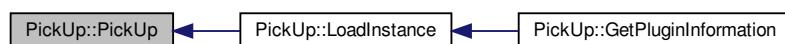
6.87.2 Constructor & Destructor Documentation

6.87.2.1 PickUp()

```
PickUp::PickUp (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file `PickUp.cpp`.

Here is the caller graph for this function:



6.87.2.2 ~PickUp()

```
virtual PickUp::~PickUp ( ) [virtual], [default]
```

6.87.3 Member Function Documentation

6.87.3.1 _check()

```
bool PickUp::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [PickUp.cpp](#).

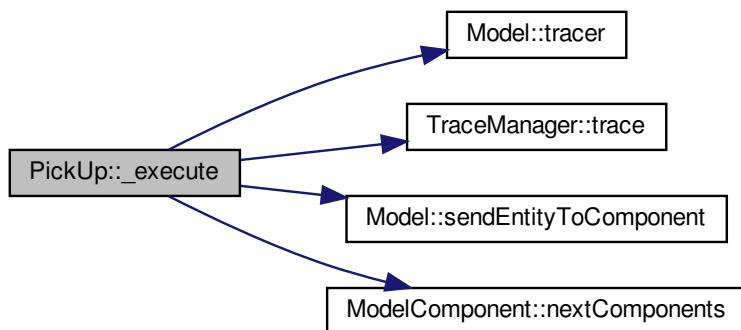
6.87.3.2 _execute()

```
void PickUp::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [PickUp.cpp](#).

Here is the call graph for this function:



6.87.3.3 `_initBetweenReplications()`

```
void PickUp::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [PickUp.cpp](#).

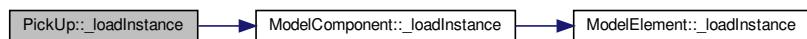
6.87.3.4 `_loadInstance()`

```
bool PickUp::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

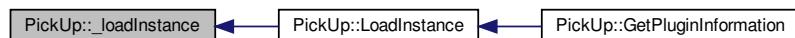
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [PickUp.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



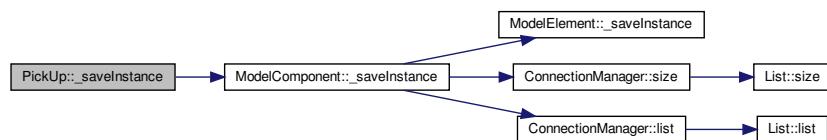
6.87.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * PickUp::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [PickUp.cpp](#).

Here is the call graph for this function:

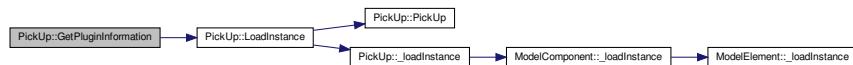


6.87.3.6 GetPluginInformation()

```
PluginInformation * PickUp::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [PickUp.cpp](#).

Here is the call graph for this function:



6.87.3.7 LoadInstance()

```
ModelComponent * PickUp::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [PickUp.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



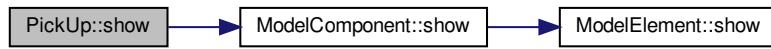
6.87.3.8 show()

```
std::string PickUp::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [PickUp.cpp](#).

Here is the call graph for this function:



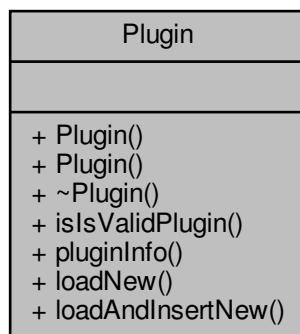
The documentation for this class was generated from the following files:

- [PickUp.h](#)
- [PickUp.cpp](#)

6.88 Plugin Class Reference

```
#include <Plugin.h>
```

Collaboration diagram for Plugin:



Public Member Functions

- [Plugin \(std::string filename_so_dll\)](#)
- [Plugin \(StaticGetPluginInformation getInformation\)](#)
- virtual [~Plugin \(\)=default](#)
- bool [isValidPlugin \(\) const](#)
- [PluginInformation * pluginInfo \(\) const](#)
- [ModelElement * loadNew \(Model *model, std::map< std::string, std::string > *fields\)](#)
- bool [loadAndInsertNew \(Model *model, std::map< std::string, std::string > *fields\)](#)

6.88.1 Detailed Description

A [Plugin](#) represents a dynamically linked component class ([ModelComponent](#)) or element class ([ModelElement](#)); It gives access to a [ModelComponent](#) so it can be used by the model. Classes like [Create](#), [Delay](#), and [Dispose](#) are examples of Plugins. It corresponds directly to the "Expansible" part (the capitalized 'E') of the GenESyS acronymous Plugins are NOT implemented yet

Definition at line [28](#) of file [Plugin.h](#).

6.88.2 Constructor & Destructor Documentation

6.88.2.1 Plugin() [1/2]

```
Plugin::Plugin (
    std::string filename_so_dll )
```

6.88.2.2 Plugin() [2/2]

```
Plugin::Plugin (
    StaticGetPluginInformation getInformation )
```

Definition at line [21](#) of file [Plugin.cpp](#).

6.88.2.3 ~Plugin()

```
virtual Plugin::~Plugin ( ) [virtual], [default]
```

Here is the caller graph for this function:



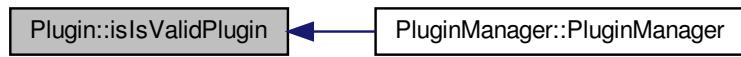
6.88.3 Member Function Documentation

6.88.3.1 isValidPlugin()

```
bool Plugin::isValidPlugin ( ) const
```

Definition at line 36 of file [Plugin.cpp](#).

Here is the caller graph for this function:

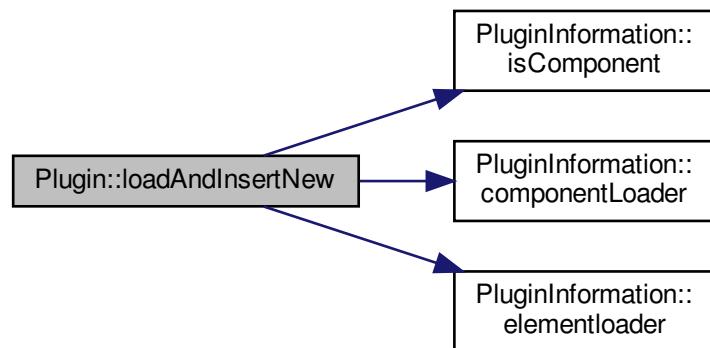


6.88.3.2 loadAndInsertNew()

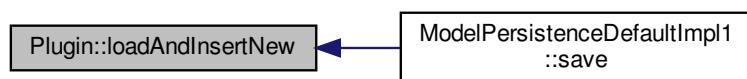
```
bool Plugin::loadAndInsertNew ( Model * model,  
                               std::map< std::string, std::string * > * fields )
```

Definition at line 54 of file [Plugin.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

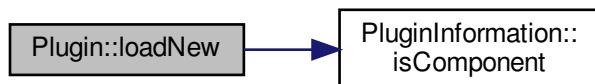


6.88.3.3 loadNew()

```
ModelElement * Plugin::loadNew (
    Model * model,
    std::map< std::string, std::string * > * fields )
```

Definition at line 46 of file [Plugin.cpp](#).

Here is the call graph for this function:

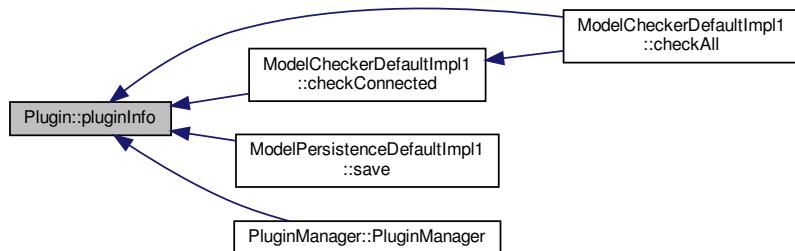


6.88.3.4 pluginInfo()

```
PluginInformation * Plugin::pluginInfo ( ) const
```

Definition at line 32 of file [Plugin.cpp](#).

Here is the caller graph for this function:



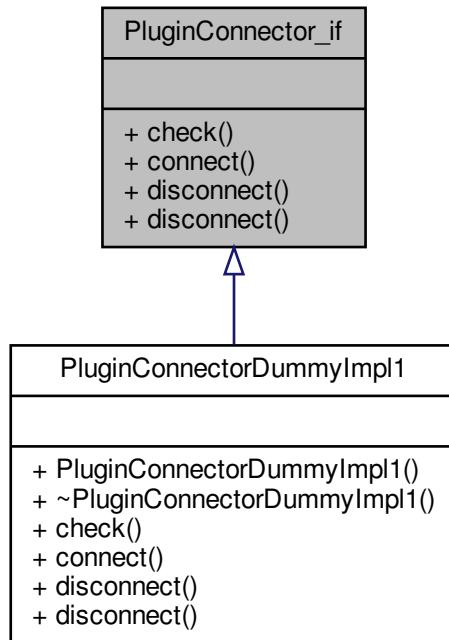
The documentation for this class was generated from the following files:

- [Plugin.h](#)
- [Plugin.cpp](#)

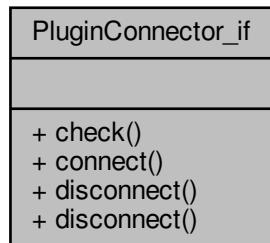
6.89 PluginConnector_if Class Reference

```
#include <PluginConnector_if.h>
```

Inheritance diagram for PluginConnector_if:



Collaboration diagram for PluginConnector_if:



Public Member Functions

- virtual **Plugin * check** (const std::string dynamicLibraryFilename)=0
- virtual **Plugin * connect** (const std::string dynamicLibraryFilename)=0
- virtual bool **disconnect** (const std::string dynamicLibraryFilename)=0
- virtual bool **disconnect** (**Plugin** *plugin)=0

6.89.1 Detailed Description

Definition at line 20 of file [PluginConnector_if.h](#).

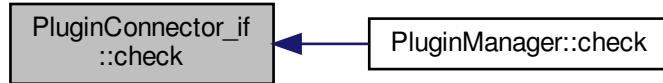
6.89.2 Member Function Documentation

6.89.2.1 check()

```
virtual Plugin* PluginConnector_if::check (
    const std::string dynamicLibraryFilename ) [pure virtual]
```

Implemented in [PluginConnectorDummyImpl1](#).

Here is the caller graph for this function:

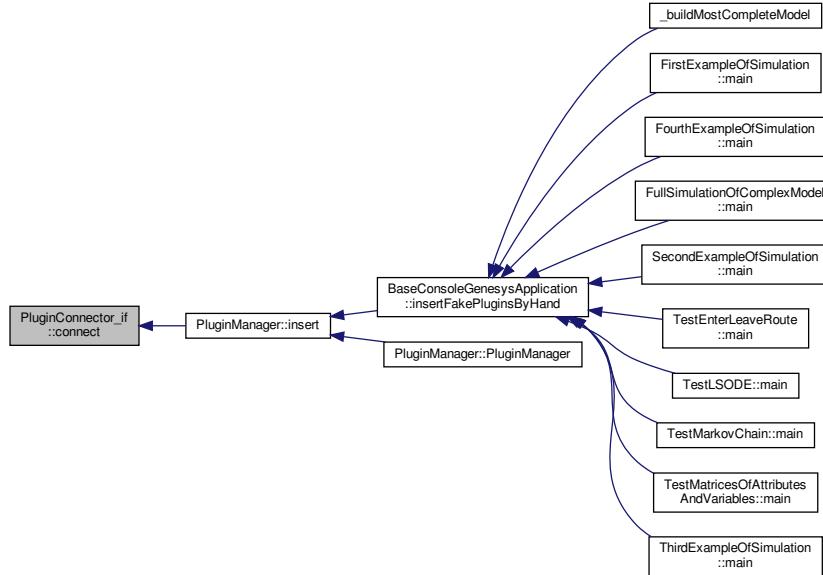


6.89.2.2 connect()

```
virtual Plugin* PluginConnector_if::connect (
    const std::string dynamicLibraryFilename ) [pure virtual]
```

Implemented in [PluginConnectorDummyImpl1](#).

Here is the caller graph for this function:

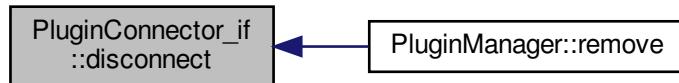


6.89.2.3 disconnect() [1/2]

```
virtual bool PluginConnector_if::disconnect (
    const std::string dynamicLibraryFilename ) [pure virtual]
```

Implemented in [PluginConnectorDummyImpl1](#).

Here is the caller graph for this function:



6.89.2.4 disconnect() [2/2]

```
virtual bool PluginConnector_if::disconnect (
    Plugin * plugin ) [pure virtual]
```

Implemented in [PluginConnectorDummyImpl1](#).

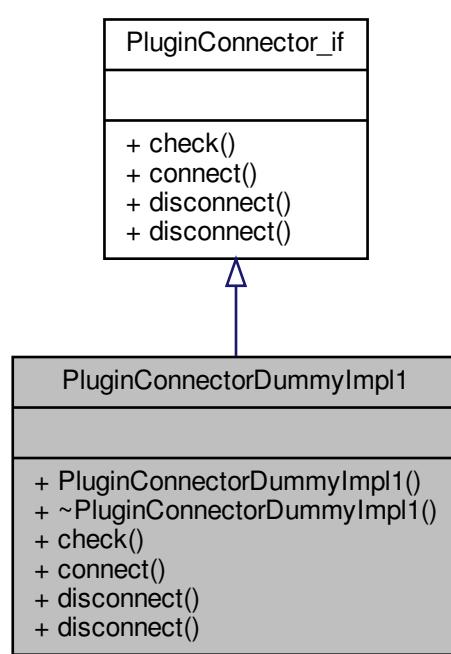
The documentation for this class was generated from the following file:

- [PluginConnector_if.h](#)

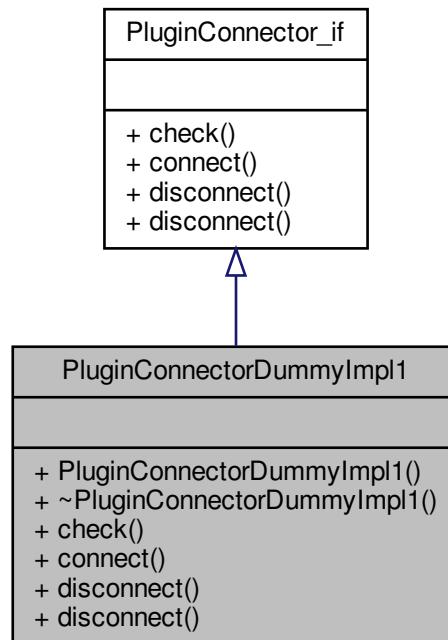
6.90 PluginConnectorDummyImpl1 Class Reference

```
#include <PluginConnectorDummyImpl1.h>
```

Inheritance diagram for PluginConnectorDummyImpl1:



Collaboration diagram for PluginConnectorDummyImpl1:



Public Member Functions

- `PluginConnectorDummyImpl1 ()`
- `virtual ~PluginConnectorDummyImpl1 ()=default`
- `virtual Plugin * check (const std::string dynamicLibraryFilename)`
- `virtual Plugin * connect (const std::string dynamicLibraryFilename)`
- `virtual bool disconnect (const std::string dynamicLibraryFilename)`
- `virtual bool disconnect (Plugin *plugin)`

6.90.1 Detailed Description

Definition at line 19 of file [PluginConnectorDummyImpl1.h](#).

6.90.2 Constructor & Destructor Documentation

6.90.2.1 PluginConnectorDummyImpl1()

```
PluginConnectorDummyImpl1::PluginConnectorDummyImpl1 ( )
```

Definition at line 46 of file [PluginConnectorDummyImpl1.cpp](#).

6.90.2.2 ~PluginConnectorDummyImpl1()

```
virtual PluginConnectorDummyImpl1::~PluginConnectorDummyImpl1 ( ) [virtual], [default]
```

6.90.3 Member Function Documentation

6.90.3.1 check()

```
Plugin * PluginConnectorDummyImpl1::check (
    const std::string dynamicLibraryFilename ) [virtual]
```

\xrefitem todo 1.

Implements [PluginConnector_if](#).

Definition at line 50 of file [PluginConnectorDummyImpl1.cpp](#).

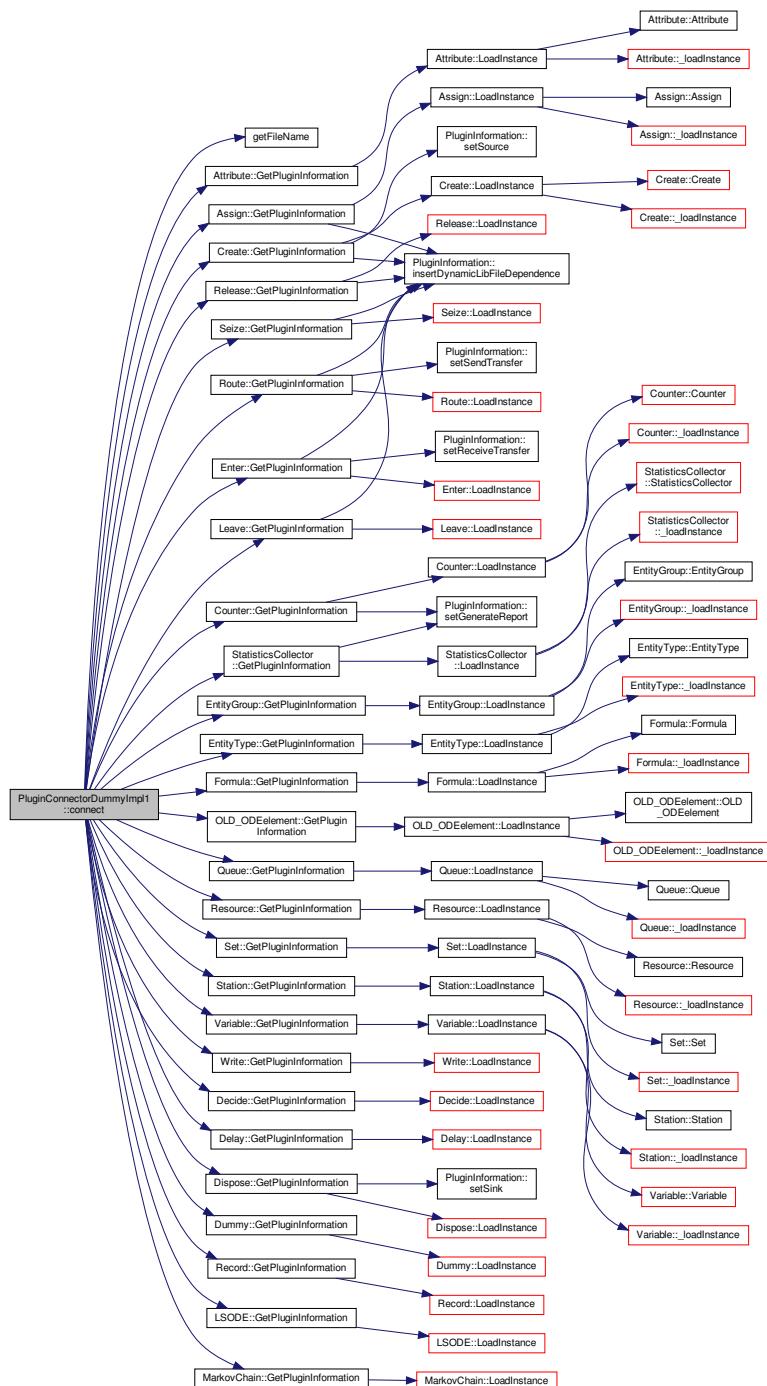
6.90.3.2 connect()

```
Plugin * PluginConnectorDummyImpl1::connect (
    const std::string dynamicLibraryFilename ) [virtual]
```

Implements [PluginConnector_if](#).

Definition at line 54 of file [PluginConnectorDummyImpl1.cpp](#).

Here is the call graph for this function:



6.90.3.3 disconnect() [1/2]

```
bool PluginConnectorDummyImpl1::disconnect (
    const std::string dynamicLibraryFilename ) [virtual]
```

Implements [PluginConnector_if](#).

Definition at line 122 of file [PluginConnectorDummyImpl1.cpp](#).

6.90.3.4 disconnect() [2/2]

```
bool PluginConnectorDummyImpl1::disconnect (
    Plugin * plugin )  [virtual]
```

Implements [PluginConnector_if](#).

Definition at line 126 of file [PluginConnectorDummyImpl1.cpp](#).

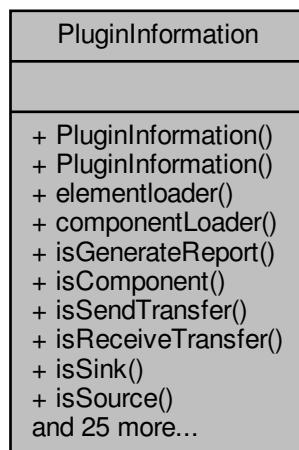
The documentation for this class was generated from the following files:

- [PluginConnectorDummyImpl1.h](#)
- [PluginConnectorDummyImpl1.cpp](#)

6.91 PluginInformation Class Reference

```
#include <PluginInformation.h>
```

Collaboration diagram for PluginInformation:



Public Member Functions

- `PluginInformation (std::string pluginTypename, StaticLoaderComponentInstance componentloader)`
- `PluginInformation (std::string pluginTypename, StaticLoaderElementInstance elementloader)`
- `StaticLoaderElementInstance elementloader () const`
- `StaticLoaderComponentInstance componentLoader () const`
- `bool isGenerateReport () const`
- `bool isComponent () const`
- `bool isSendTransfer () const`
- `bool isReceiveTransfer () const`
- `bool isSink () const`
- `bool isSource () const`
- `std::string observation () const`
- `std::string version () const`
- `std::string date () const`
- `std::string author () const`
- `std::string pluginTypename () const`
- `void insertDynamicLibFileDependence (std::string filename)`
- `void setDynamicLibFilenameDependencies (std::list< std::string > *dynamicLibFilenameDependencies)`
- `std::list< std::string > * dynamicLibFilenameDependencies () const`
- `void setGenerateReport (bool generateReport)`
- `void setSendTransfer (bool sendTransfer)`
- `void setReceiveTransfer (bool receiveTransfer)`
- `void setSink (bool Sink)`
- `void setSource (bool Source)`
- `void setObservation (std::string observation)`
- `void setVersion (std::string version)`
- `void setDate (std::string date)`
- `void setAuthor (std::string author)`
- `void setMaximumOutputs (unsigned short _maximumOutputs)`
- `unsigned short maximumOutputs () const`
- `void setMinimumOutputs (unsigned short _minimumOutputs)`
- `unsigned short minimumOutputs () const`
- `void setMaximumInputs (unsigned short _maximumInputs)`
- `unsigned short maximumInputs () const`
- `void setMinimumInputs (unsigned short _minimumInputs)`
- `unsigned short minimumInputs () const`

6.91.1 Detailed Description

Definition at line 32 of file [PluginInformation.h](#).

6.91.2 Constructor & Destructor Documentation

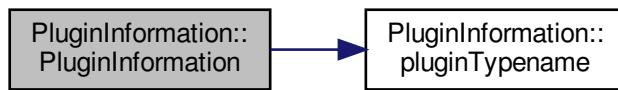
6.91.2.1 PluginInformation() [1/2]

```
PluginInformation::PluginInformation (
    std::string pluginTypename,
    StaticLoaderComponentInstance componentloader )
```

Todo :

Definition at line 17 of file [PluginInformation.cpp](#).

Here is the call graph for this function:

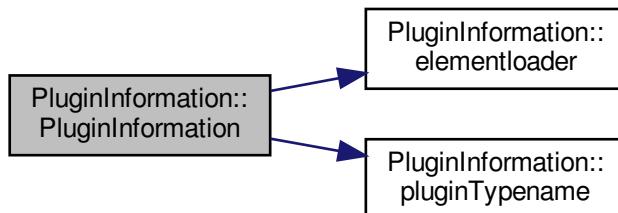


6.91.2.2 PluginInformation() [2/2]

```
PluginInformation::PluginInformation (
    std::string pluginTypename,
    StaticLoaderElementInstance elementloader )
```

Definition at line 24 of file [PluginInformation.cpp](#).

Here is the call graph for this function:



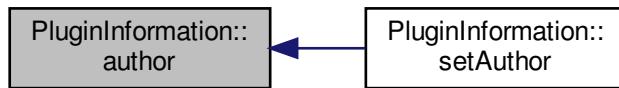
6.91.3 Member Function Documentation

6.91.3.1 author()

```
std::string PluginInformation::author ( ) const
```

Definition at line 75 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.2 componentLoader()

```
StaticLoaderComponentInstance PluginInformation::componentLoader ( ) const
```

Definition at line 35 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

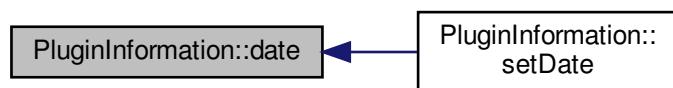


6.91.3.3 date()

```
std::string PluginInformation::date ( ) const
```

Definition at line 71 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

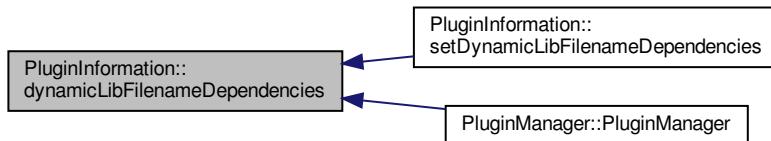


6.91.3.4 dynamicLibFilenameDependencies()

```
std::list< std::string > * PluginInformation::dynamicLibFilenameDependencies ( ) const
```

Definition at line 91 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

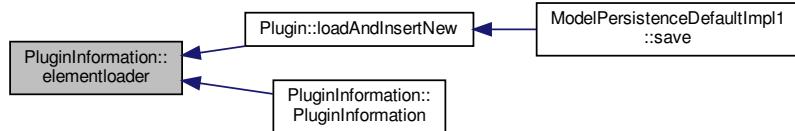


6.91.3.5 elementloader()

```
StaticLoaderElementInstance PluginInformation::elementloader ( ) const
```

Definition at line 31 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

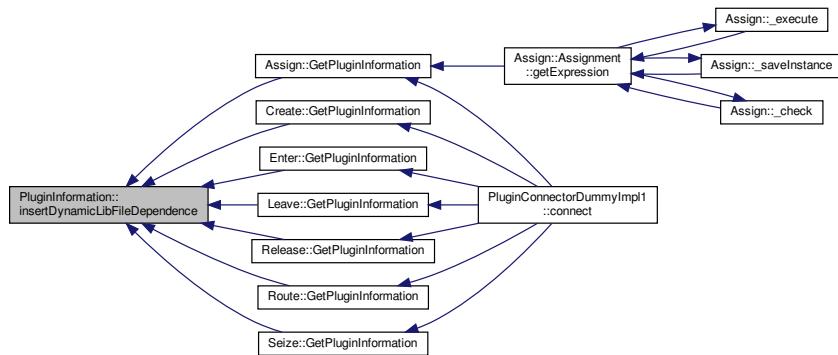


6.91.3.6 insertDynamicLibFileDependence()

```
void PluginInformation::insertDynamicLibFileDependence (
    std::string filename )
```

Definition at line 83 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

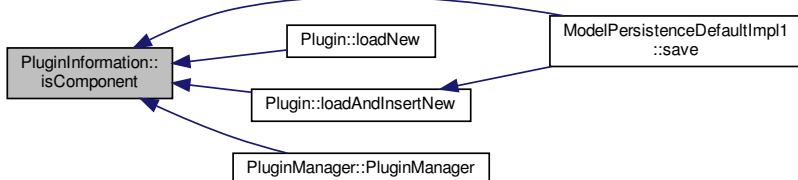


6.91.3.7 isComponent()

```
bool PluginInformation::isComponent ( ) const
```

Definition at line 43 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.8 isGenerateReport()

```
bool PluginInformation::isGenerateReport ( ) const
```

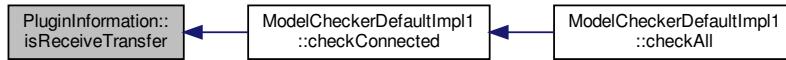
Definition at line 39 of file [PluginInformation.cpp](#).

6.91.3.9 `isReceiveTransfer()`

```
bool PluginInformation::isReceiveTransfer ( ) const
```

Definition at line 51 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.10 `isSendTransfer()`

```
bool PluginInformation::isSendTransfer ( ) const
```

Definition at line 47 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.11 `isSink()`

```
bool PluginInformation::isSink ( ) const
```

Definition at line 55 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.12 isSource()

```
bool PluginInformation::isSource ( ) const
```

Definition at line 59 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.13 maximumInputs()

```
unsigned short PluginInformation::maximumInputs ( ) const
```

Definition at line 151 of file [PluginInformation.cpp](#).

6.91.3.14 maximumOutputs()

```
unsigned short PluginInformation::maximumOutputs ( ) const
```

Definition at line 135 of file [PluginInformation.cpp](#).

6.91.3.15 minimumInputs()

```
unsigned short PluginInformation::minimumInputs ( ) const
```

Definition at line 159 of file [PluginInformation.cpp](#).

6.91.3.16 minimumOutputs()

```
unsigned short PluginInformation::minimumOutputs ( ) const
```

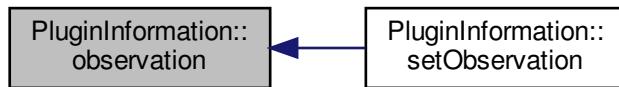
Definition at line 143 of file [PluginInformation.cpp](#).

6.91.3.17 `observation()`

```
std::string PluginInformation::observation () const
```

Definition at line 63 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

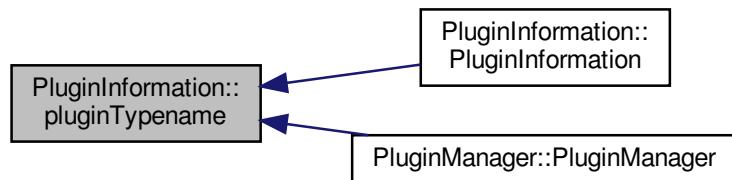


6.91.3.18 `pluginTypename()`

```
std::string PluginInformation::pluginTypename () const
```

Definition at line 79 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

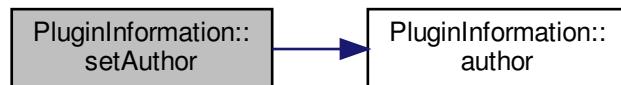


6.91.3.19 `setAuthor()`

```
void PluginInformation::setAuthor (
    std::string author )
```

Definition at line 127 of file [PluginInformation.cpp](#).

Here is the call graph for this function:

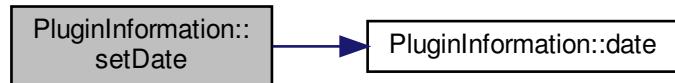


6.91.3.20 setDate()

```
void PluginInformation::setDate ( std::string date )
```

Definition at line 123 of file [PluginInformation.cpp](#).

Here is the call graph for this function:

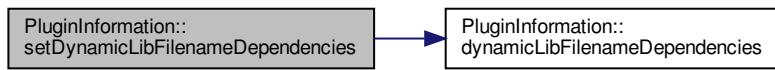


6.91.3.21 setDynamicLibFilenameDependencies()

```
void PluginInformation::setDynamicLibFilenameDependencies ( std::list< std::string > * dynamicLibFilenameDependencies )
```

Definition at line 87 of file [PluginInformation.cpp](#).

Here is the call graph for this function:

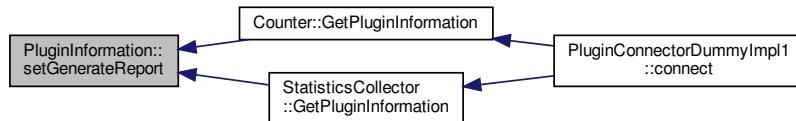


6.91.3.22 setGenerateReport()

```
void PluginInformation::setGenerateReport (
    bool generateReport )
```

Definition at line 95 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



6.91.3.23 setMaximumInputs()

```
void PluginInformation::setMaximumInputs (
    unsigned short _maximumInputs )
```

Definition at line 147 of file [PluginInformation.cpp](#).

6.91.3.24 setMaximumOutputs()

```
void PluginInformation::setMaximumOutputs (
    unsigned short _maximumOutputs )
```

Definition at line 131 of file [PluginInformation.cpp](#).

6.91.3.25 setMinimumInputs()

```
void PluginInformation::setMinimumInputs (
    unsigned short _minimumInputs )
```

Definition at line 155 of file [PluginInformation.cpp](#).

6.91.3.26 setMinimumOutputs()

```
void PluginInformation::setMinimumOutputs (
    unsigned short _minimumOutputs )
```

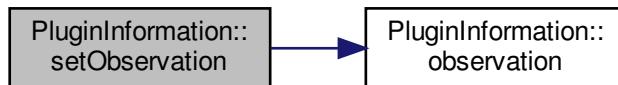
Definition at line 139 of file [PluginInformation.cpp](#).

6.91.3.27 setObservation()

```
void PluginInformation::setObservation (
    std::string observation )
```

Definition at line 115 of file [PluginInformation.cpp](#).

Here is the call graph for this function:

**6.91.3.28 setReceiveTransfer()**

```
void PluginInformation::setReceiveTransfer (
    bool receiveTransfer )
```

Definition at line 103 of file [PluginInformation.cpp](#).

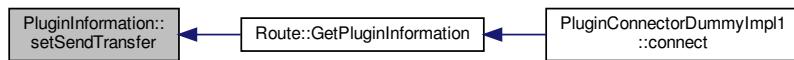
Here is the caller graph for this function:

**6.91.3.29 setSendTransfer()**

```
void PluginInformation::setSendTransfer (
    bool sendTransfer )
```

Definition at line 99 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

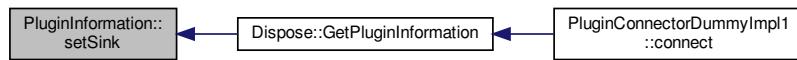


6.91.3.30 setSink()

```
void PluginInformation::setSink (
    bool Sink )
```

Definition at line 107 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

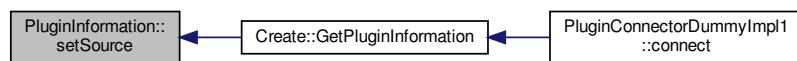


6.91.3.31 setSource()

```
void PluginInformation::setSource (
    bool Source )
```

Definition at line 111 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:

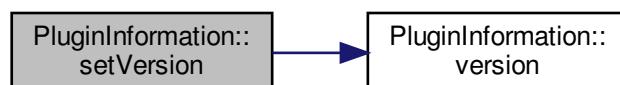


6.91.3.32 setVersion()

```
void PluginInformation::setVersion (
    std::string version )
```

Definition at line 119 of file [PluginInformation.cpp](#).

Here is the call graph for this function:

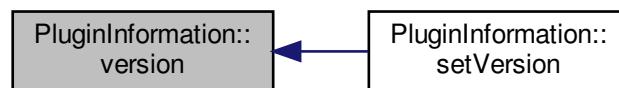


6.91.3.33 version()

```
std::string PluginInformation::version ( ) const
```

Definition at line 67 of file [PluginInformation.cpp](#).

Here is the caller graph for this function:



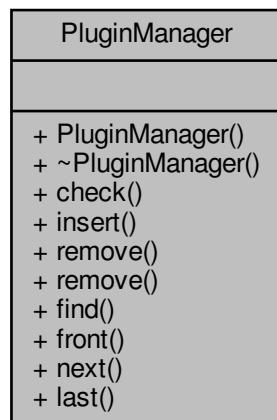
The documentation for this class was generated from the following files:

- [PluginInformation.h](#)
- [PluginInformation.cpp](#)

6.92 PluginManager Class Reference

```
#include <PluginManager.h>
```

Collaboration diagram for PluginManager:



Public Member Functions

- `PluginManager (Simulator *simulator)`
- `virtual ~PluginManager ()=default`
- `bool check (const std::string dynamicLibraryFilename)`
- `Plugin * insert (const std::string dynamicLibraryFilename)`
- `bool remove (const std::string dynamicLibraryFilename)`
- `bool remove (Plugin *plugin)`
- `Plugin * find (std::string pluginTypeName)`
- `Plugin * front ()`
- `Plugin * next ()`
- `Plugin * last ()`

6.92.1 Detailed Description

Definition at line [23](#) of file [PluginManager.h](#).

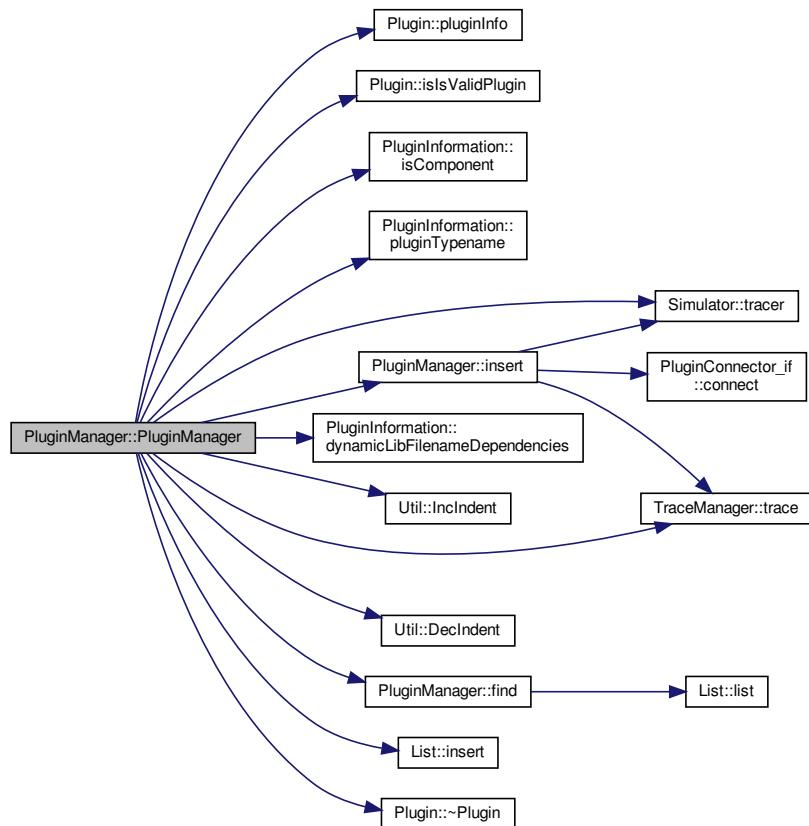
6.92.2 Constructor & Destructor Documentation

6.92.2.1 PluginManager()

```
PluginManager::PluginManager (
    Simulator * simulator )
```

Definition at line [18](#) of file [PluginManager.cpp](#).

Here is the call graph for this function:



6.92.2.2 ~PluginManager()

```
virtual PluginManager::~PluginManager() [virtual], [default]
```

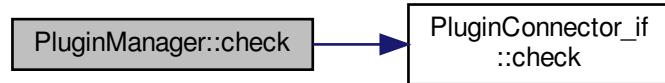
6.92.3 Member Function Documentation

6.92.3.1 check()

```
bool PluginManager::check (
    const std::string dynamicLibraryFilename )
```

Definition at line 68 of file [PluginManager.cpp](#).

Here is the call graph for this function:



6.92.3.2 find()

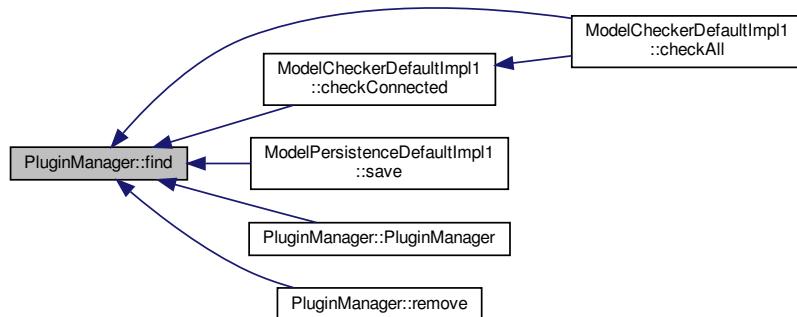
```
Plugin * PluginManager::find (
    std::string pluginTypeName )
```

Definition at line 115 of file [PluginManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.92.3.3 front()

```
Plugin * PluginManager::front ( )
```

Definition at line 125 of file [PluginManager.cpp](#).

Here is the call graph for this function:

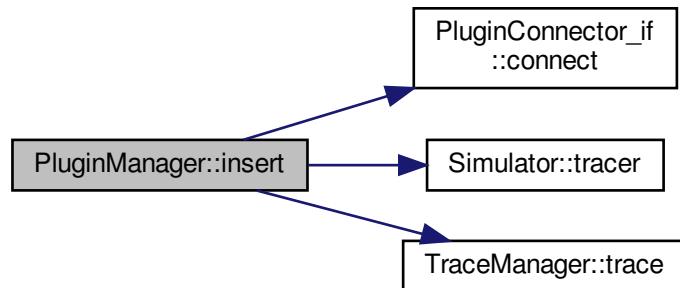


6.92.3.4 insert()

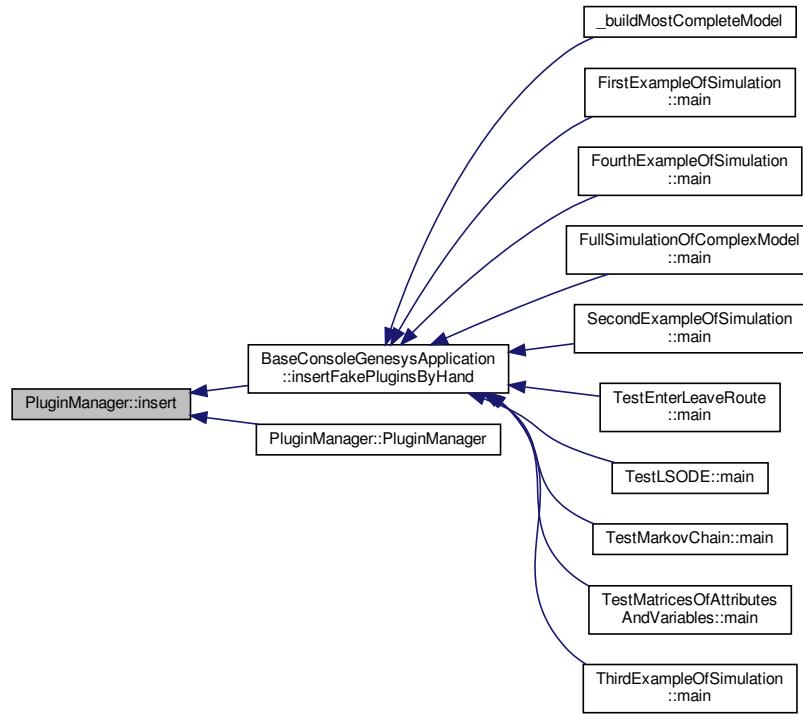
```
Plugin * PluginManager::insert (
    const std::string dynamicLibraryFilename )
```

Definition at line 78 of file [PluginManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.92.3.5 last()

```
Plugin * PluginManager::last ( )
```

Definition at line 135 of file [PluginManager.cpp](#).

Here is the call graph for this function:

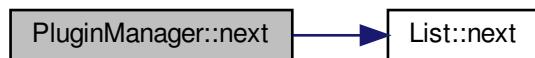


6.92.3.6 next()

```
Plugin * PluginManager::next ( )
```

Definition at line 130 of file [PluginManager.cpp](#).

Here is the call graph for this function:

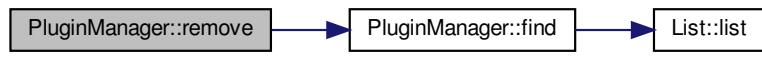


6.92.3.7 remove() [1/2]

```
bool PluginManager::remove (
    const std::string dynamicLibraryFilename )
```

Definition at line 94 of file [PluginManager.cpp](#).

Here is the call graph for this function:

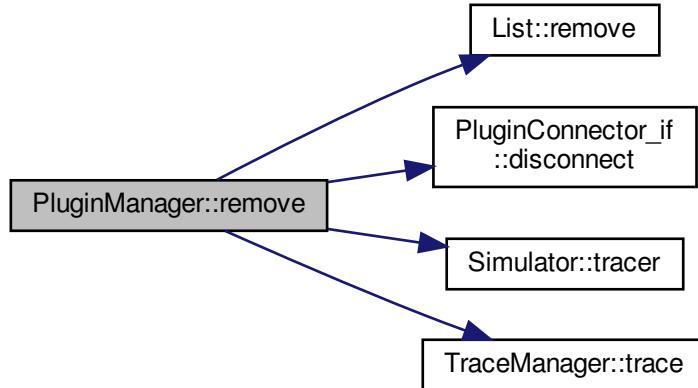


6.92.3.8 remove() [2/2]

```
bool PluginManager::remove (
    Plugin * plugin )
```

Definition at line 100 of file [PluginManager.cpp](#).

Here is the call graph for this function:



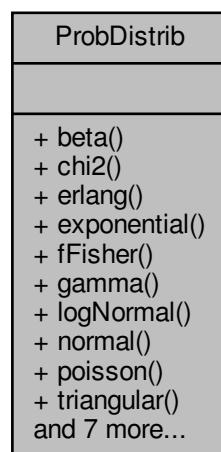
The documentation for this class was generated from the following files:

- [PluginManager.h](#)
- [PluginManager.cpp](#)

6.93 ProbDistrib Class Reference

```
#include <ProbDistrib.h>
```

Collaboration diagram for ProbDistrib:



Static Public Member Functions

- static double [beta](#) (double x, double alpha, double beta)
- static double [chi2](#) (double x, double m)
- static double [erlang](#) (double x, double shape, double scale)
- static double [exponential](#) (double x, double mean)
- static double [fFisher](#) (double x, double k, double m)
- static double [gamma](#) (double x, double shape, double scale)
- static double [logNormal](#) (double x, double mean, double stddev)
- static double [normal](#) (double x, double mean, double stddev)
- static double [poisson](#) (double x, double mean)
- static double [triangular](#) (double x, double min, double mode, double max)
- static double [tStudent](#) (double x, double mean, double stddev, unsigned int degreeFreedom)
- static double [uniform](#) (double x, double min, double max)
- static double [weibull](#) (double x, double shape, double scale)
- static double [inverseChi2](#) (double cumulativeProbability, double m)
- static double [inverseFFisher](#) (double cumulativeProbability, double k, double m)
- static double [inverseNormal](#) (double cumulativeProbability, double mean, double stddev)
- static double [inverseTStudent](#) (double cumulativeProbability, double mean, double stddev, double degreeFreedom)

6.93.1 Detailed Description

Definition at line [20](#) of file [ProbDistrib.h](#).

6.93.2 Member Function Documentation

6.93.2.1 [beta\(\)](#)

```
double ProbDistrib::beta (
    double x,
    double alpha,
    double beta )  [static]
```

Definition at line [52](#) of file [ProbDistrib.cpp](#).

6.93.2.2 [chi2\(\)](#)

```
double ProbDistrib::chi2 (
    double x,
    double m )  [static]
```

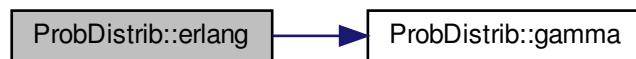
Definition at line [82](#) of file [ProbDistrib.cpp](#).

6.93.2.3 erlang()

```
double ProbDistrib::erlang (
    double x,
    double shape,
    double scale ) [static]
```

Definition at line 36 of file [ProbDistrib.cpp](#).

Here is the call graph for this function:



6.93.2.4 exponential()

```
double ProbDistrib::exponential (
    double x,
    double mean ) [static]
```

Definition at line 31 of file [ProbDistrib.cpp](#).

6.93.2.5 fFisher()

```
double ProbDistrib::fFisher (
    double x,
    double k,
    double m ) [static]
```

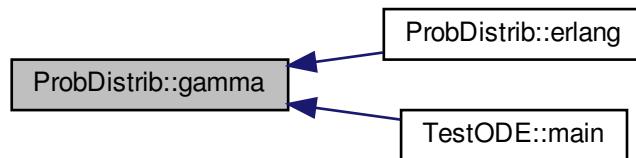
Definition at line 77 of file [ProbDistrib.cpp](#).

6.93.2.6 gamma()

```
double ProbDistrib::gamma (
    double x,
    double shape,
    double scale ) [static]
```

Definition at line 47 of file [ProbDistrib.cpp](#).

Here is the caller graph for this function:



6.93.2.7 inverseChi2()

```
double ProbDistrib::inverseChi2 (
    double cumulativeProbability,
    double m ) [static]
```

Definition at line 107 of file [ProbDistrib.cpp](#).

6.93.2.8 inverseFFisher()

```
double ProbDistrib::inverseFFisher (
    double cumulativeProbability,
    double k,
    double m ) [static]
```

Definition at line 102 of file [ProbDistrib.cpp](#).

6.93.2.9 inverseNormal()

```
double ProbDistrib::inverseNormal (
    double cumulativeProbability,
    double mean,
    double stddev ) [static]
```

Definition at line 92 of file [ProbDistrib.cpp](#).

6.93.2.10 inverseTStudent()

```
double ProbDistrib::inverseTStudent (
    double cumulativeProbability,
    double mean,
    double stddev,
    double degreeFreedom ) [static]
```

Definition at line 97 of file [ProbDistrib.cpp](#).

6.93.2.11 logNormal()

```
double ProbDistrib::logNormal (
    double x,
    double mean,
    double stddev ) [static]
```

Definition at line 62 of file [ProbDistrib.cpp](#).

6.93.2.12 normal()

```
double ProbDistrib::normal (
    double x,
    double mean,
    double stddev ) [static]
```

Definition at line 40 of file [ProbDistrib.cpp](#).

Here is the caller graph for this function:



6.93.2.13 poisson()

```
double ProbDistrib::poisson (
    double x,
    double mean ) [static]
```

Definition at line 87 of file [ProbDistrib.cpp](#).

6.93.2.14 triangular()

```
double ProbDistrib::triangular (
    double x,
    double min,
    double mode,
    double max ) [static]
```

Definition at line 67 of file [ProbDistrib.cpp](#).

6.93.2.15 tStudent()

```
double ProbDistrib::tStudent (
    double x,
    double mean,
    double stddev,
    unsigned int degreeFreedom ) [static]
```

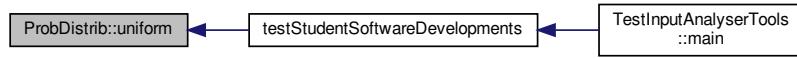
Definition at line 72 of file [ProbDistrib.cpp](#).

6.93.2.16 uniform()

```
double ProbDistrib::uniform (
    double x,
    double min,
    double max ) [static]
```

Definition at line 24 of file [ProbDistrib.cpp](#).

Here is the caller graph for this function:



6.93.2.17 weibull()

```
double ProbDistrib::weibull (
    double x,
    double shape,
    double scale ) [static]
```

Definition at line 57 of file [ProbDistrib.cpp](#).

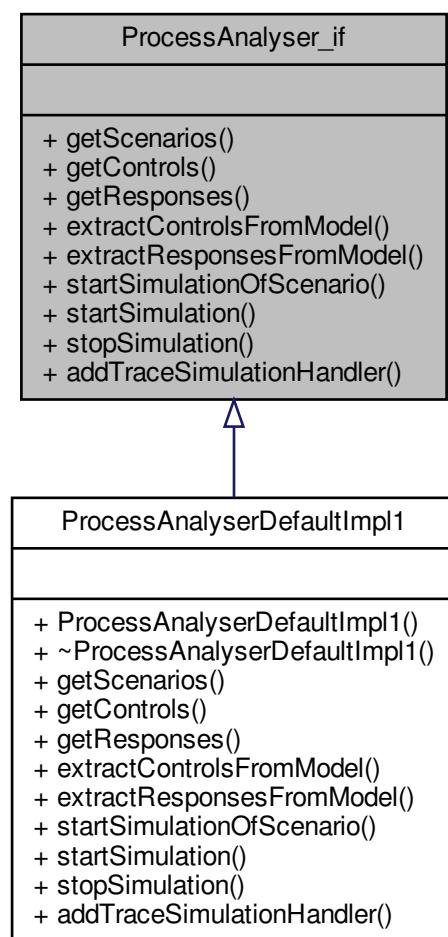
The documentation for this class was generated from the following files:

- [ProbDistrib.h](#)
- [ProbDistrib.cpp](#)

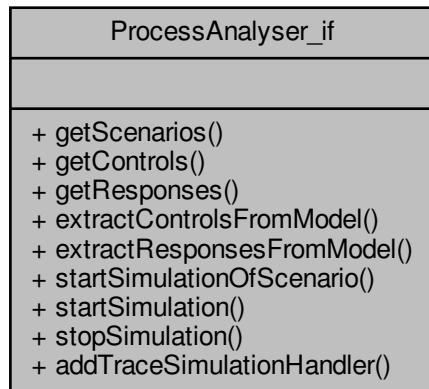
6.94 ProcessAnalyser_if Class Reference

```
#include <ProcessAnalyser_if.h>
```

Inheritance diagram for ProcessAnalyser_if:



Collaboration diagram for ProcessAnalyser_if:



Public Member Functions

- virtual `List< SimulationScenario * > * getScenarios () const =0`
- virtual `List< SimulationControl * > * getControls () const =0`
- virtual `List< SimulationResponse * > * getResponses () const =0`
- virtual `List< SimulationControl * > * extractControlsFromModel (std::string modelFilename) const =0`
- virtual `List< SimulationResponse * > * extractResponsesFromModel (std::string modelFilename) const =0`
- virtual void `startSimulationOfScenario (SimulationScenario *scenario)=0`
- virtual void `startSimulation ()=0`
- virtual void `stopSimulation ()=0`
- virtual void `addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)=0`

6.94.1 Detailed Description

The process analyser allows to extract controls and responses from a model, include some of them as controls and responses for a set of scenarios to be simulated

Definition at line 26 of file [ProcessAnalyser_if.h](#).

6.94.2 Member Function Documentation

6.94.2.1 addTraceSimulationHandler()

```
virtual void ProcessAnalyser_if::addTraceSimulationHandler (
    traceSimulationProcessListener traceSimulationProcessListener ) [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.2 extractControlsFromModel()

```
virtual List<SimulationControl*>* ProcessAnalyser_if::extractControlsFromModel ( std::string modelfilename ) const [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.3 extractResponsesFromModel()

```
virtual List<SimulationResponse*>* ProcessAnalyser_if::extractResponsesFromModel ( std::string modelfilename ) const [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.4 getControls()

```
virtual List<SimulationControl*>* ProcessAnalyser_if::getControls ( ) const [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.5 getResponses()

```
virtual List<SimulationResponse*>* ProcessAnalyser_if::getResponses ( ) const [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.6 getScenarios()

```
virtual List<SimulationScenario*>* ProcessAnalyser_if::getScenarios ( ) const [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.7 startSimulation()

```
virtual void ProcessAnalyser_if::startSimulation ( ) [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.8 startSimulationOfScenario()

```
virtual void ProcessAnalyser_if::startSimulationOfScenario (
    SimulationScenario * scenario ) [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

6.94.2.9 stopSimulation()

```
virtual void ProcessAnalyser_if::stopSimulation ( ) [pure virtual]
```

Implemented in [ProcessAnalyserDefaultImpl1](#).

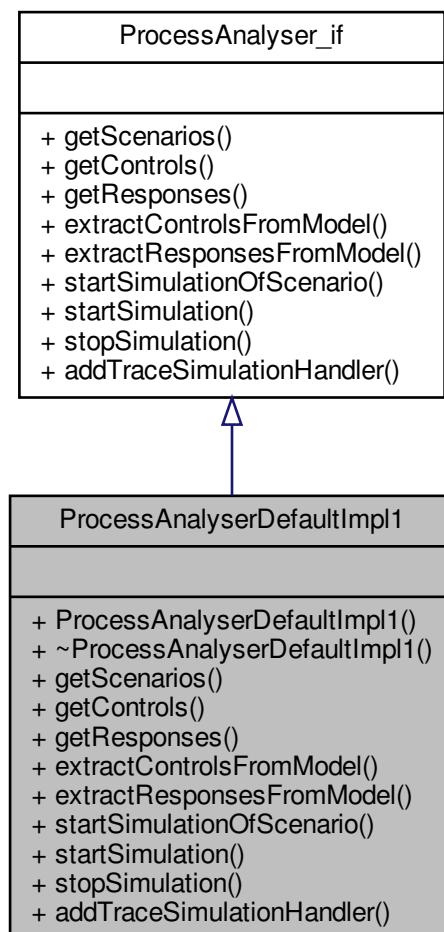
The documentation for this class was generated from the following file:

- [ProcessAnalyser_if.h](#)

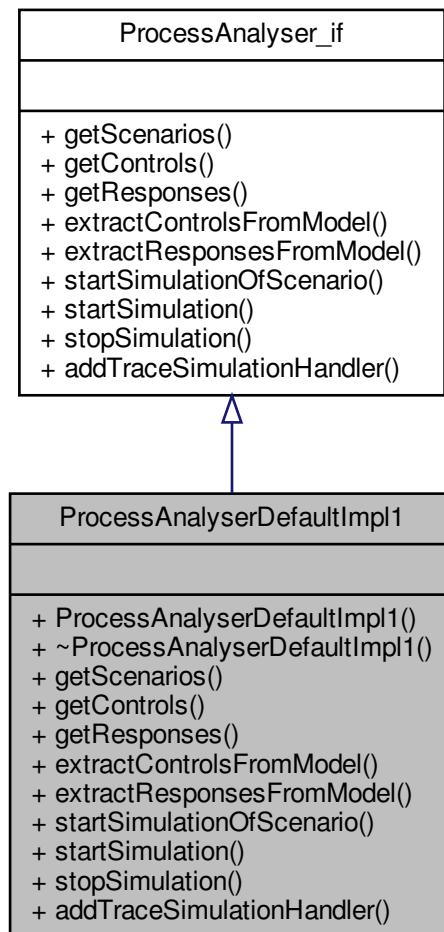
6.95 ProcessAnalyserDefaultImpl1 Class Reference

```
#include <ProcessAnalyserDefaultImpl1.h>
```

Inheritance diagram for ProcessAnalyserDefaultImpl1:



Collaboration diagram for ProcessAnalyserDefaultImpl1:



Public Member Functions

- `ProcessAnalyserDefaultImpl1 ()`
- `virtual ~ProcessAnalyserDefaultImpl1 ()=default`
- `virtual List< SimulationScenario * > * getScenarios () const`
- `virtual List< SimulationControl * > * getControls () const`
- `virtual List< SimulationResponse * > * getResponses () const`
- `virtual List< SimulationControl * > * extractControlsFromModel (std::string modelFilename) const`
- `virtual List< SimulationResponse * > * extractResponsesFromModel (std::string modelFilename) const`
- `virtual void startSimulationOfScenario (SimulationScenario *scenario)`
- `virtual void startSimulation ()`
- `virtual void stopSimulation ()`
- `virtual void addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)`

6.95.1 Detailed Description

Definition at line 22 of file [ProcessAnalyserDefaultImpl1.h](#).

6.95.2 Constructor & Destructor Documentation

6.95.2.1 ProcessAnalyserDefaultImpl1()

```
ProcessAnalyserDefaultImpl1::ProcessAnalyserDefaultImpl1 ( )
```

Definition at line 16 of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.2.2 ~ProcessAnalyserDefaultImpl1()

```
virtual ProcessAnalyserDefaultImpl1::~ProcessAnalyserDefaultImpl1 ( ) [virtual], [default]
```

6.95.3 Member Function Documentation

6.95.3.1 addTraceSimulationHandler()

```
void ProcessAnalyserDefaultImpl1::addTraceSimulationHandler (
    traceSimulationProcessListener traceSimulationProcessListener ) [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line 45 of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.2 extractControlsFromModel()

```
List< SimulationControl * > * ProcessAnalyserDefaultImpl1::extractControlsFromModel (
    std::string modelFilename ) const [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line 30 of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.3 extractResponsesFromModel()

```
List< SimulationResponse * > * ProcessAnalyserDefaultImpl1::extractResponsesFromModel (
    std::string modelFilename ) const [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line 33 of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.4 getControls()

```
List< SimulationControl * > * ProcessAnalyserDefaultImpl1::getControls ( ) const [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line [23](#) of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.5 getResponses()

```
List< SimulationResponse * > * ProcessAnalyserDefaultImpl1::getResponses ( ) const [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line [27](#) of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.6 getScenarios()

```
List< SimulationScenario * > * ProcessAnalyserDefaultImpl1::getScenarios ( ) const [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line [20](#) of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.7 startSimulation()

```
void ProcessAnalyserDefaultImpl1::startSimulation ( ) [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line [39](#) of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.8 startSimulationOfScenario()

```
void ProcessAnalyserDefaultImpl1::startSimulationOfScenario (
    SimulationScenario * scenario ) [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line [36](#) of file [ProcessAnalyserDefaultImpl1.cpp](#).

6.95.3.9 stopSimulation()

```
void ProcessAnalyserDefaultImpl1::stopSimulation ( ) [virtual]
```

Implements [ProcessAnalyser_if](#).

Definition at line [42](#) of file [ProcessAnalyserDefaultImpl1.cpp](#).

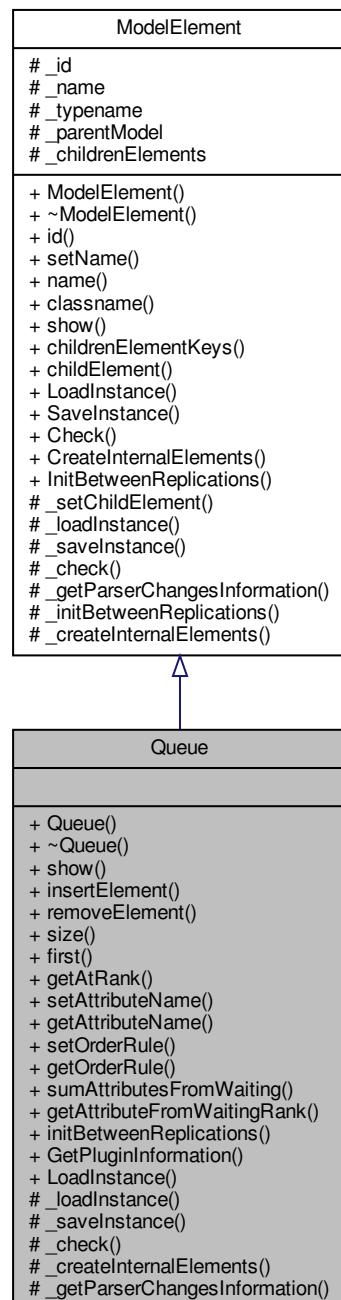
The documentation for this class was generated from the following files:

- [ProcessAnalyserDefaultImpl1.h](#)
- [ProcessAnalyserDefaultImpl1.cpp](#)

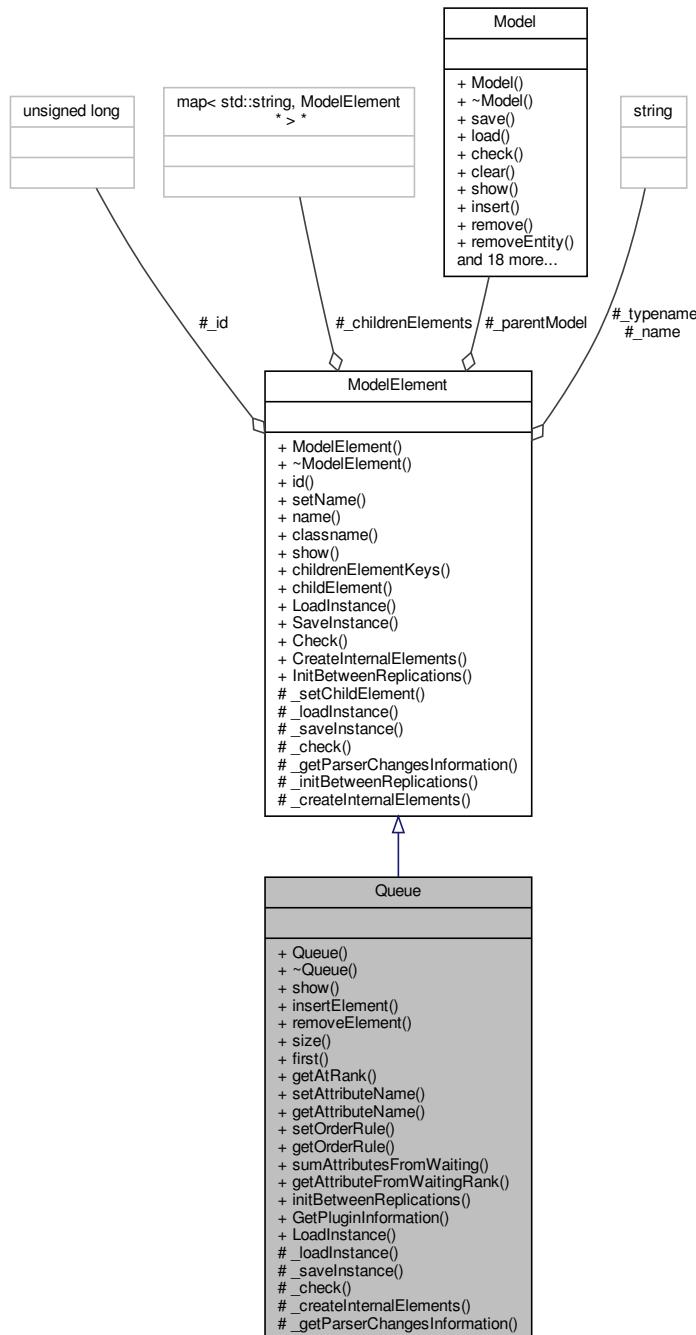
6.96 Queue Class Reference

```
#include <Queue.h>
```

Inheritance diagram for Queue:



Collaboration diagram for Queue:



Public Types

- enum OrderRule : int { OrderRule::FIFO = 1, OrderRule::LIFO = 2, OrderRule::HIGHESTVALUE = 3, OrderRule::SMALLESTVALUE = 4 }

Public Member Functions

- Queue (Model *model, std::string name="")

- virtual `~Queue ()`
- virtual std::string `show ()`
- void `insertElement (Waiting *element)`
- void `removeElement (Waiting *element)`
- unsigned int `size ()`
- `Waiting * first ()`
- `Waiting * getAtRank (unsigned int rank)`
- void `setAttributeName (std::string _attributeName)`
- std::string `getAttributeName () const`
- void `setOrderRule (OrderRule _orderRule)`
- `Queue::OrderRule getOrderRule () const`
- double `sumAttributesFromWaiting (Util::identification attributeID)`
- double `getattributeFromWaitingRank (unsigned int rank, Util::identification attributeID)`
- void `initBetweenReplications ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

- virtual `ParserChangesInformation * _getParserChangesInformation ()`

Additional Inherited Members

6.96.1 Detailed Description

Queue module DESCRIPTION This data module may be utilized to change the ranking rule for a specified queue. The default ranking rule for all queues is First In, First Out unless otherwise specified in this module. There is an additional field that allows the queue to be defined as shared. TYPICAL USES Stack of work waiting for a resource at a Process module Holding area for documents waiting to be collated at a **Batch** module Prompt Description Name The name of the queue whose characteristics are being defined. This name must be unique. Type Ranking rule for the queue, which can be based on an attribute. Types include First In, First Out; Last In, First Out; Lowest **Attribute** Value (first); and Highest **Attribute** Value (first). A low attribute value would be 0 or 1, while a high value may be 200 or 300. **Attribute** Name **Attribute** that will be evaluated for the Lowest **Attribute** Value or Highest **Attribute** Value types. Entities with lowest or highest values of the attribute will be ranked first in the queue, with ties being broken using the First In, First Out rule. Shared Check box that determines whether a specific queue is used in multiple places within the simulation model. Shared queues can only be used for seizing resources (for example, with the **Seize** module from the Advanced Process panel). Report Statistics Specifies whether or not statistics will be collected automatically and stored in the report database for this queue.

Definition at line 91 of file **Queue.h**.

6.96.2 Member Enumeration Documentation

6.96.2.1 OrderRule

```
enum Queue::OrderRule : int [strong]
```

Enumerator

FIFO	
LIFO	
HIGHESTVALUE	
SMALLESTVALUE	

Definition at line 94 of file [Queue.h](#).

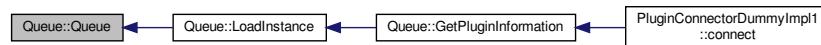
6.96.3 Constructor & Destructor Documentation

6.96.3.1 Queue()

```
Queue::Queue (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Queue.cpp](#).

Here is the caller graph for this function:



6.96.3.2 ~Queue()

```
Queue::~Queue ( ) [virtual]
```

Definition at line 29 of file [Queue.cpp](#).

6.96.4 Member Function Documentation

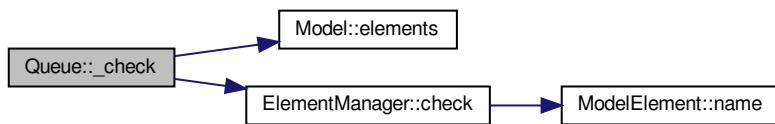
6.96.4.1 `_check()`

```
bool Queue::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 134 of file [Queue.cpp](#).

Here is the call graph for this function:



6.96.4.2 `_createInternalElements()`

```
void Queue::_createInternalElements ( ) [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelElement](#).

Definition at line 138 of file [Queue.cpp](#).

6.96.4.3 `_getParserChangesInformation()`

```
ParserChangesInformation * Queue::_getParserChangesInformation ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 142 of file [Queue.cpp](#).

6.96.4.4 `_loadInstance()`

```
bool Queue::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 115 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.96.4.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Queue::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 127 of file [Queue.cpp](#).

Here is the call graph for this function:



6.96.4.6 first()

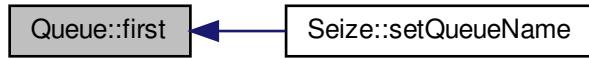
```
Waiting * Queue::first ( )
```

Definition at line 60 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.96.4.7 getAtRank()

```
Waiting * Queue::getAtRank ( unsigned int rank )
```

Definition at line 64 of file [Queue.cpp](#).

Here is the call graph for this function:

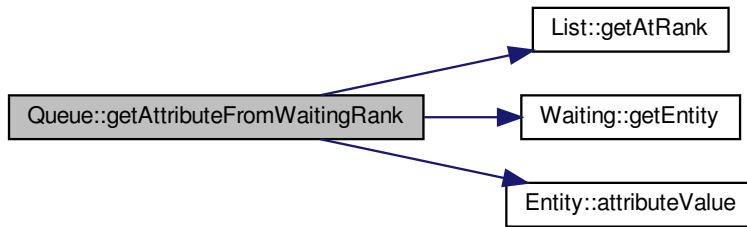


6.96.4.8 `getAttributeFromWaitingRank()`

```
double Queue::getAttributeFromWaitingRank (
    unsigned int rank,
    Util::identification attributeID )
```

Definition at line 92 of file [Queue.cpp](#).

Here is the call graph for this function:



6.96.4.9 `getAttributeName()`

```
std::string Queue::getAttributeName ( ) const
```

Definition at line 72 of file [Queue.cpp](#).

6.96.4.10 `getOrderRule()`

```
Queue::OrderRule Queue::getOrderRule ( ) const
```

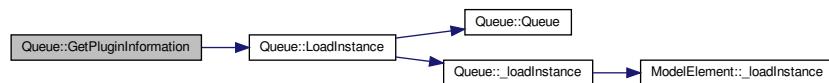
Definition at line 80 of file [Queue.cpp](#).

6.96.4.11 `GetPluginInformation()`

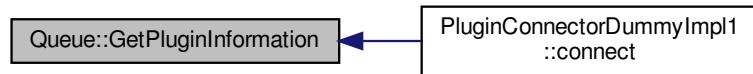
```
PluginInformation * Queue::GetPluginInformation ( ) [static]
```

Definition at line 100 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.96.4.12 initBetweenReplications()

```
void Queue::initBetweenReplications ( )
```

Definition at line 52 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

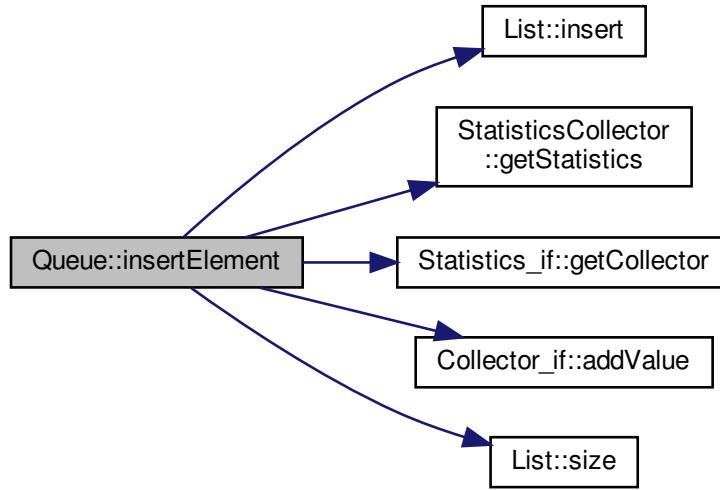


6.96.4.13 insertElement()

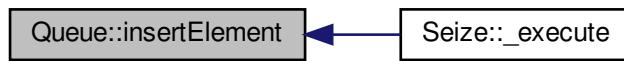
```
void Queue::insertElement ( Waiting * element )
```

Definition at line 39 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



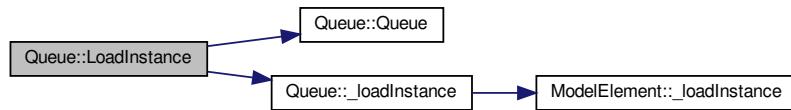
6.96.4.14 LoadInstance()

```

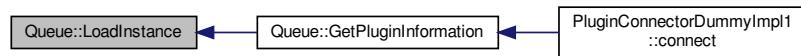
ModelElement * Queue::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
  
```

Definition at line 105 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

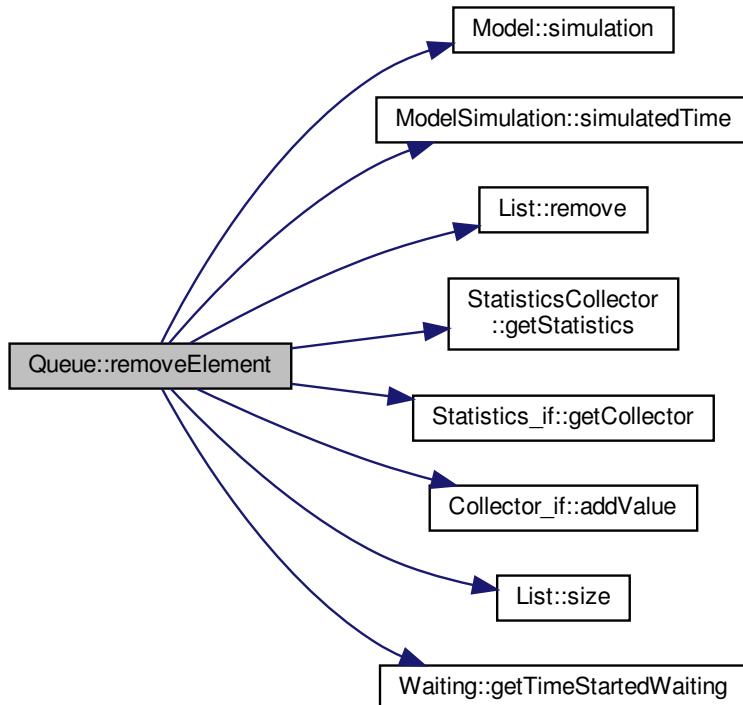


6.96.4.15 removeElement()

```
void Queue::removeElement ( Waiting * element )
```

Definition at line 44 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.96.4.16 setAttributeName()

```
void Queue::setAttributeName ( std::string _attributeName )
```

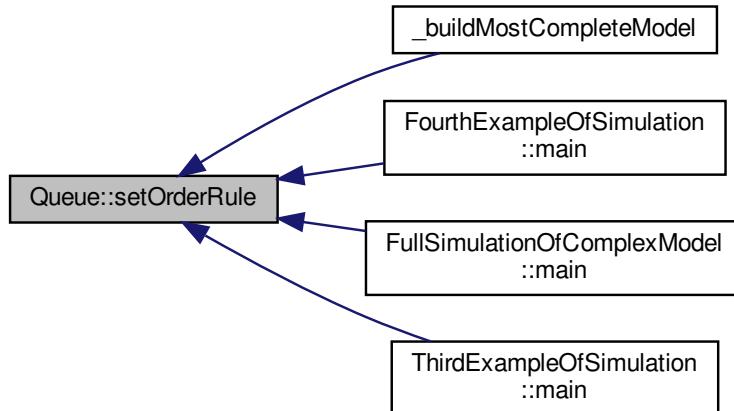
Definition at line 68 of file [Queue.cpp](#).

6.96.4.17 setOrderRule()

```
void Queue::setOrderRule ( OrderRule _orderRule )
```

Definition at line 76 of file [Queue.cpp](#).

Here is the caller graph for this function:



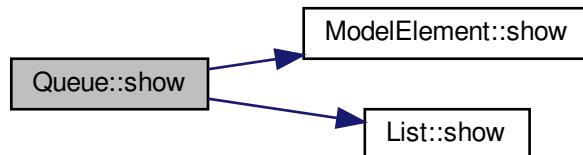
6.96.4.18 show()

```
std::string Queue::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 34 of file [Queue.cpp](#).

Here is the call graph for this function:



6.96.4.19 size()

```
unsigned int Queue::size ( )
```

Definition at line 56 of file [Queue.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.96.4.20 sumAttributesFromWaiting()

```
double Queue::sumAttributesFromWaiting (
    Util::identification attributeID )
```

Definition at line 84 of file [Queue.cpp](#).

Here is the call graph for this function:



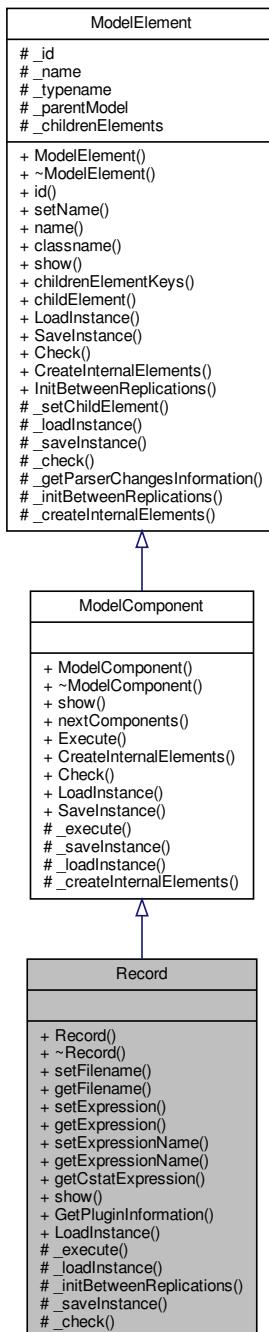
The documentation for this class was generated from the following files:

- [Queue.h](#)
- [Queue.cpp](#)

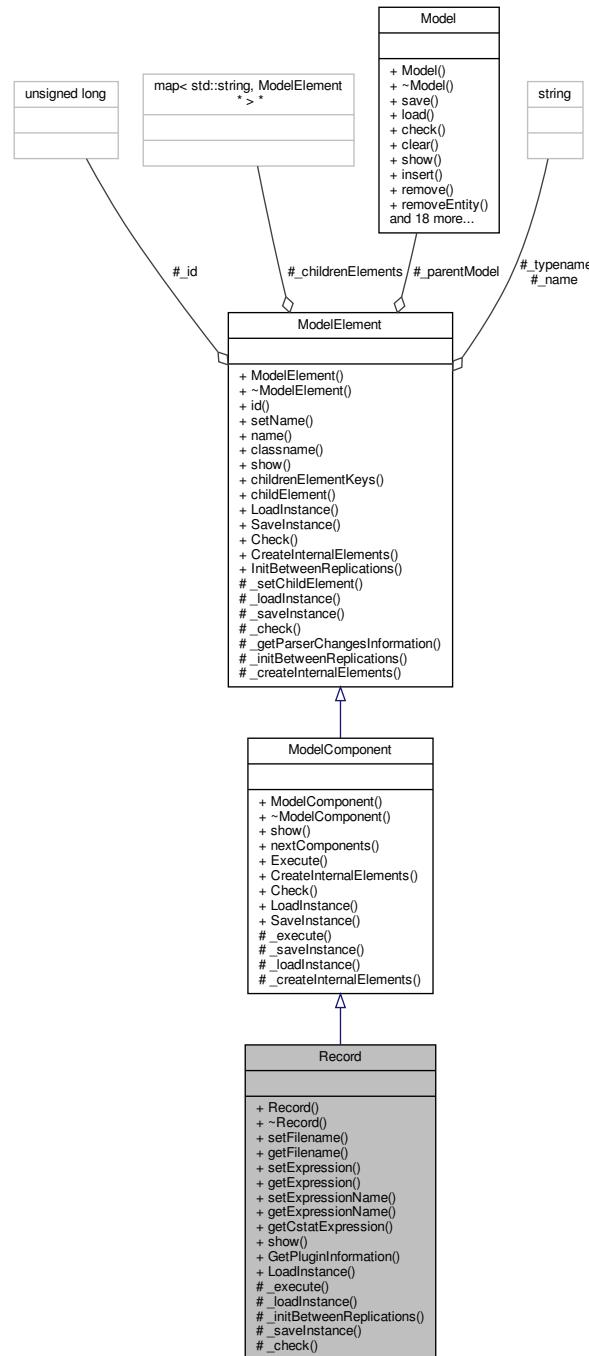
6.97 Record Class Reference

```
#include <Record.h>
```

Inheritance diagram for Record:



Collaboration diagram for Record:



Public Member Functions

- **Record** (`Model *model, std::string name=("")`)
- virtual **~Record** ()
- void **setFilename** (`std::string filename`)
- `std::string getFilename () const`
- void **setExpression** (`std::string expression`)

- std::string `getExpression () const`
- void `setExpressionName (std::string expressionName)`
- std::string `getExpressionName () const`
- `StatisticsCollector * getCstatExpression () const`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.97.1 Detailed Description

Record module DESCRIPTION This module is used to collect statistics in the simulation model. Various types of observational statistics are available, including time between exits through the module, entity statistics (such as time or costing), general observations, and interval statistics (from some time stamp to the current simulation time). A count type of statistic is available as well. Tally and Counter sets can also be specified. TYPICAL USES Collect the number of jobs completed each hour Count how many orders have been late being fulfilled Record the time spent by priority customers in the main check-out line PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Type of observational (tally) or count statistic to be generated. Count will increase or decrease the value of the named statistic by the specified value. Entity Statistics will generate general entity statistics, such as time and costing/duration information. Time Interval will calculate and record the difference between a specified attribute's value and current simulation time. Time Between will track and record the time between entities entering the module. Expression will record the value of the specified expression. Attribute Name Name of the attribute whose value will be used for the interval statistics. Applies only when Type is Interval. Value Value that will be recorded to the observational statistic when Type is Expression or added to the counter when Type is Count. Tally Name This field defines the symbol name of the tally into which the observation is to be recorded. Applies only when Type is Time Interval, Time Between, or Expression. Counter This field defines the symbol name of the counter to Name increment/decrement. Applies only when Type is Counter. Record into Set Check box to specify whether or not a tally or counter set will be used. Tally Set Name Name of the tally set that will be used to record the observational-type statistic. Applies only when Type is Time Interval, Time Between, or Expression. Counter Set Name Name of the counter set that will be used to record the count-type statistic. Applies only when Type is Count. Set Index Index into the tally or counter set.

Definition at line 62 of file `Record.h`.

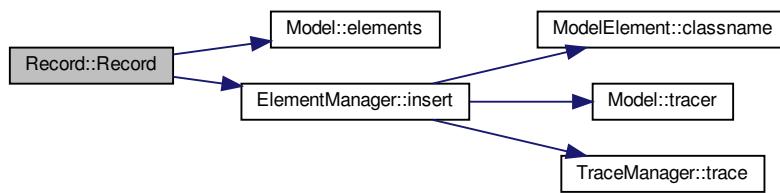
6.97.2 Constructor & Destructor Documentation

6.97.2.1 Record()

```
Record::Record (
    Model * model,
    std::string name = "" )
```

Definition at line 20 of file [Record.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

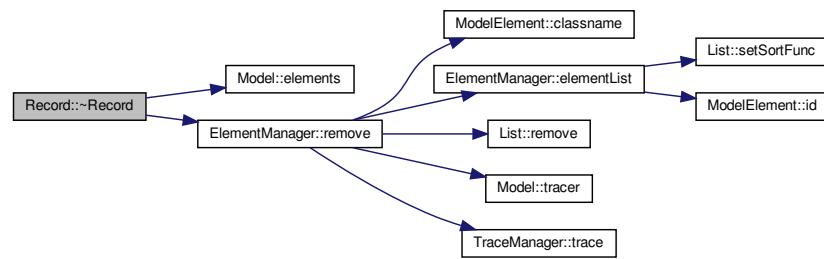


6.97.2.2 ~Record()

```
Record::~Record ( ) [virtual]
```

Definition at line 24 of file [Record.cpp](#).

Here is the call graph for this function:



6.97.3 Member Function Documentation

6.97.3.1 `_check()`

```
bool Record::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 97 of file [Record.cpp](#).

Here is the call graph for this function:



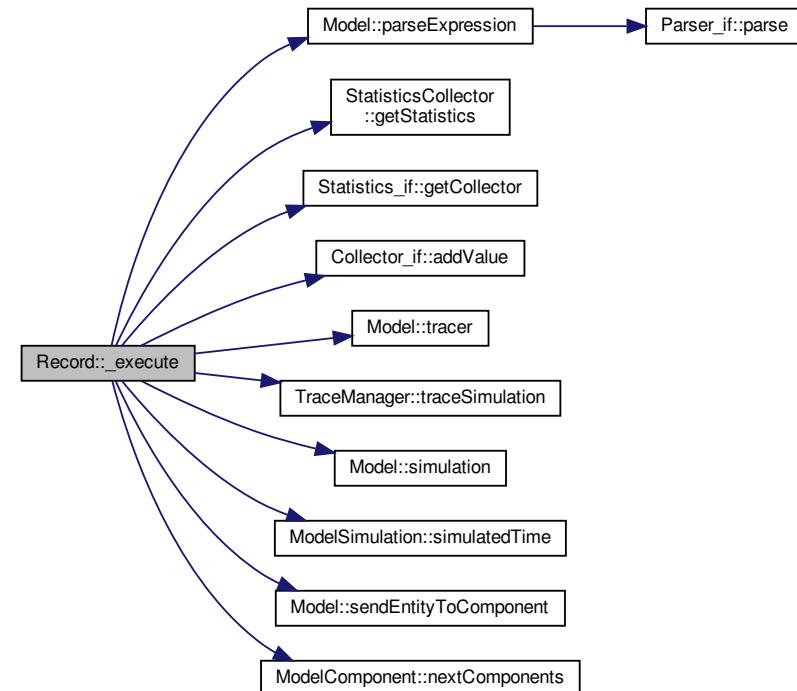
6.97.3.2 `_execute()`

```
void Record::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 64 of file [Record.cpp](#).

Here is the call graph for this function:



6.97.3.3 `_initBetweenReplications()`

```
void Record::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 94 of file [Record.cpp](#).

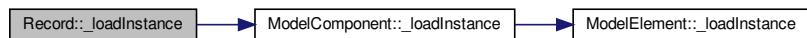
6.97.3.4 `_loadInstance()`

```
bool Record::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 84 of file [Record.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



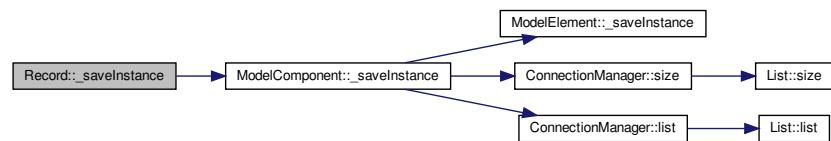
6.97.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Record::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 76 of file [Record.cpp](#).

Here is the call graph for this function:



6.97.3.6 getCstatExpression()

```
StatisticsCollector * Record::getCstatExpression ( ) const
```

Definition at line 44 of file [Record.cpp](#).

6.97.3.7 getExpression()

```
std::string Record::getExpression ( ) const
```

Definition at line 60 of file [Record.cpp](#).

6.97.3.8 getExpressionName()

```
std::string Record::getExpressionName ( ) const
```

Definition at line 40 of file [Record.cpp](#).

6.97.3.9 getFilename()

```
std::string Record::getFilename ( ) const
```

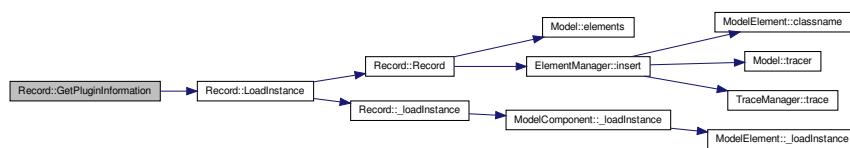
Definition at line 52 of file [Record.cpp](#).

6.97.3.10 GetPluginInformation()

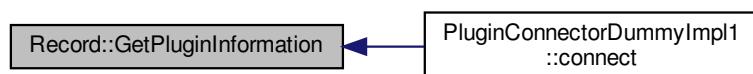
```
PluginInformation * Record::GetPluginInformation ( ) [static]
```

Definition at line 103 of file [Record.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

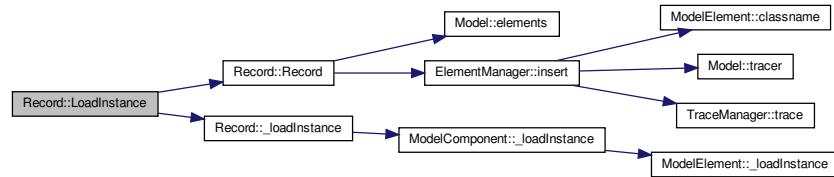


6.97.3.11 LoadInstance()

```
ModelComponent * Record::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 108 of file [Record.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

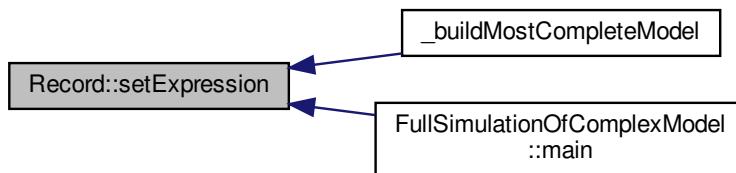


6.97.3.12 setExpression()

```
void Record::setExpression (
    std::string expression )
```

Definition at line 56 of file [Record.cpp](#).

Here is the caller graph for this function:

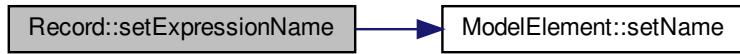


6.97.3.13 setExpressionName()

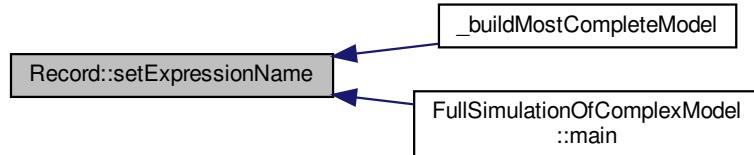
```
void Record::setExpressionName ( std::string expressionName )
```

Definition at line 35 of file [Record.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

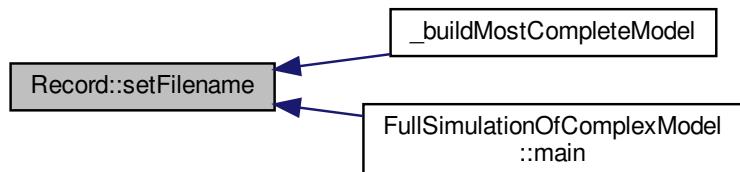


6.97.3.14 setFilename()

```
void Record::setFilename ( std::string filename )
```

Definition at line 48 of file [Record.cpp](#).

Here is the caller graph for this function:



6.97.3.15 show()

```
std::string Record::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [28](#) of file [Record.cpp](#).

Here is the call graph for this function:



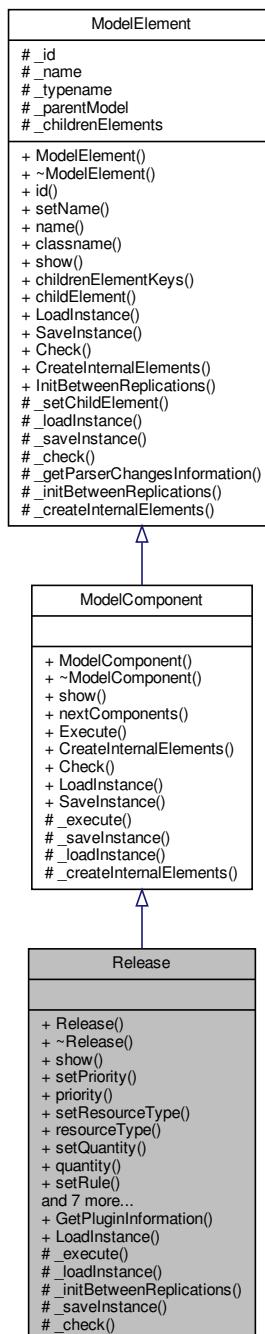
The documentation for this class was generated from the following files:

- [Record.h](#)
- [Record.cpp](#)

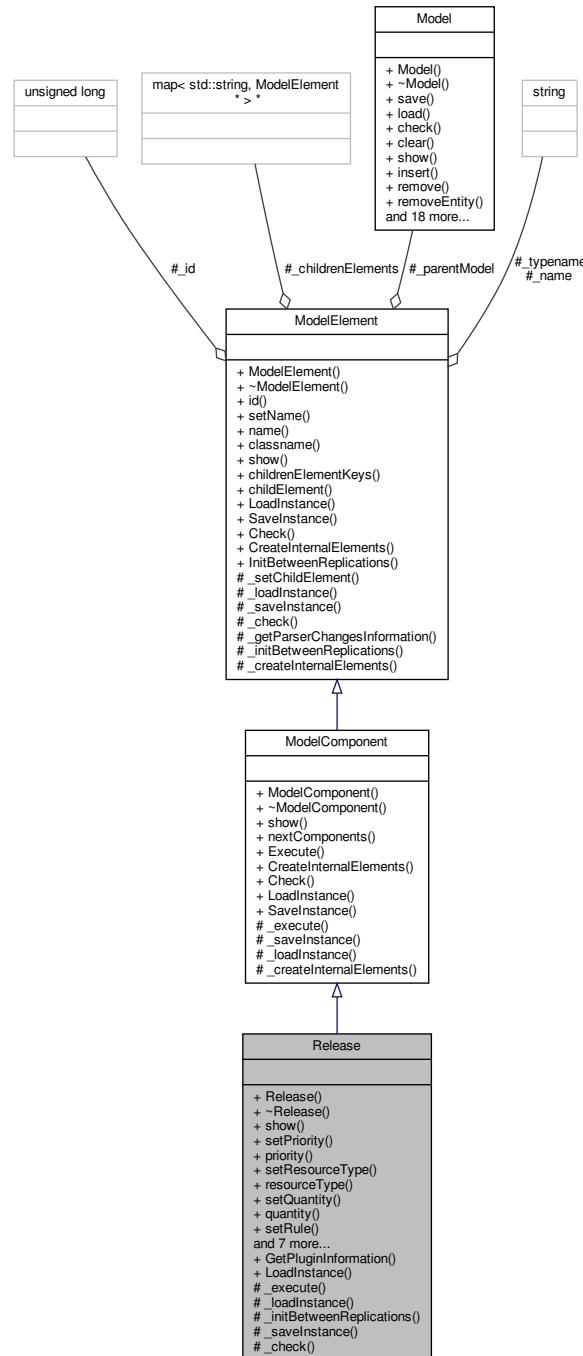
6.98 Release Class Reference

```
#include <Release.h>
```

Inheritance diagram for Release:



Collaboration diagram for Release:



Public Member Functions

- **Release (Model *model, std::string name="")**
- virtual **~Release ()=default**
- virtual std::string **show ()**
- void **setPriority (unsigned short _priority)**
- unsigned short **priority () const**

- void `setResourceType` (`Resource:: ResourceType` `_resourceType`)
- `Resource:: ResourceType` `resourceType () const`
- void `setQuantity` (`std::string` `_quantity`)
- `std::string` `quantity () const`
- void `setRule` (`Resource:: ResourceRule` `_rule`)
- `Resource:: ResourceRule` `rule () const`
- void `setSaveAttribute` (`std::string` `_saveAttribute`)
- `std::string` `saveAttribute () const`
- void `setResource` (`Resource *``_resource`)
- `Resource *` `resource () const`
- void `set resourceName` (`std::string` `resourceName`) `throw ()`
- `std::string` `resourceName () const`

Static Public Member Functions

- static `PluginInformation *` `GetPluginInformation ()`
- static `ModelComponent *` `LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.98.1 Detailed Description

Release module DESCRIPTION The **Release** module is used to release units of a resource that an entity previously has seized. This module may be used to release individual resources or may be used to release resources within a set. For each resource to be released, the name and quantity to release are specified. When the entity enters the **Release** module, it gives up control of the specified resource(s). Any entities waiting in queues for those resources will gain control of the resources immediately. TYPICAL USES Finishing a customer order (release the operator) Completing a tax return (release the accountant) Leaving the hospital (release the doctor, nurse, hospital room) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Type of resource for releasing, either specifying a particular resource, or selecting from a pool of resources (that is, a resource set). The resource name may also be specified by an expression or attribute value. **Resource** Name Name of the resource that will be released. **Set** Name Name of the resource set from which a member will be released. **Attribute** Name Name of the attribute that specifies the resource name to be released. Expression Name of the expression that specifies the name of the resource to be released. Quantity Number of resources of a given name or from a given set that will be released. For sets, this value specifies only the number of a selected resource that will be released (based on the resource's capacity), not the number of members to be released within the set. **Release Rule** Method of determining which resource within a set to release. Last Member Seized and First Member Seized will release the last/first member from within the set that was seized. Specific member indicates that a member number or attribute (with a member number value) will be used to specify the member to release. **Set Index** Member index of the resource set that the entity will release.

Definition at line 63 of file `Release.h`.

6.98.2 Constructor & Destructor Documentation

6.98.2.1 Release()

```
Release::Release (
    Model * model,
    std::string name = "" )
```

Definition at line 20 of file [Release.cpp](#).

Here is the caller graph for this function:



6.98.2.2 ~Release()

```
virtual Release::~Release () [virtual], [default]
```

6.98.3 Member Function Documentation

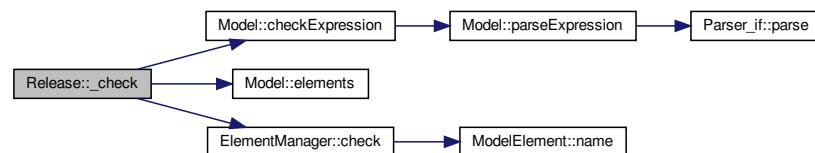
6.98.3.1 _check()

```
bool Release::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 127 of file [Release.cpp](#).

Here is the call graph for this function:



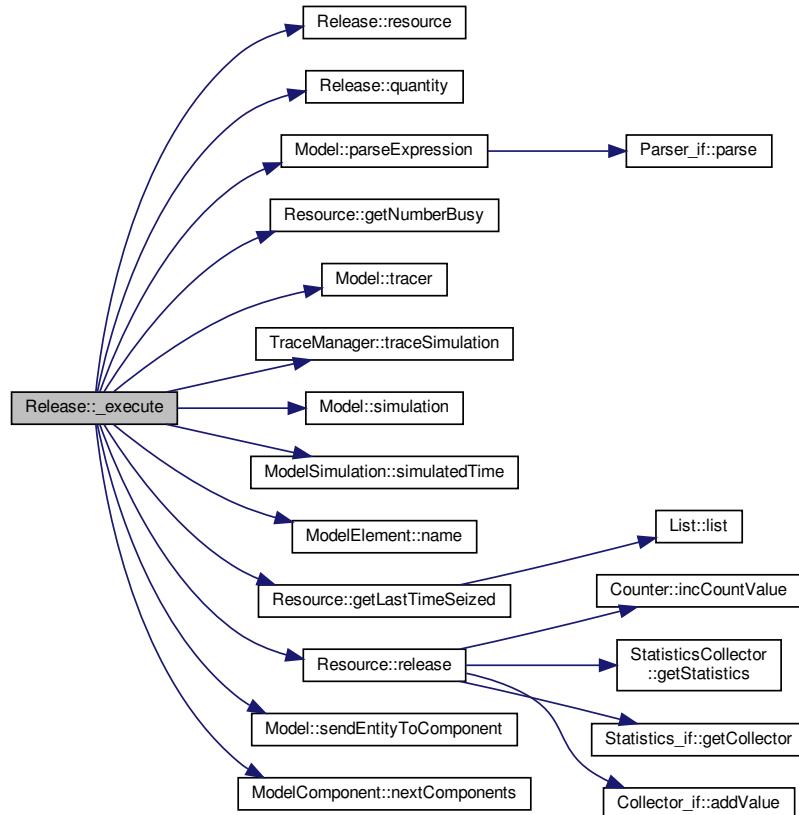
6.98.3.2 `_execute()`

```
void Release::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 79 of file [Release.cpp](#).

Here is the call graph for this function:



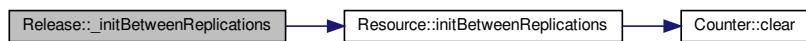
6.98.3.3 `_initBetweenReplications()`

```
void Release::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 93 of file [Release.cpp](#).

Here is the call graph for this function:



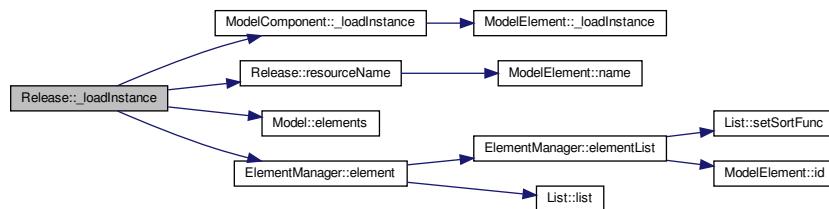
6.98.3.4 `_loadInstance()`

```
bool Release::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 97 of file [Release.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



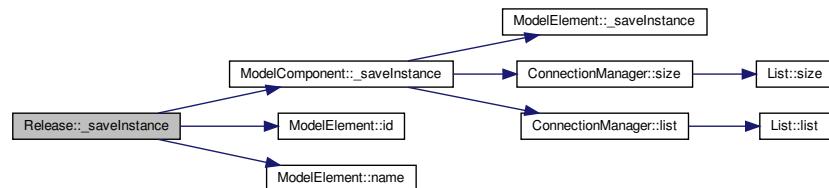
6.98.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Release::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 114 of file [Release.cpp](#).

Here is the call graph for this function:

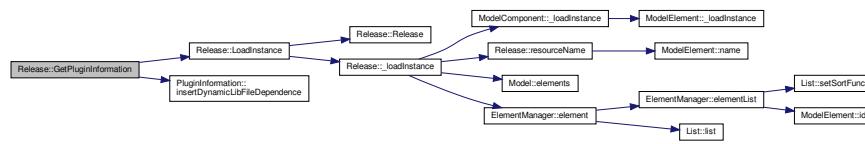


6.98.3.6 GetPluginInformation()

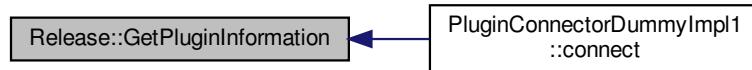
```
PluginInformation * Release::GetPluginInformation ( ) [static]
```

Definition at line 148 of file [Release.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

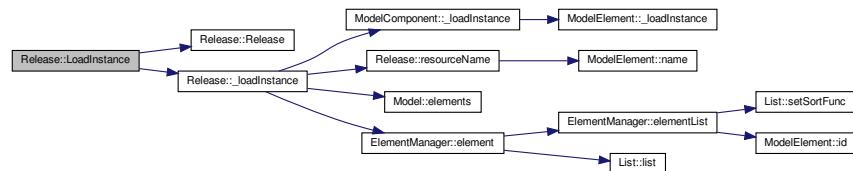


6.98.3.7 LoadInstance()

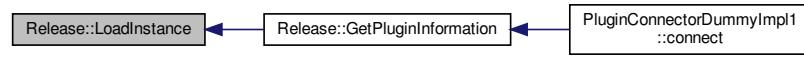
```
ModelComponent * Release::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 154 of file [Release.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.98.3.8 priority()

```
unsigned short Release::priority () const
```

Definition at line 35 of file [Release.cpp](#).

6.98.3.9 quantity()

```
std::string Release::quantity () const
```

Definition at line 51 of file [Release.cpp](#).

Here is the caller graph for this function:

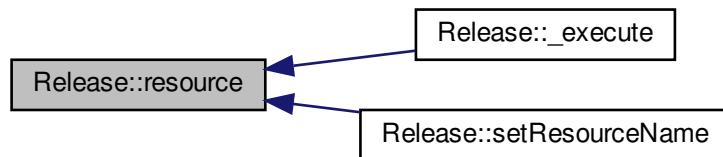


6.98.3.10 resource()

```
Resource * Release::resource () const
```

Definition at line 75 of file [Release.cpp](#).

Here is the caller graph for this function:



6.98.3.11 resourceName()

```
std::string Release::resourceName( ) const
```

Definition at line 144 of file [Release.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.98.3.12 resourceType()

```
Resource::ResourceType Release::resourceType( ) const
```

Definition at line 43 of file [Release.cpp](#).

6.98.3.13 rule()

```
Resource::ResourceRule Release::rule( ) const
```

Definition at line 59 of file [Release.cpp](#).

6.98.3.14 saveAttribute()

```
std::string Release::saveAttribute( ) const
```

Definition at line 67 of file [Release.cpp](#).

6.98.3.15 setPriority()

```
void Release::setPriority (
    unsigned short _priority )
```

Definition at line 31 of file [Release.cpp](#).

6.98.3.16 setQuantity()

```
void Release::setQuantity (
    std::string _quantity )
```

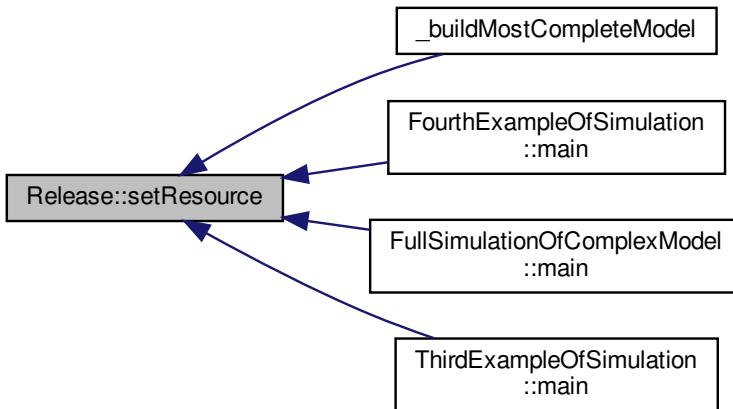
Definition at line 47 of file [Release.cpp](#).

6.98.3.17 setResource()

```
void Release::setResource (
    Resource * _resource )
```

Definition at line 71 of file [Release.cpp](#).

Here is the caller graph for this function:

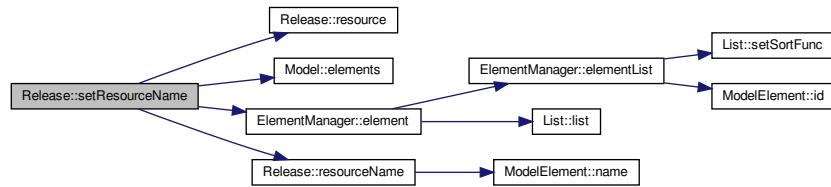


6.98.3.18 setResourceName()

```
void Release::setResourceName (
    std::string resourceName ) throw ()
```

Definition at line 135 of file [Release.cpp](#).

Here is the call graph for this function:



6.98.3.19 setResourceType()

```
void Release::setResourceType (
    Resource::ResourceType _resourceType )
```

Definition at line 39 of file [Release.cpp](#).

6.98.3.20 setRule()

```
void Release::setRule (
    Resource::ResourceRule _rule )
```

Definition at line 55 of file [Release.cpp](#).

6.98.3.21 setSaveAttribute()

```
void Release::setSaveAttribute (
    std::string _saveAttribute )
```

Definition at line 63 of file [Release.cpp](#).

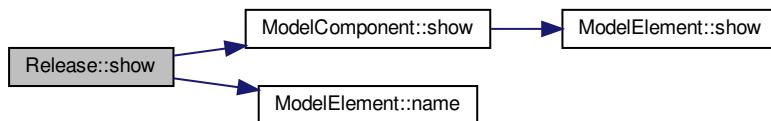
6.98.3.22 show()

```
std::string Release::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [24](#) of file [Release.cpp](#).

Here is the call graph for this function:



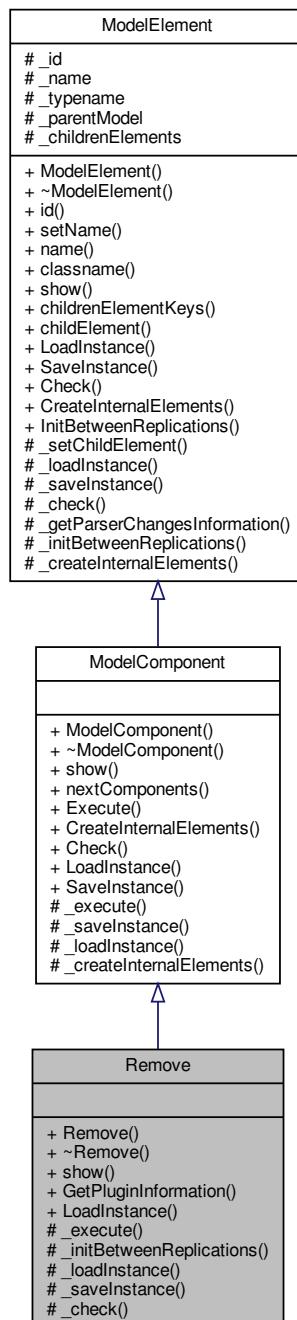
The documentation for this class was generated from the following files:

- [Release.h](#)
- [Release.cpp](#)

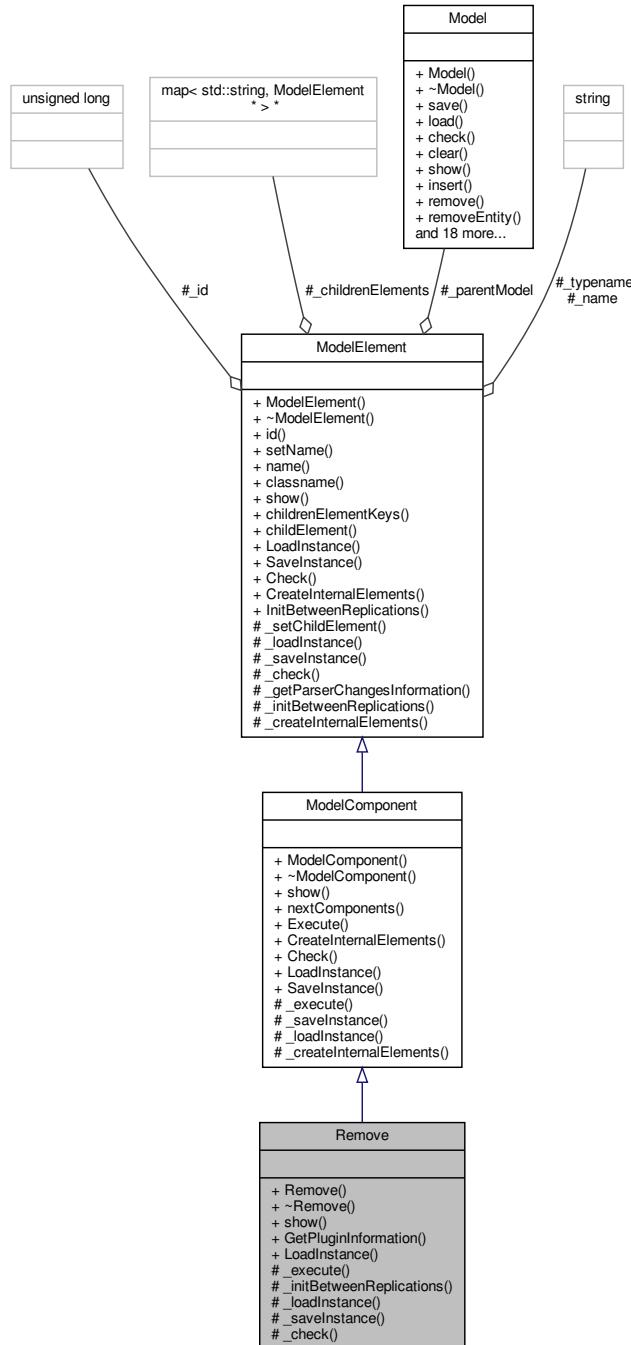
6.99 Remove Class Reference

```
#include <Remove.h>
```

Inheritance diagram for Remove:



Collaboration diagram for Remove:



Public Member Functions

- `Remove (Model *model, std::string name=""")`
- virtual `~Remove ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.99.1 Detailed Description

Remove module DESCRIPTION The `Remove` module removes a single entity from a specified position in a queue and sends it to a designated module. When an entity arrives at a `Remove` module, it removes the entity from the specified queue and sends it to the connected module. The rank of the entity signifies the location of the entity within the queue. The entity that caused the removal proceeds to the next module specified and is processed before the removed entity. TYPICAL USES Removing an order from a queue that is due to be completed next Calling a patient from a waiting room for an examination Retrieving the next order to be processed from a pile of documents Prompt Description Name Unique module identifier displayed on the module shape. Queue Name Name of the queue from which the entity will be removed. Rank of Entity Rank of the entity to remove from within the queue.

Definition at line 37 of file `Remove.h`.

6.99.2 Constructor & Destructor Documentation

6.99.2.1 Remove()

```
Remove::Remove (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file `Remove.cpp`.

Here is the caller graph for this function:



6.99.2.2 ~Remove()

```
virtual Remove::~Remove ( ) [virtual], [default]
```

6.99.3 Member Function Documentation

6.99.3.1 _check()

```
bool Remove::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Remove.cpp](#).

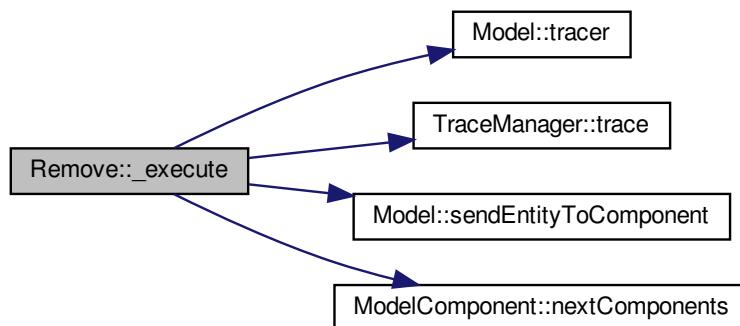
6.99.3.2 _execute()

```
void Remove::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Remove.cpp](#).

Here is the call graph for this function:



6.99.3.3 `_initBetweenReplications()`

```
void Remove::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [48](#) of file [Remove.cpp](#).

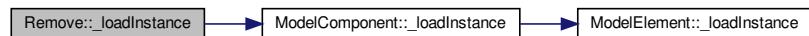
6.99.3.4 `_loadInstance()`

```
bool Remove::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

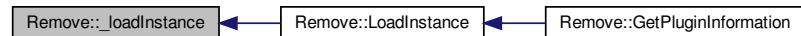
Reimplemented from [ModelComponent](#).

Definition at line [40](#) of file [Remove.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



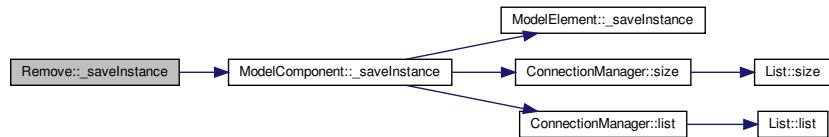
6.99.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Remove::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [51](#) of file [Remove.cpp](#).

Here is the call graph for this function:

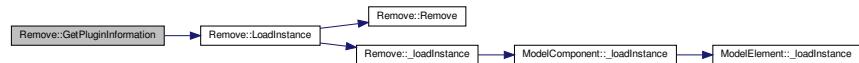


6.99.3.6 GetPluginInformation()

```
PluginInformation * Remove::GetPluginInformation( ) [static]
```

Definition at line 63 of file [Remove.cpp](#).

Here is the call graph for this function:

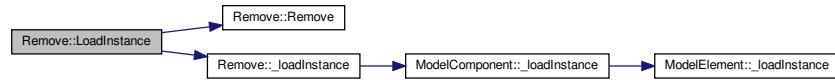


6.99.3.7 LoadInstance()

```
ModelComponent * Remove::LoadInstance(
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Remove.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



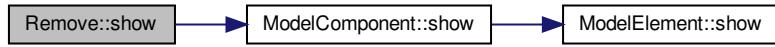
6.99.3.8 show()

```
std::string Remove::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [21](#) of file [Remove.cpp](#).

Here is the call graph for this function:



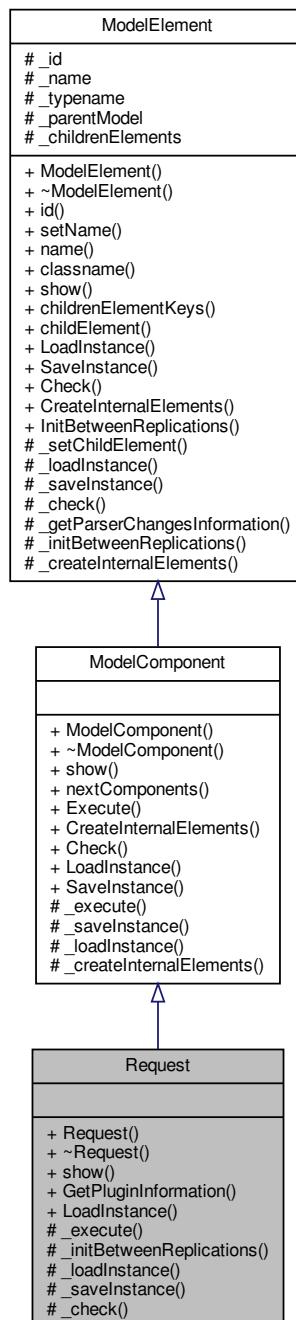
The documentation for this class was generated from the following files:

- [Remove.h](#)
- [Remove.cpp](#)

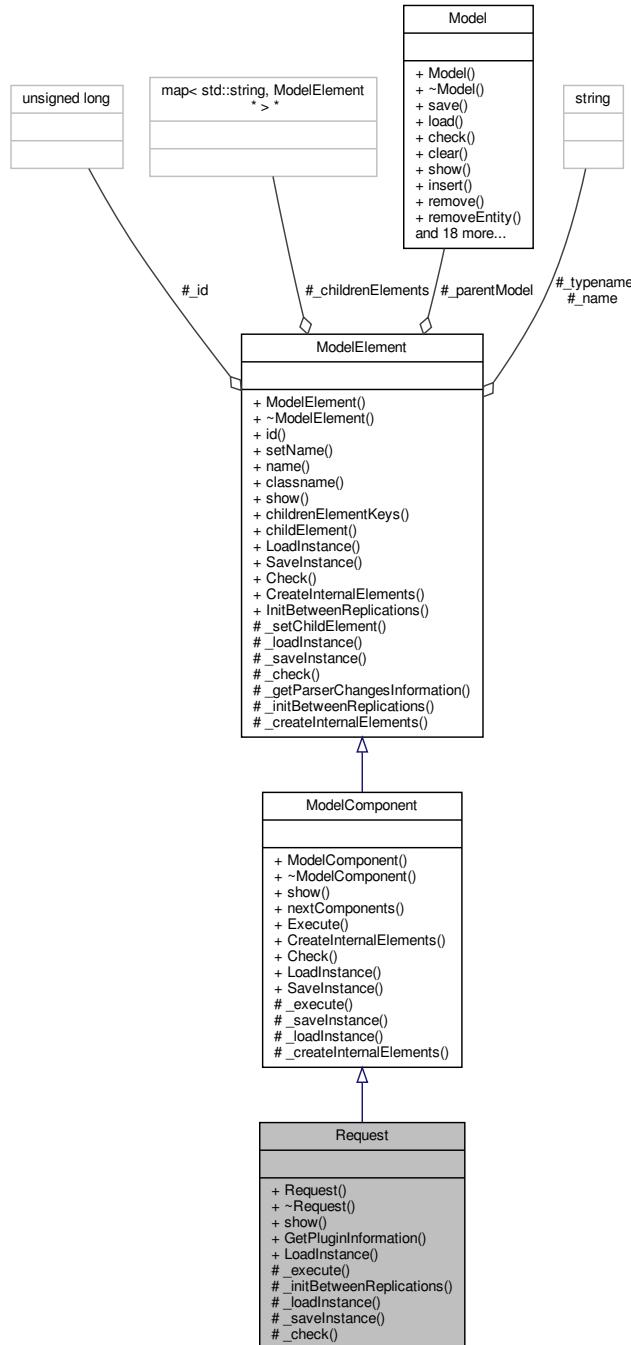
6.100 Request Class Reference

```
#include <Request.h>
```

Inheritance diagram for Request:



Collaboration diagram for Request:



Public Member Functions

- [Request \(Model *model, std::string name=""\)](#)
- virtual [~Request \(\)=default](#)
- virtual std::string [show \(\)](#)

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.100.1 Detailed Description

Request module DESCRIPTION The `Request` module assigns a transporter unit to an entity and moves the unit to the entity's station location. A specific transporter unit may be specified or the selection may occur based on a rule. When the entity arrives at the `Request` module, it is allocated a transporter when one is available. The entity remains at the `Request` module until the transporter unit has reached the entity's station. The entity then moves out of the `Request` module. TYPICAL USES A sanded part requests a cart to take it to the paint shop. Customers in a restaurant are ready to order and thus request a waiter. PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Transporter Name Name of the transporter to allocate. Selection Rule Rule for determining which of the transporters to allocate to the entity, either Cyclical, Random, Preferred Order, Specific Member, Largest Distance, or Smallest Distance. Save Attribute Denotes the attribute name that will store the unit number of the selected transporter. Unit Number Determines the specific transporter unit in the transporter set to request. Priority Priority value of the entity waiting at this module for the transporter specified if one or more entities are waiting for the same transporter anywhere in the model. Entity Location The location to which the transporter will move after being allocated. For free-path transporters, the entity location must be specified as a station. For guided transporters, the entity location may be specified as a station, intersection, or network link. Velocity Speed at which the transporter is moved to the entity location in length units per unit time. The unit time is specified in the Units field. Units Velocity time units.

Definition at line 55 of file `Request.h`.

6.100.2 Constructor & Destructor Documentation

6.100.2.1 Request()

```
Request::Request (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file `Request.cpp`.

Here is the caller graph for this function:



6.100.2.2 ~Request()

```
virtual Request::~Request ( ) [virtual], [default]
```

6.100.3 Member Function Documentation

6.100.3.1 _check()

```
bool Request::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Request.cpp](#).

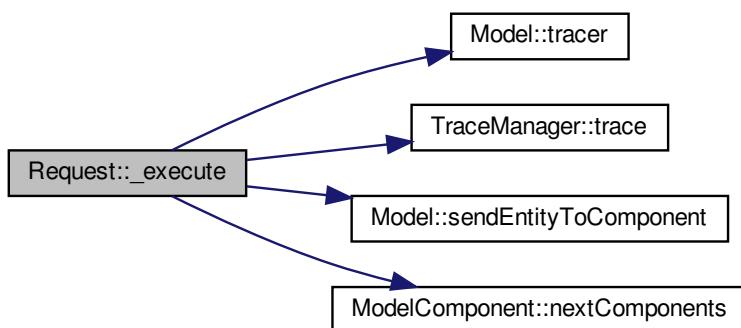
6.100.3.2 _execute()

```
void Request::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Request.cpp](#).

Here is the call graph for this function:



6.100.3.3 `_initBetweenReplications()`

```
void Request::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [48](#) of file [Request.cpp](#).

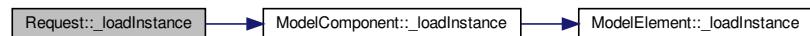
6.100.3.4 `_loadInstance()`

```
bool Request::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

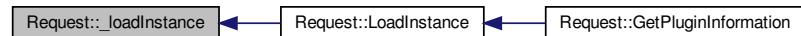
Reimplemented from [ModelComponent](#).

Definition at line [40](#) of file [Request.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



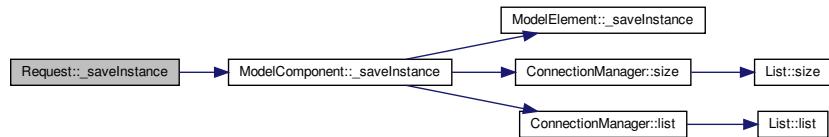
6.100.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Request::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [51](#) of file [Request.cpp](#).

Here is the call graph for this function:

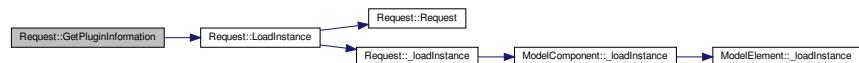


6.100.3.6 GetPluginInformation()

```
PluginInformation * Request::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [Request.cpp](#).

Here is the call graph for this function:

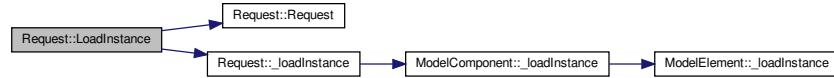


6.100.3.7 LoadInstance()

```
ModelComponent * Request::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Request.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.100.3.8 show()

```
std::string Request::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Request.cpp](#).

Here is the call graph for this function:



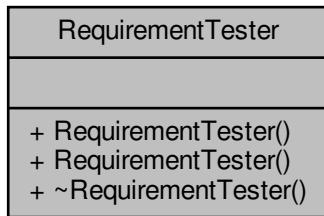
The documentation for this class was generated from the following files:

- [Request.h](#)
- [Request.cpp](#)

6.101 RequirementTester Class Reference

```
#include <RequirementTester.h>
```

Collaboration diagram for RequirementTester:



Public Member Functions

- [RequirementTester \(\)](#)
- [RequirementTester \(const RequirementTester &orig\)](#)
- [virtual ~RequirementTester \(\)](#)

6.101.1 Detailed Description

Definition at line 17 of file [RequirementTester.h](#).

6.101.2 Constructor & Destructor Documentation

6.101.2.1 RequirementTester() [1/2]

```
RequirementTester::RequirementTester ( )
```

Definition at line 16 of file [RequirementTester.cpp](#).

6.101.2.2 RequirementTester() [2/2]

```
RequirementTester::RequirementTester (
    const RequirementTester & orig )
```

Definition at line 19 of file [RequirementTester.cpp](#).

6.101.2.3 ~RequirementTester()

```
RequirementTester::~RequirementTester ( ) [virtual]
```

Definition at line 22 of file [RequirementTester.cpp](#).

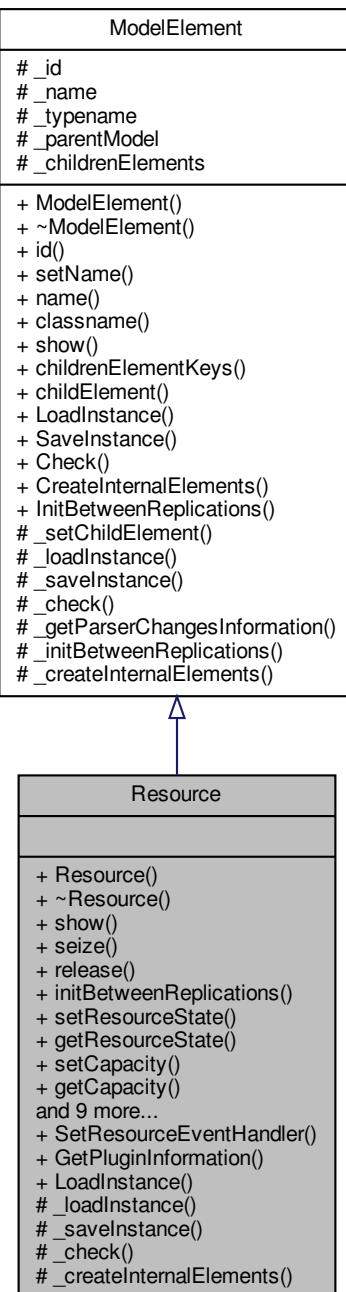
The documentation for this class was generated from the following files:

- [RequirementTester.h](#)
- [RequirementTester.cpp](#)

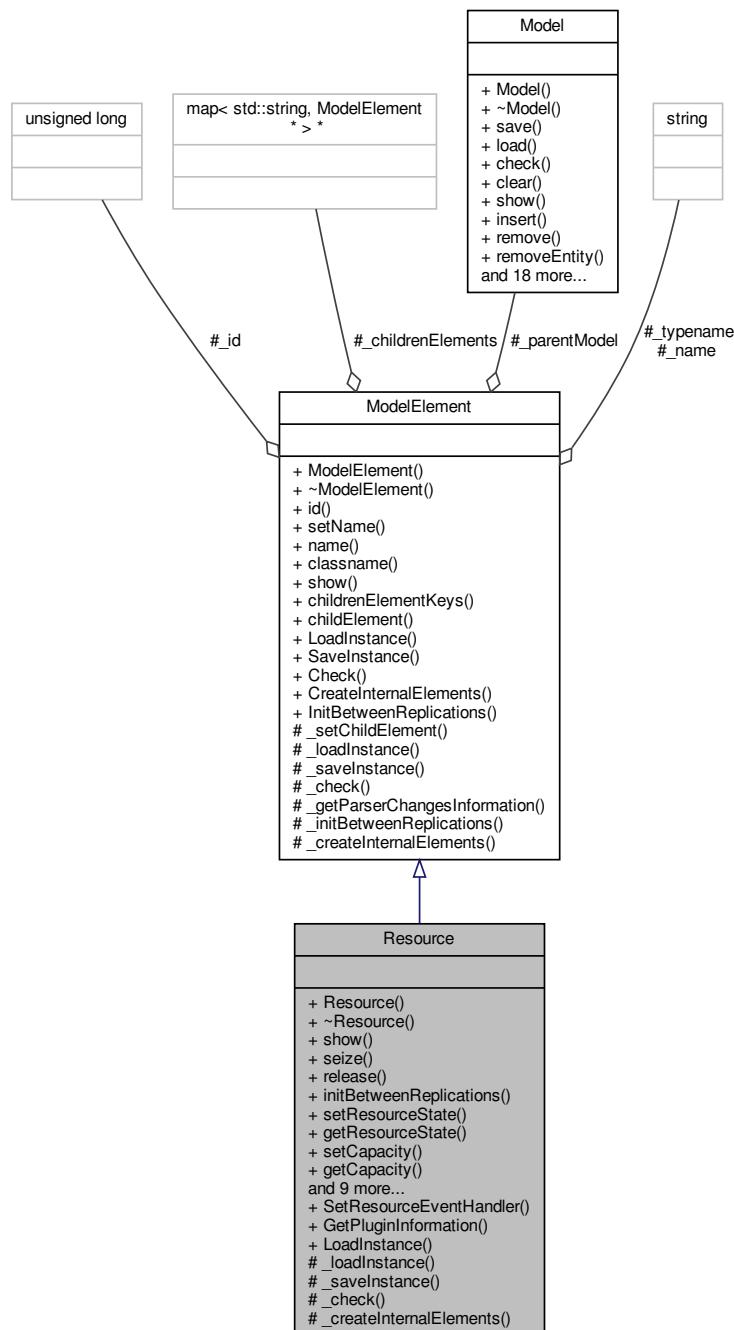
6.102 Resource Class Reference

```
#include <Resource.h>
```

Inheritance diagram for Resource:



Collaboration diagram for Resource:



Public Types

- enum **ResourceType** : int { **ResourceType::SET** = 1, **ResourceType::RESOURCE** = 2 }
- enum **ResourceRule** : int {
 ResourceRule::RANDOM = 1, **ResourceRule::CICLICAL** = 2, **ResourceRule::ESPECIFIC** = 3, **ResourceRule::SMALLESTBUSY** = 4,
 ResourceRule::LARGESTREMAININGCAPACITY = 5
 }

- enum `ResourceState` : int {
 `ResourceState::IDLE` = 1, `ResourceState::BUSY` = 2, `ResourceState::FAILED` = 3, `ResourceState::INACTIVE` = 4,
 `ResourceState::OTHER` = 5 }
- `typedef std::function< void(Resource *) > ResourceEventHandler`

Public Member Functions

- `Resource (Model *model, std::string name="")`
- virtual `~Resource ()`
- virtual `std::string show ()`
- void `seize (unsigned int quantity, double tnow)`
- void `release (unsigned int quantity, double tnow)`
- void `initBetweenReplications ()`
- void `setResourceState (ResourceState _resourceState)`
- `Resource::ResourceState getResourceState () const`
- void `setCapacity (unsigned int _capacity)`
- unsigned int `getCapacity () const`
- void `setCostBusyHour (double _costBusyHour)`
- double `getCostBusyHour () const`
- void `setCostIdleHour (double _costIdleHour)`
- double `getCostIdleHour () const`
- void `setCostPerUse (double _costPerUse)`
- double `getCostPerUse () const`
- unsigned int `getNumberBusy () const`
- void `addResourceEventHandler (ResourceEventHandler eventHandler)`
- double `getLastTimeSeized () const`

Static Public Member Functions

- template<typename Class >
 static `ResourceEventHandler SetResourceEventHandler (void(Class::*function)(Resource *), Class *object)`
- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

6.102.1 Detailed Description

Resource module DESCRIPTION This data module defines the resources in the simulation system, including costing information and resource availability. Resources may have a fixed capacity that does not vary over the simulation run or may operate based on a schedule. **Resource** failures and states can also be specified in this module. TYPICAL USES Equipment (machinery, cash register, phone line) People (clerical, order processing, sales clerks, operators) PROMPTS Prompt Description Name The name of the resource whose characteristics are being defined. This name must be unique. Type Method for determining the capacity for a resource. Fixed Capacity will not change during the simulation run. Based on **Schedule** signifies that a **Schedule** module is used to specify the capacity and duration information for the resource. Capacity Number of resource units of a given name that are available to the system for processing. Applies only when Type is Fixed Capacity. **Schedule** Name Identifies the name of the schedule to be used by the resource. The schedule defines the capacity of a resource for a given period of time. Applies only when type is **Schedule**. **Schedule** Rule Dictates when the actual capacity change is to occur when a decrease in capacity is required for a busy resource unit. Applies only when Type is **Schedule**. Busy/Hour Cost per hour of a resource that is processing an entity. The resource becomes busy when it is originally allocated to an entity and becomes idle when it is released. During the time when it is busy, cost will accumulate based on the busy/hour cost. The busy cost per hour is automatically converted to the appropriate base time unit specified within the Replication Parameters page of the Run > Setup menu item. Idle/Hour Cost per hour of a resource that is idle. The resource is idle while it is not processing an entity. During the time when it is idle, cost will accumulate based on the idle/hour cost. The idle cost per hour is automatically converted to the appropriate base time unit specified within the Replication Parameters page of the Run > Setup menu item. Per Use Cost of a resource on a usage basis, regardless of the time for which it is used. Each time the resource is allocated to an entity, it will incur a per-use cost. StateSet Name Name of states that the resource may be assigned during the simulation run. Initial State Initial state of a resource. If specified, the name must be defined within the repeat group of state names. This field is shown only when a StateSet Name is defined. Failures Lists all failures that will be associated with the resource. Failure Name—Name of the failure associated with the resource. Failure Rule—Behavior that should occur when a failure is to occur for a busy resource unit. Report Statistics Specifies whether or not statistics will be collected automatically and stored in the report database for this resource.

Definition at line 79 of file [Resource.h](#).

6.102.2 Member Typedef Documentation

6.102.2.1 ResourceEventHandler

```
typedef std::function<void(Resource*)> Resource::ResourceEventHandler
```

Definition at line 81 of file [Resource.h](#).

6.102.3 Member Enumeration Documentation

6.102.3.1 ResourceRule

```
enum Resource::ResourceRule : int [strong]
```

Enumerator

RANDOM	
CICLICAL	
ESPECIFIC	
SMALLESTBUSY	
LARGESTREMAININGCAPACITY	

Definition at line 92 of file [Resource.h](#).

6.102.3.2 ResourceState

```
enum Resource::ResourceState : int [strong]
```

Enumerator

IDLE	
BUSY	
FAILED	
INACTIVE	
OTHER	

Definition at line 96 of file [Resource.h](#).

6.102.3.3 ResourceType

```
enum Resource::ResourceType : int [strong]
```

Enumerator

SET	
RESOURCE	

Definition at line 88 of file [Resource.h](#).

6.102.4 Constructor & Destructor Documentation**6.102.4.1 Resource()**

```
Resource::Resource (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file [Resource.cpp](#).

Here is the caller graph for this function:



6.102.4.2 ~Resource()

```
Resource::~Resource ( ) [virtual]
```

Definition at line 30 of file [Resource.cpp](#).

6.102.5 Member Function Documentation

6.102.5.1 _check()

```
bool Resource::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 168 of file [Resource.cpp](#).

6.102.5.2 _createInternalElements()

```
void Resource::_createInternalElements ( ) [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelElement](#).

Definition at line 172 of file [Resource.cpp](#).

6.102.5.3 `_loadInstance()`

```
bool Resource::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 148 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.102.5.4 `_saveInstance()`

```
std::map< std::string, std::string * > * Resource::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 159 of file [Resource.cpp](#).

Here is the call graph for this function:

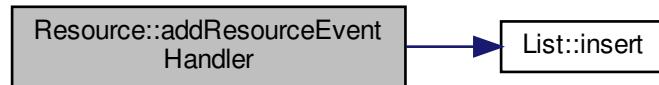


6.102.5.5 addResourceEventHandler()

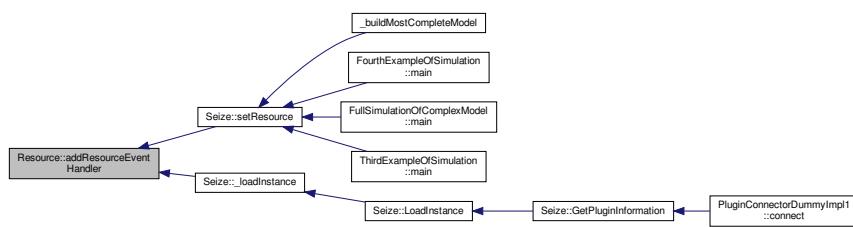
```
void Resource::addResourceEventHandler (
    ResourceEventHandler eventHandler )
```

Definition at line 120 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

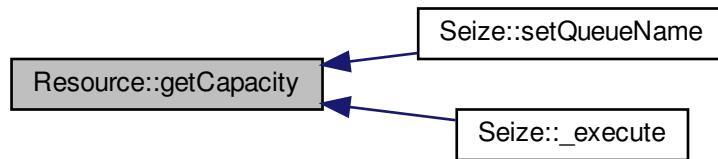


6.102.5.6 getCapacity()

```
unsigned int Resource::getCapacity ( ) const
```

Definition at line 88 of file [Resource.cpp](#).

Here is the caller graph for this function:



6.102.5.7 `getCostBusyHour()`

```
double Resource::getCostBusyHour ( ) const
```

Definition at line [96](#) of file [Resource.cpp](#).

6.102.5.8 `getCostIdleHour()`

```
double Resource::getCostIdleHour ( ) const
```

Definition at line [104](#) of file [Resource.cpp](#).

6.102.5.9 `getCostPerUse()`

```
double Resource::getCostPerUse ( ) const
```

Definition at line [112](#) of file [Resource.cpp](#).

6.102.5.10 `getLastTimeSeized()`

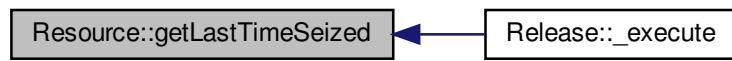
```
double Resource::getLastTimeSeized ( ) const
```

Definition at line [124](#) of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

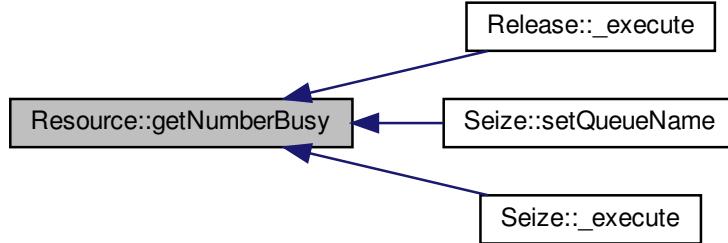


6.102.5.11 getNumberBusy()

```
unsigned int Resource::getNumberBusy ( ) const
```

Definition at line 116 of file [Resource.cpp](#).

Here is the caller graph for this function:

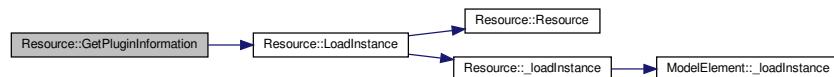


6.102.5.12 GetPluginInformation()

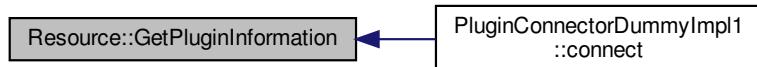
```
PluginInformation * Resource::GetPluginInformation ( ) [static]
```

Definition at line 134 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.102.5.13 getResourceState()

```
Resource::ResourceState Resource::getResourceState ( ) const
```

Definition at line 80 of file [Resource.cpp](#).

6.102.5.14 initBetweenReplications()

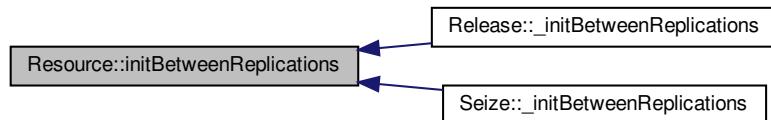
```
void Resource::initBetweenReplications ( )
```

Definition at line 69 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

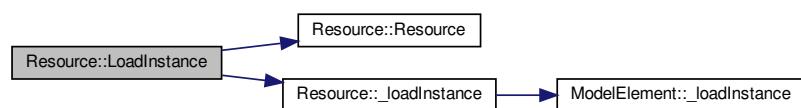


6.102.5.15 LoadInstance()

```
ModelElement * Resource::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 138 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

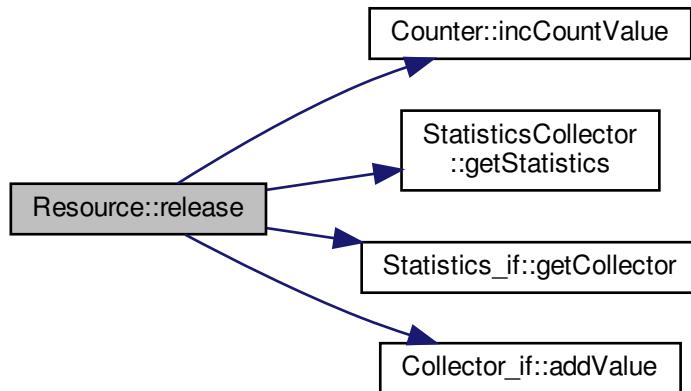


6.102.5.16 release()

```
void Resource::release (
    unsigned int quantity,
    double tnow )
```

Definition at line 51 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

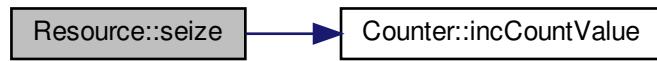


6.102.5.17 seize()

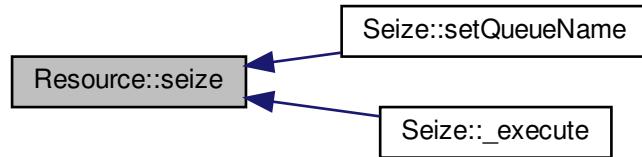
```
void Resource::seize (
    unsigned int quantity,
    double tnow )
```

Definition at line 44 of file [Resource.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

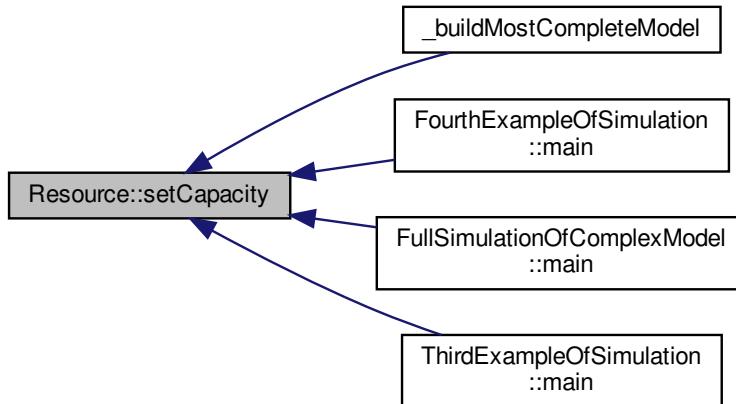


6.102.5.18 setCapacity()

```
void Resource::setCapacity (
    unsigned int _capacity )
```

Definition at line 84 of file [Resource.cpp](#).

Here is the caller graph for this function:



6.102.5.19 `setCostBusyHour()`

```
void Resource::setCostBusyHour (
    double _costBusyHour )
```

Definition at line 92 of file [Resource.cpp](#).

6.102.5.20 `setCostIdleHour()`

```
void Resource::setCostIdleHour (
    double _costIdleHour )
```

Definition at line 100 of file [Resource.cpp](#).

6.102.5.21 `setCostPerUse()`

```
void Resource::setCostPerUse (
    double _costPerUse )
```

Definition at line 108 of file [Resource.cpp](#).

6.102.5.22 SetResourceEventHandler()

```
template<typename Class >
static ResourceEventHandler Resource::SetResourceEventHandler (
    void(Class::*)(Resource *) function,
    Class * object ) [inline], [static]
```

Definition at line 84 of file [Resource.h](#).

6.102.5.23 setResourceState()

```
void Resource::setResourceState (
    ResourceState _resourceState )
```

Definition at line 76 of file [Resource.cpp](#).

6.102.5.24 show()

```
std::string Resource::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 35 of file [Resource.cpp](#).

Here is the call graph for this function:



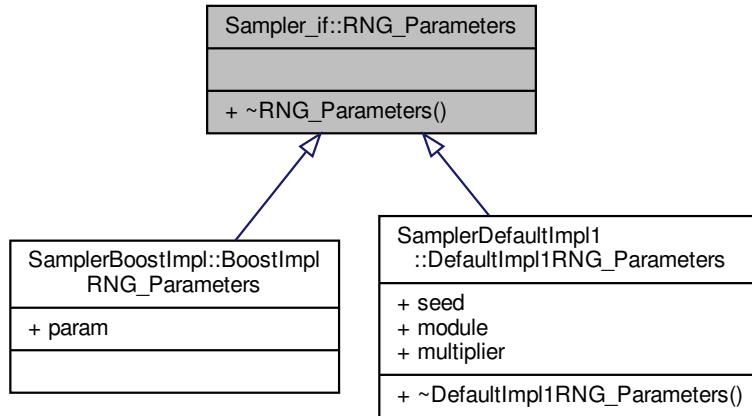
The documentation for this class was generated from the following files:

- [Resource.h](#)
- [Resource.cpp](#)

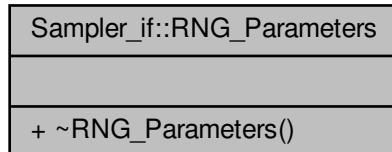
6.103 Sampler_if::RNG_Parameters Struct Reference

```
#include <Sampler_if.h>
```

Inheritance diagram for Sampler_if::RNG_Parameters:



Collaboration diagram for Sampler_if::RNG_Parameters:



Public Member Functions

- virtual `~RNG_Parameters ()=default`

6.103.1 Detailed Description

class that encapsulates attributes required to generate random numbers, which depends on the generation method used.

Definition at line 26 of file [Sampler_if.h](#).

6.103.2 Constructor & Destructor Documentation

6.103.2.1 ~RNG_Parameters()

```
virtual Sampler_if::RNG_Parameters::~RNG_Parameters ( ) [virtual], [default]
```

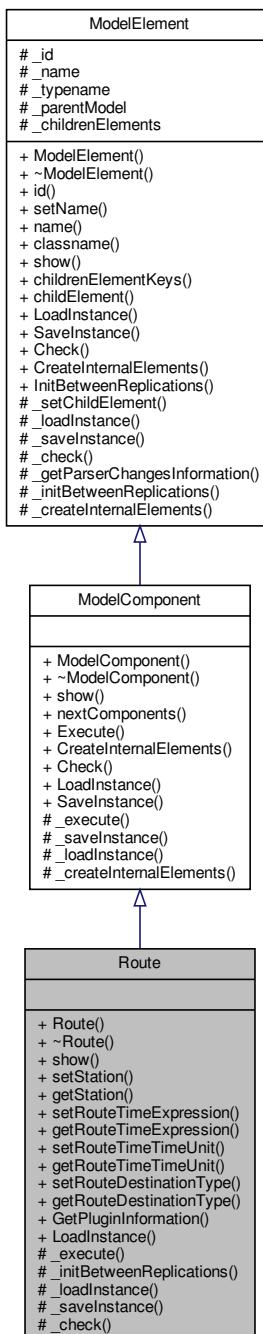
The documentation for this struct was generated from the following file:

- [Sampler_if.h](#)

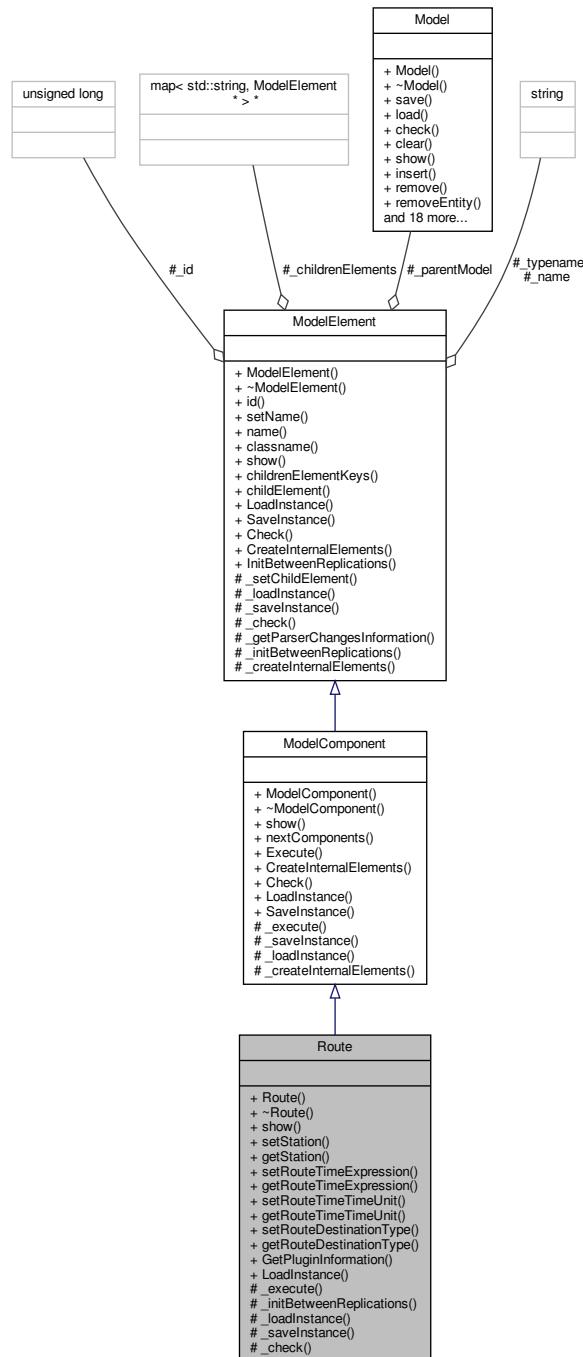
6.104 Route Class Reference

```
#include <Route.h>
```

Inheritance diagram for Route:



Collaboration diagram for Route:



Public Types

- enum **DestinationType** : int { **DestinationType::Station** = 0, **DestinationType::BySequence** = 1 }

Public Member Functions

- Route** (**Model** *model, **std::string** name="")

- virtual ~Route ()=default
- virtual std::string show ()
- void setStation (Station *_station)
- Station * getStation () const
- void setRouteTimeExpression (std::string _routeTimeExpression)
- std::string getRouteTimeExpression () const
- void setRouteTimeTimeUnit (Util::TimeUnit _routeTimeTimeUnit)
- Util::TimeUnit getRouteTimeTimeUnit () const
- void setRouteDestinationType (DestinationType _routeDestinationType)
- Route::DestinationType getRouteDestinationType () const

Static Public Member Functions

- static PluginInformation * GetPluginInformation ()
- static ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual void _execute (Entity *entity)
- virtual void _initBetweenReplications ()
- virtual bool _loadInstance (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * _saveInstance ()
- virtual bool _check (std::string *errorMessage)

Additional Inherited Members

6.104.1 Detailed Description

Route module DESCRIPTION The **Route** module transfers an entity to a specified station or the next station in the station visitation sequence defined for the entity. A delay time to transfer to the next station may be defined. When an entity enters the **Route** module, its **Station** attribute (Entity.Station) is set to the destination station. The entity is then sent to the destination station, using the route time specified. If the station destination is entered as By **Sequence**, the next station is determined by the entity's sequence and step within the set (defined by special-purpose attributes Entity.Sequence and Entity.Jobstep, respectively). TYPICAL USES Send a part to its next processing station based on its routing slip Send an account balance call to an account agent Send restaurant customers to a specific table PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. **Route** Time Travel time from the entity's current location to the destination station. Units Time units for route-time parameters. Destination Type Method for determining the entity destination location. Selection of By **Sequence** requires that the entity has been assigned a sequence name and that the sequence itself has been defined. **Station** Name Name of the individual destination station. **Attribute** Name Name of the attribute that stores the station name to which entities will route. Expression Expression that is evaluated to the station name where entities will route.

Definition at line 52 of file [Route.h](#).

6.104.2 Member Enumeration Documentation

6.104.2.1 DestinationType

```
enum Route::DestinationType : int [strong]
```

Enumerator

Station	
BySequence	

Definition at line 55 of file [Route.h](#).

6.104.3 Constructor & Destructor Documentation

6.104.3.1 Route()

```
Route::Route (
    Model * model,
    std::string name = "")
```

Definition at line 19 of file [Route.cpp](#).

Here is the caller graph for this function:



6.104.3.2 ~Route()

```
virtual Route::~Route ( ) [virtual], [default]
```

6.104.4 Member Function Documentation

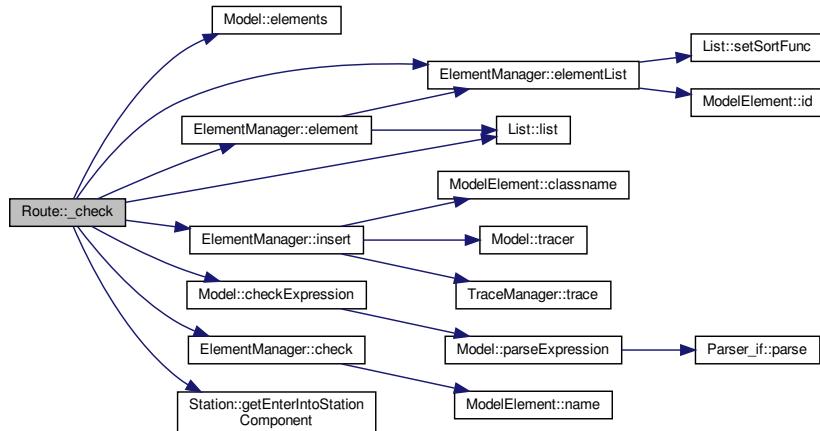
6.104.4.1 `_check()`

```
bool Route::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 117 of file [Route.cpp](#).

Here is the call graph for this function:



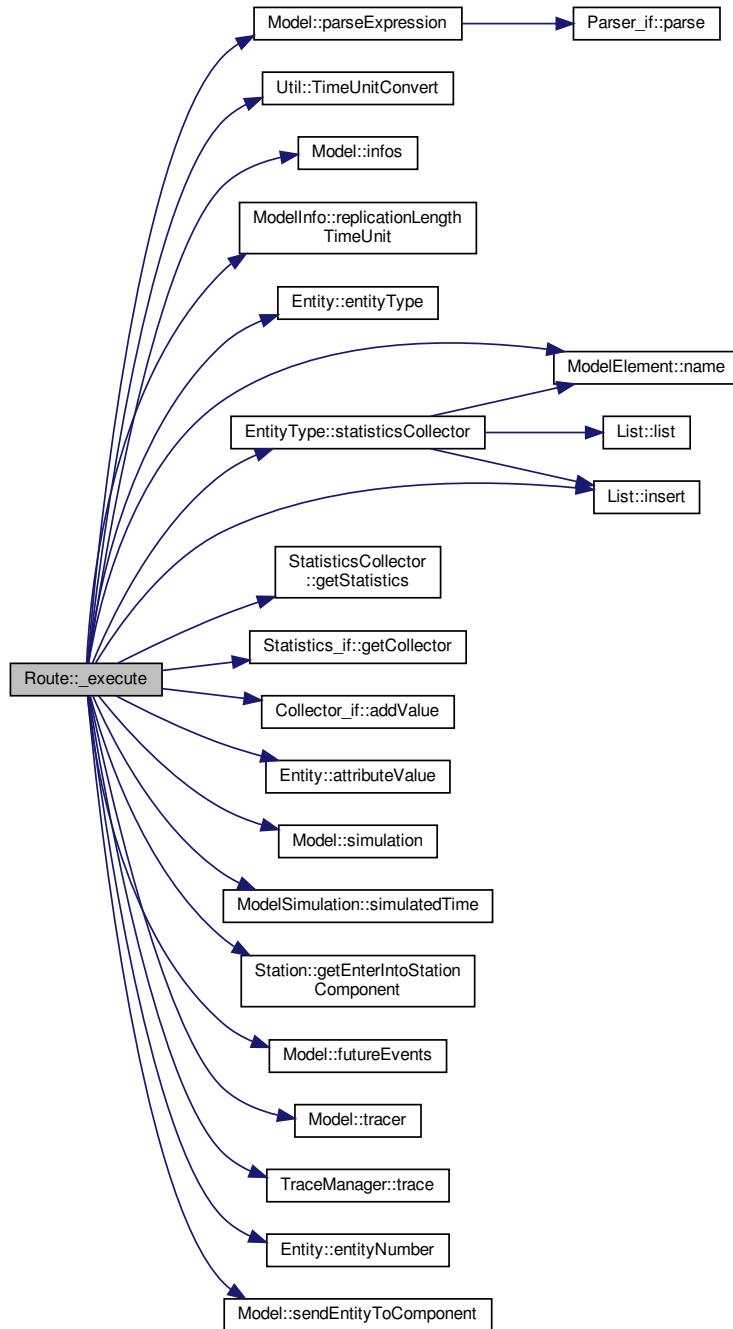
6.104.4.2 `_execute()`

```
void Route::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 74 of file [Route.cpp](#).

Here is the call graph for this function:



6.104.4.3 `_initBetweenReplications()`

```
void Route::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 104 of file [Route.cpp](#).

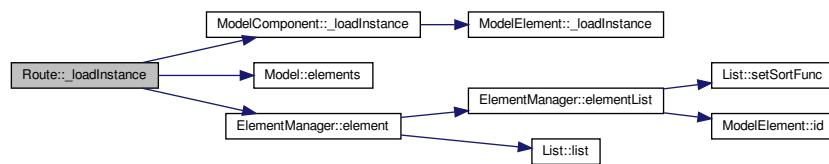
6.104.4.4 `_loadInstance()`

```
bool Route::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 91 of file [Route.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



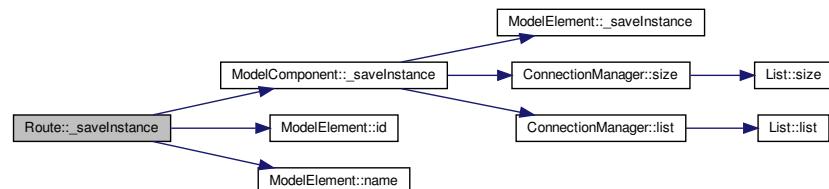
6.104.4.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Route::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 107 of file [Route.cpp](#).

Here is the call graph for this function:

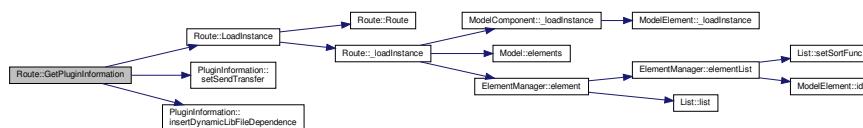


6.104.4.6 GetPluginInformation()

```
PluginInformation * Route::GetPluginInformation ( ) [static]
```

Definition at line 147 of file [Route.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.104.4.7 getRouteDestinationType()

```
Route::DestinationType Route::getRouteDestinationType ( ) const
```

Definition at line 70 of file [Route.cpp](#).

6.104.4.8 getRouteTimeExpression()

```
std::string Route::getRouteTimeExpression ( ) const
```

Definition at line 54 of file [Route.cpp](#).

6.104.4.9 getRouteTimeTimeUnit()

```
Util::TimeUnit Route::getRouteTimeTimeUnit ( ) const
```

Definition at line 62 of file [Route.cpp](#).

6.104.4.10 getStation()

```
Station * Route::getStation ( ) const
```

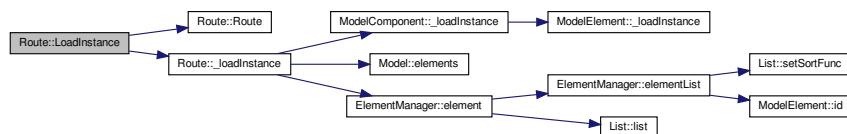
Definition at line 46 of file [Route.cpp](#).

6.104.4.11 LoadInstance()

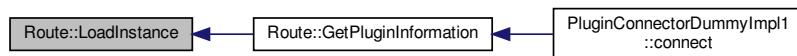
```
ModelComponent * Route::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 32 of file [Route.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.104.4.12 setRouteDestinationType()

```
void Route::setRouteDestinationType (
    DestinationType _routeDestinationType )
```

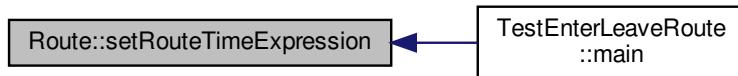
Definition at line 66 of file [Route.cpp](#).

6.104.4.13 setRouteTimeExpression()

```
void Route::setRouteTimeExpression (
    std::string _routeTimeExpression )
```

Definition at line 50 of file [Route.cpp](#).

Here is the caller graph for this function:



6.104.4.14 setRouteTimeTimeUnit()

```
void Route::setRouteTimeTimeUnit (
    Util::TimeUnit _routeTimeTimeUnit )
```

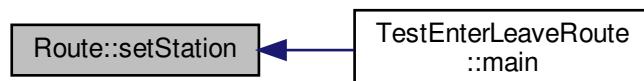
Definition at line 58 of file [Route.cpp](#).

6.104.4.15 setStation()

```
void Route::setStation (
    Station * _station )
```

Definition at line 42 of file [Route.cpp](#).

Here is the caller graph for this function:



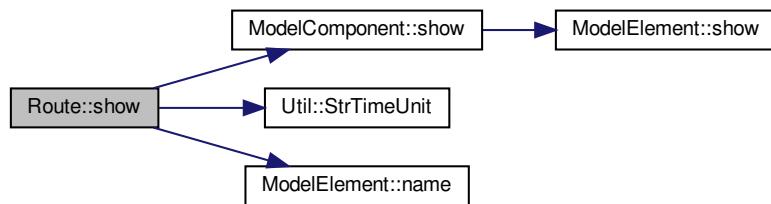
6.104.4.16 show()

```
std::string Route::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [23](#) of file [Route.cpp](#).

Here is the call graph for this function:



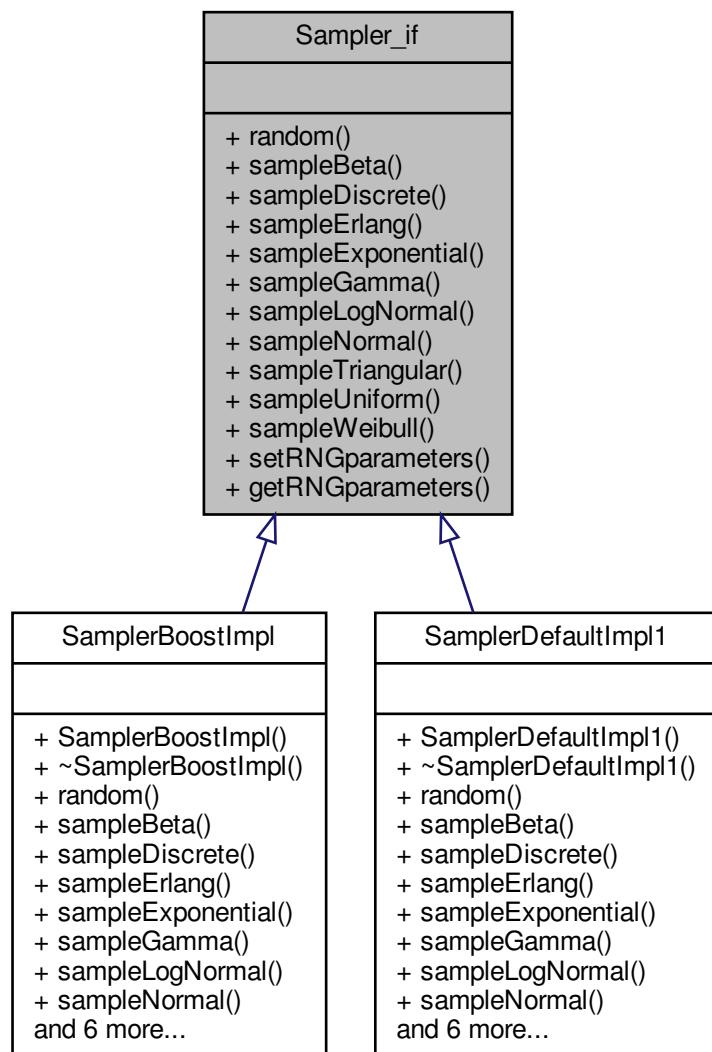
The documentation for this class was generated from the following files:

- [Route.h](#)
- [Route.cpp](#)

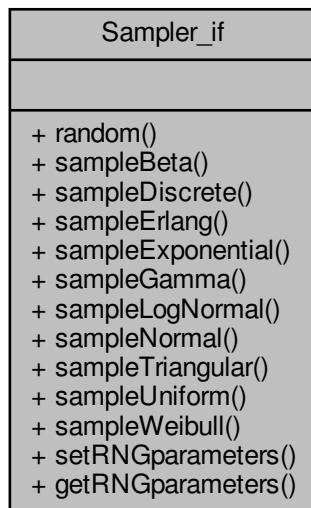
6.105 Sampler_if Class Reference

```
#include <Sampler_if.h>
```

Inheritance diagram for Sampler_if:



Collaboration diagram for Sampler_if:



Classes

- struct [RNG_Parameters](#)

Public Member Functions

- virtual double [random](#) ()=0
- virtual double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)=0
- virtual double [sampleDiscrete](#) (double acumProb, double value,...)=0
- virtual double [sampleErlang](#) (double mean, int M)=0
- virtual double [sampleExponential](#) (double mean)=0
- virtual double [sampleGamma](#) (double mean, double alpha)=0
- virtual double [sampleLogNormal](#) (double mean, double stddev)=0
- virtual double [sampleNormal](#) (double mean, double stddev)=0
- virtual double [sampleTriangular](#) (double min, double mode, double max)=0
- virtual double [sampleUniform](#) (double min, double max)=0
- virtual double [sampleWeibull](#) (double alpha, double scale)=0
- virtual void [setRNGparameters](#) ([RNG_Parameters](#) *param)=0
- virtual [RNG_Parameters](#) * [getRNGparameters](#) () const =0

6.105.1 Detailed Description

Interface that describes the methods to be implemented by classes that generate random values that follow a specific probability distribution.

Definition at line [20](#) of file [Sampler_if.h](#).

6.105.2 Member Function Documentation

6.105.2.1 getRNGparameters()

```
virtual RNG_Parameters* Sampler_if::getRNGparameters ( ) const [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:



6.105.2.2 random()

```
virtual double Sampler_if::random ( ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:

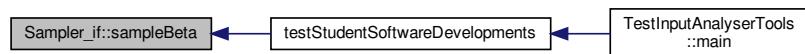


6.105.2.3 sampleBeta()

```
virtual double Sampler_if::sampleBeta (
    double alpha,
    double beta,
    double infLimit,
    double supLimit ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:



6.105.2.4 sampleDiscrete()

```
virtual double Sampler_if::sampleDiscrete (
    double acumProb,
    double value,
    ... ) [pure virtual]
```

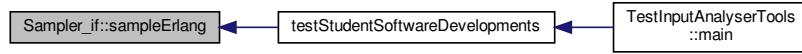
Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

6.105.2.5 sampleErlang()

```
virtual double Sampler_if::sampleErlang (
    double mean,
    int M ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:



6.105.2.6 sampleExponential()

```
virtual double Sampler_if::sampleExponential (
    double mean ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

6.105.2.7 sampleGamma()

```
virtual double Sampler_if::sampleGamma (
    double mean,
    double alpha ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:



6.105.2.8 sampleLogNormal()

```
virtual double Sampler_if::sampleLogNormal (
    double mean,
    double stddev ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

6.105.2.9 sampleNormal()

```
virtual double Sampler_if::sampleNormal (
    double mean,
    double stddev ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:

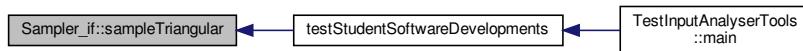


6.105.2.10 sampleTriangular()

```
virtual double Sampler_if::sampleTriangular (
    double min,
    double mode,
    double max ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:



6.105.2.11 sampleUniform()

```
virtual double Sampler_if::sampleUniform (
    double min,
    double max ) [pure virtual]
```

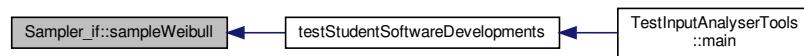
Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

6.105.2.12 sampleWeibull()

```
virtual double Sampler_if::sampleWeibull (
    double alpha,
    double scale ) [pure virtual]
```

Implemented in [SamplerDefaultImpl1](#), and [SamplerBoostImpl](#).

Here is the caller graph for this function:

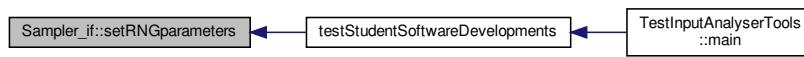


6.105.2.13 setRNGparameters()

```
virtual void Sampler_if::setRNGparameters (
    RNG_Parameters * param ) [pure virtual]
```

Implemented in [SamplerBoostImpl](#).

Here is the caller graph for this function:



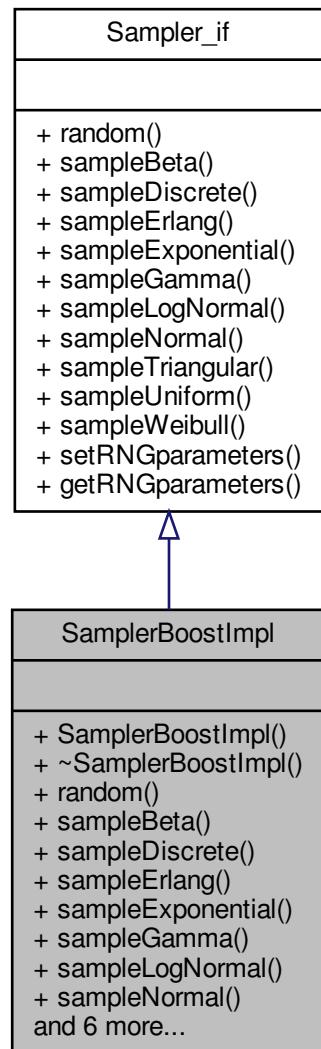
The documentation for this class was generated from the following file:

- [Sampler_if.h](#)

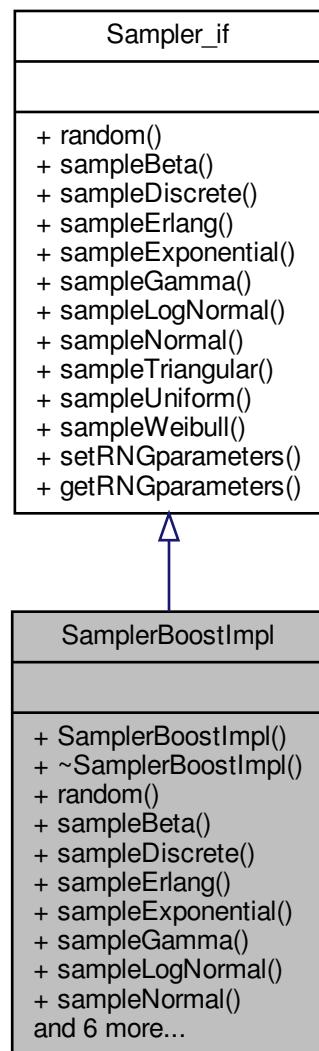
6.106 SamplerBoostImpl Class Reference

```
#include <SamplerBoostImpl.h>
```

Inheritance diagram for SamplerBoostImpl:



Collaboration diagram for SamplerBoostImpl:



Classes

- struct [BoostImplRNG_Parameters](#)

Public Member Functions

- [SamplerBoostImpl \(\)](#)
- virtual [~SamplerBoostImpl \(\)=default](#)
- virtual double [random \(\)](#)
- virtual double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)
- virtual double [sampleDiscrete](#) (double acumProb, double value,...)
- virtual double [sampleErlang](#) (double mean, int M)

- virtual double `sampleExponential` (double mean)
- virtual double `sampleGamma` (double mean, double alpha)
- virtual double `sampleLogNormal` (double mean, double stddev)
- virtual double `sampleNormal` (double mean, double stddev)
- virtual double `sampleTriangular` (double min, double mode, double max)
- virtual double `sampleUniform` (double min, double max)
- virtual double `sampleWeibull` (double alpha, double scale)
- void `reset` ()
reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.
- virtual void `setRNGparameters` (`Sampler_if::RNG_Parameters` *param)
- virtual `RNG_Parameters` * `getRNGparameters` () const

6.106.1 Detailed Description

Definition at line 20 of file `SamplerBoostImpl.h`.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 `SamplerBoostImpl()`

`SamplerBoostImpl::SamplerBoostImpl ()`

Definition at line 18 of file `SamplerBoostImpl.cpp`.

6.106.2.2 `~SamplerBoostImpl()`

`virtual SamplerBoostImpl::~SamplerBoostImpl () [virtual], [default]`

6.106.3 Member Function Documentation

6.106.3.1 `getRNGparameters()`

`Sampler_if::RNG_Parameters * SamplerBoostImpl::getRNGparameters () const [virtual]`

Implements `Sampler_if`.

Definition at line 71 of file `SamplerBoostImpl.cpp`.

6.106.3.2 random()

```
double SamplerBoostImpl::random ( ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 21 of file [SamplerBoostImpl.cpp](#).

6.106.3.3 reset()

```
void SamplerBoostImpl::reset ( )
```

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

6.106.3.4 sampleBeta()

```
double SamplerBoostImpl::sampleBeta (
    double alpha,
    double beta,
    double infLimit,
    double supLimit ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 26 of file [SamplerBoostImpl.cpp](#).

6.106.3.5 sampleDiscrete()

```
double SamplerBoostImpl::sampleDiscrete (
    double acumProb,
    double value,
    ... ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 30 of file [SamplerBoostImpl.cpp](#).

6.106.3.6 sampleErlang()

```
double SamplerBoostImpl::sampleErlang (
    double mean,
    int M ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 34 of file [SamplerBoostImpl.cpp](#).

6.106.3.7 sampleExponential()

```
double SamplerBoostImpl::sampleExponential (
    double mean ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 38 of file [SamplerBoostImpl.cpp](#).

6.106.3.8 sampleGamma()

```
double SamplerBoostImpl::sampleGamma (
    double mean,
    double alpha ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 42 of file [SamplerBoostImpl.cpp](#).

6.106.3.9 sampleLogNormal()

```
double SamplerBoostImpl::sampleLogNormal (
    double mean,
    double stddev ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 46 of file [SamplerBoostImpl.cpp](#).

6.106.3.10 sampleNormal()

```
double SamplerBoostImpl::sampleNormal (
    double mean,
    double stddev ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 50 of file [SamplerBoostImpl.cpp](#).

6.106.3.11 sampleTriangular()

```
double SamplerBoostImpl::sampleTriangular (
    double min,
    double mode,
    double max ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 54 of file [SamplerBoostImpl.cpp](#).

6.106.3.12 sampleUniform()

```
double SamplerBoostImpl::sampleUniform (
    double min,
    double max ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 58 of file [SamplerBoostImpl.cpp](#).

6.106.3.13 sampleWeibull()

```
double SamplerBoostImpl::sampleWeibull (
    double alpha,
    double scale ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 63 of file [SamplerBoostImpl.cpp](#).

6.106.3.14 setRNGparameters()

```
void SamplerBoostImpl::setRNGparameters (
    Sampler_if::RNG_Parameters * param ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 67 of file [SamplerBoostImpl.cpp](#).

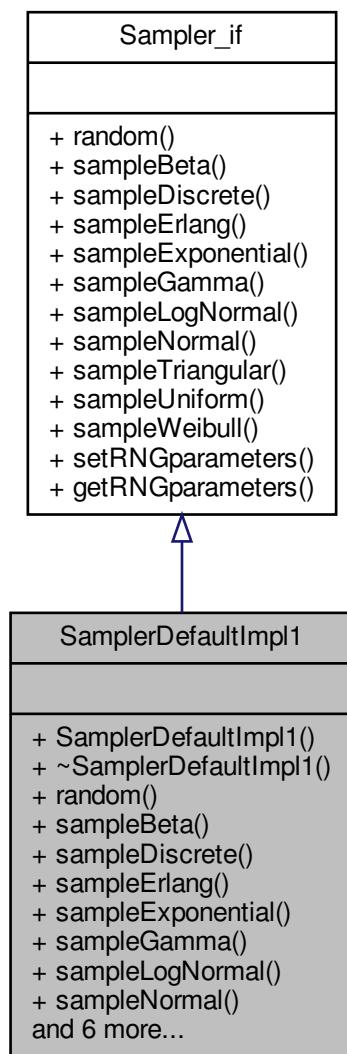
The documentation for this class was generated from the following files:

- [SamplerBoostImpl.h](#)
- [SamplerBoostImpl.cpp](#)

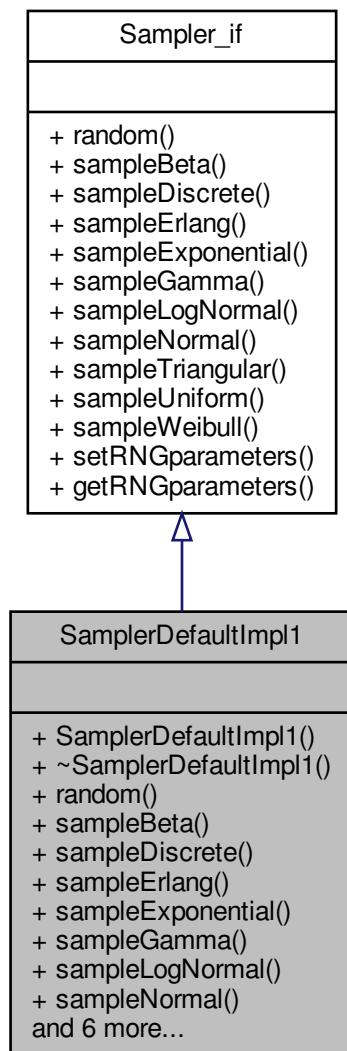
6.107 SamplerDefaultImpl1 Class Reference

```
#include <SamplerDefaultImpl1.h>
```

Inheritance diagram for SamplerDefaultImpl1:



Collaboration diagram for SamplerDefaultImpl1:



Classes

- struct [DefaultImpl1RNG_Parameters](#)

Public Member Functions

- [SamplerDefaultImpl1 \(\)](#)
- virtual [~SamplerDefaultImpl1 \(\)=default](#)
- virtual double [random \(\)](#)
- virtual double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)
- virtual double [sampleDiscrete](#) (double acumProb, double value,...)
- virtual double [sampleErlang](#) (double mean, int M)

- virtual double `sampleExponential` (double mean)
- virtual double `sampleGamma` (double mean, double alpha)
- virtual double `sampleLogNormal` (double mean, double stddev)
- virtual double `sampleNormal` (double mean, double stddev)
- virtual double `sampleTriangular` (double min, double mode, double max)
- virtual double `sampleUniform` (double min, double max)
- virtual double `sampleWeibull` (double alpha, double scale)
- void `reset` ()

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.
- virtual void `setRNGParameters` (RNG_Parameters *param)
- virtual RNG_Parameters * `getRNGparameters` () const

6.107.1 Detailed Description

Definition at line 19 of file [SamplerDefaultImpl1.h](#).

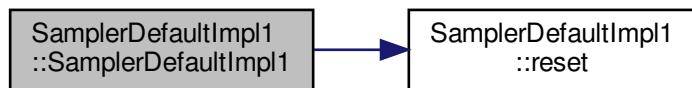
6.107.2 Constructor & Destructor Documentation

6.107.2.1 SamplerDefaultImpl1()

`SamplerDefaultImpl1::SamplerDefaultImpl1 ()`

Definition at line 21 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.107.2.2 ~SamplerDefaultImpl1()

`virtual SamplerDefaultImpl1::~SamplerDefaultImpl1 () [virtual], [default]`

6.107.3 Member Function Documentation

6.107.3.1 getRNGparameters()

```
Sampler_if::RNG_Parameters * SamplerDefaultImpl1::getRNGparameters ( ) const [virtual]
```

Implements [Sampler_if](#).

Definition at line 157 of file [SamplerDefaultImpl1.cpp](#).

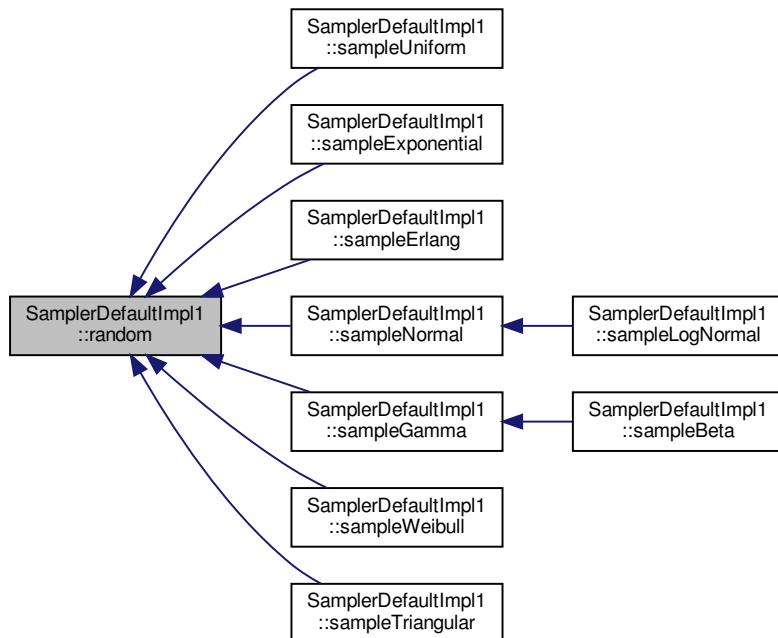
6.107.3.2 random()

```
double SamplerDefaultImpl1::random ( ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 31 of file [SamplerDefaultImpl1.cpp](#).

Here is the caller graph for this function:



6.107.3.3 reset()

```
void SamplerDefaultImpl1::reset ( )
```

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

Definition at line 25 of file [SamplerDefaultImpl1.cpp](#).

Here is the caller graph for this function:



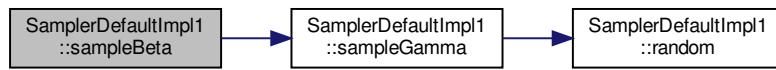
6.107.3.4 sampleBeta()

```
double SamplerDefaultImpl1::sampleBeta (
    double alpha,
    double beta,
    double infLimit,
    double supLimit ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 112 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.107.3.5 sampleDiscrete()

```
double SamplerDefaultImpl1::sampleDiscrete (
    double acumProb,
    double value,
    ... ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 149 of file [SamplerDefaultImpl1.cpp](#).

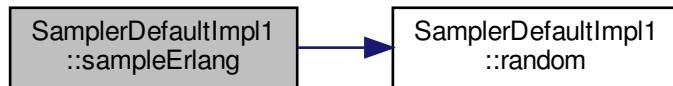
6.107.3.6 sampleErlang()

```
double SamplerDefaultImpl1::sampleErlang (
    double mean,
    int M )  [virtual]
```

Implements [Sampler_if](#).

Definition at line 46 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



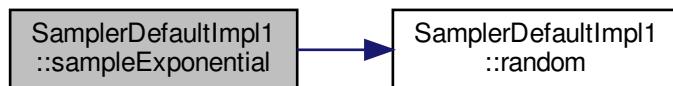
6.107.3.7 sampleExponential()

```
double SamplerDefaultImpl1::sampleExponential (
    double mean )  [virtual]
```

Implements [Sampler_if](#).

Definition at line 42 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



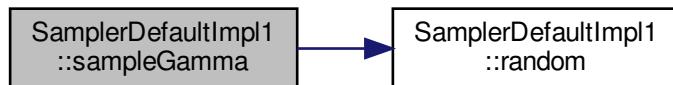
6.107.3.8 sampleGamma()

```
double SamplerDefaultImpl1::sampleGamma (
    double mean,
    double alpha ) [virtual]
```

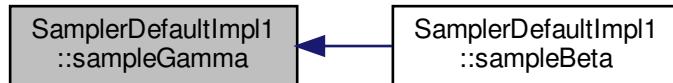
Implements [Sampler_if](#).

Definition at line 85 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



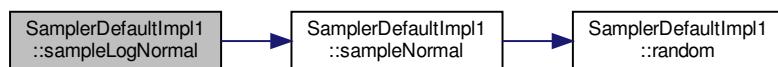
6.107.3.9 sampleLogNormal()

```
double SamplerDefaultImpl1::sampleLogNormal (
    double mean,
    double stddev ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 128 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



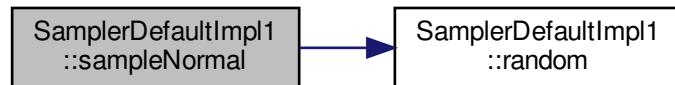
6.107.3.10 sampleNormal()

```
double SamplerDefaultImpl1::sampleNormal (
    double mean,
    double stddev ) [virtual]
```

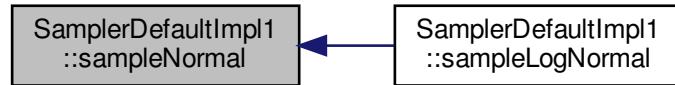
Implements [Sampler_if](#).

Definition at line 57 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



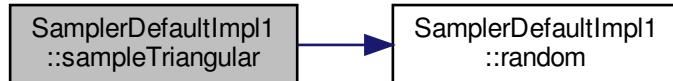
6.107.3.11 sampleTriangular()

```
double SamplerDefaultImpl1::sampleTriangular (
    double min,
    double mode,
    double max ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 136 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



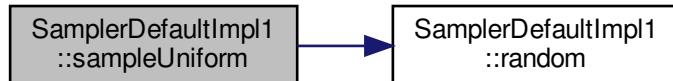
6.107.3.12 sampleUniform()

```
double SamplerDefaultImpl1::sampleUniform (
    double min,
    double max ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 38 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



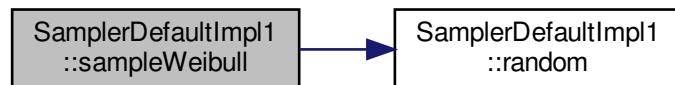
6.107.3.13 sampleWeibull()

```
double SamplerDefaultImpl1::sampleWeibull (
    double alpha,
    double scale ) [virtual]
```

Implements [Sampler_if](#).

Definition at line 123 of file [SamplerDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.107.3.14 setRNGparameters()

```
void SamplerDefaultImpl1::setRNGparameters (
    RNG_Parameters * param ) [virtual]
```

Definition at line 152 of file [SamplerDefaultImpl1.cpp](#).

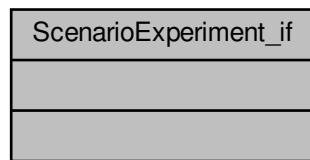
The documentation for this class was generated from the following files:

- [SamplerDefaultImpl1.h](#)
- [SamplerDefaultImpl1.cpp](#)

6.108 ScenarioExperiment_if Class Reference

```
#include <ScenarioExperiment_if.h>
```

Collaboration diagram for ScenarioExperiment_if:



6.108.1 Detailed Description

Definition at line 17 of file [ScenarioExperiment_if.h](#).

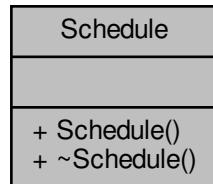
The documentation for this class was generated from the following file:

- [ScenarioExperiment_if.h](#)

6.109 Schedule Class Reference

```
#include <Schedule.h>
```

Collaboration diagram for Schedule:



Public Member Functions

- [Schedule \(Model *model\)](#)
- virtual [~Schedule \(\)=default](#)

6.109.1 Detailed Description

Schedule module DESCRIPTION This data module may be used in conjunction with the [Resource](#) module to define an operating schedule for a resource or with the [Create](#) module to define an arrival schedule. Additionally, a schedule may be used and referenced to factor time delays based on the simulation time. TYPICAL USES Work schedule for staff, including breaks Breakdown patterns for equipment Volume of customers arriving at a store Learning-curve factors for new workers PROMPTS [File](#) Read Time Specifies when to read the values from the file into the variable. If you select PreCheck, the values for the variable are read while the model is still in Edit mode (prior to the model being checked and compiled). If you select BeginSimulation, values are read when the model is compiled, prior to the first replication. If you select BeginReplication, values are read prior to each replication. Initial Values Lists the initial value or values of the variable. You can assign new values to the variable at different stages of the model by using the [Assign](#) module. Initial Value [Variable](#) value at the start of the simulation. Prompt Description Name The name of the schedule being defined. This name must be unique. Type Type of schedule being defined. This may be Capacity-related (for resource schedules), Arrival-related (for the [Create](#) module), or Other (miscellaneous time delays or factors) Time Units Time units used for the time-duration information. Scale Factor Method of scaling the schedule for increases or decreases in Arrival/Other values. The specified Value fields will be multiplied by the scale factor to determine the new values. Not available for Capacity-type schedules. Durations Lists the value and duration pairs for the schedule. Values can be capacity, arrival, or other type values, while the duration is specified in time units. [Schedule](#) pairs will repeat after all durations have been completed, unless the last duration is left blank (infinite). [Schedule](#) data can be entered graphically using the graphical schedule editor or manually using the Value/ Duration fields. Value Represents either the capacity of a resource (if Type is Capacity), arrival rate (if Type is Arrival), or some other value (if Type is Other). Examples of Other may be a factor that is used in a delay expression to scale a delay time during various parts of the day. Duration Time duration for which a specified Value will be valid.

Definition at line [67](#) of file [Schedule.h](#).

6.109.2 Constructor & Destructor Documentation

6.109.2.1 Schedule()

```
Schedule::Schedule (
    Model * model )
```

Definition at line 16 of file [Schedule.cpp](#).

6.109.2.2 ~Schedule()

```
virtual Schedule::~Schedule ( ) [virtual], [default]
```

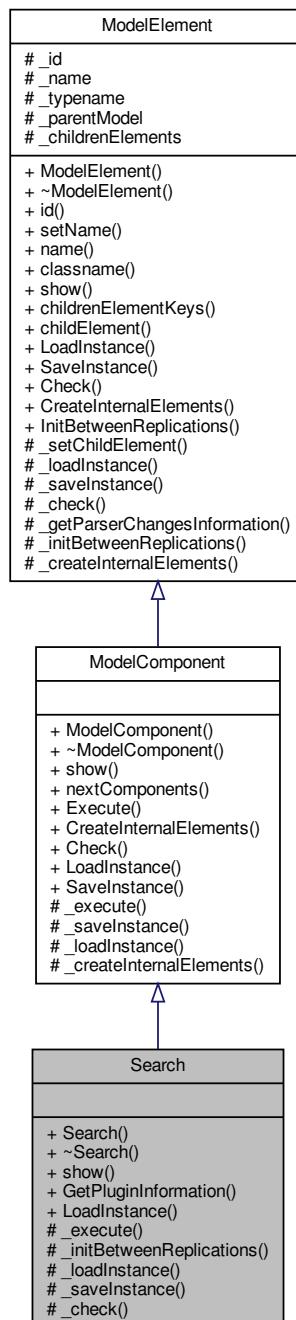
The documentation for this class was generated from the following files:

- [Schedule.h](#)
- [Schedule.cpp](#)

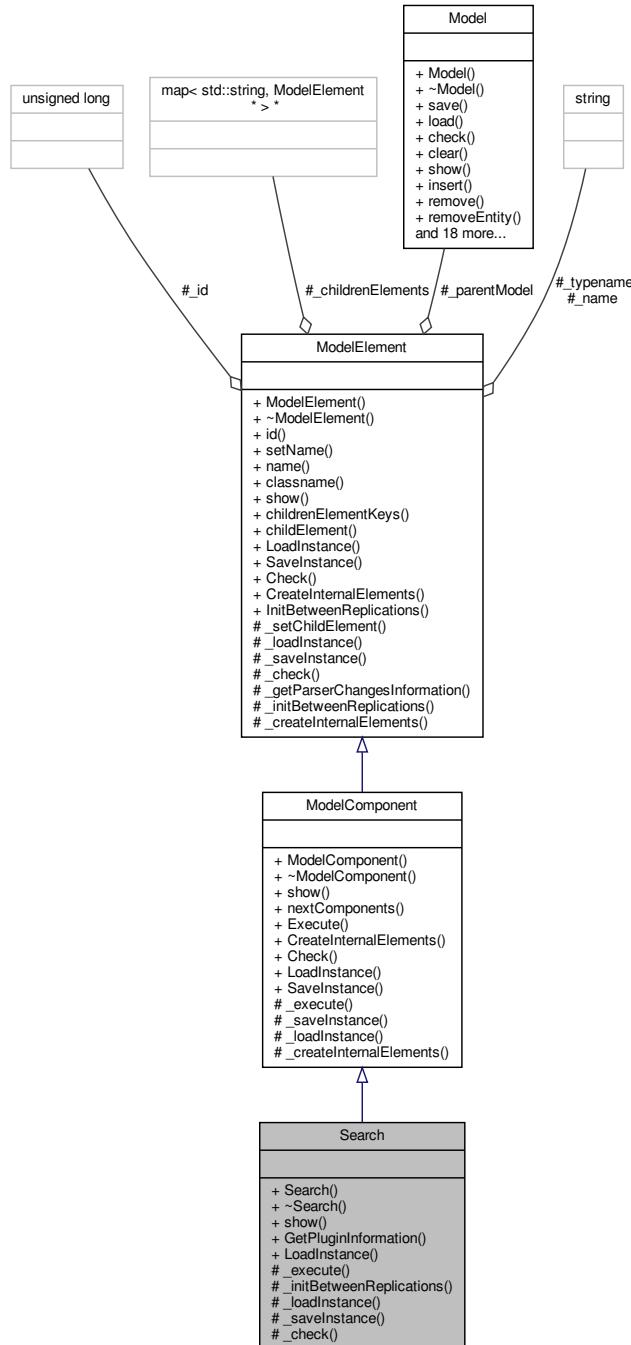
6.110 Search Class Reference

```
#include <Search.h>
```

Inheritance diagram for Search:



Collaboration diagram for Search:



Public Member Functions

- `Search (Model *model, std::string name="")`
- virtual `~Search ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.110.1 Detailed Description

Search module DESCRIPTION The **Search** module searches a queue, a group (batch), or an expression to find the entity rank (for entities in a queue or group) or the value of the global variable J that satisfies the specified search condition. When searching a queue or group, the value of the global system variable J is set to the rank of the first entity that satisfies **Search** Condition, or to 0 if **Search** Condition is not satisfied. When searching an expression, the global system variable J is set to the value of the first index value that satisfies the search condition or to zero if no value of J in the specified range satisfies the search condition. When an entity arrives at a **Search** module, the index J is set to the starting index and the search condition is then checked. If the search condition is satisfied, the search ends and the current value of J is retained. Otherwise, the value of J is increased or decreased and the condition is rechecked. This process repeats until the search condition is satisfied or the ending value is reached. If the condition is not met or there are no entities in the queue or group, J is set equal to 0. TYPICAL USES Looking for a particular order number in a queue Searching a group for a certain part type Determining which process to enter based on availability of resources (search an expression) Prompt Description Name Unique module identifier displayed on the module shape. Type Determination of what will be searched. **Search** options include entities in a queue, entities within a group (batch) or some expression(s). **Queue** Name Name of the queue that will be searched. Applies only when the Type is **Search** a **Queue**. Starting Value Starting rank in the queue or group or starting value for J in an expression. Ending Value Ending rank in the queue or group or ending value for J in an expression. **Search** Condition Condition containing the index J for searching expressions or containing an attribute name(s) for searching queues or batches.

Definition at line 55 of file [Search.h](#).

6.110.2 Constructor & Destructor Documentation

6.110.2.1 Search()

```
Search::Search (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Search.cpp](#).

Here is the caller graph for this function:



6.110.2.2 ~Search()

```
virtual Search::~Search ( ) [virtual], [default]
```

6.110.3 Member Function Documentation

6.110.3.1 _check()

```
bool Search::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Search.cpp](#).

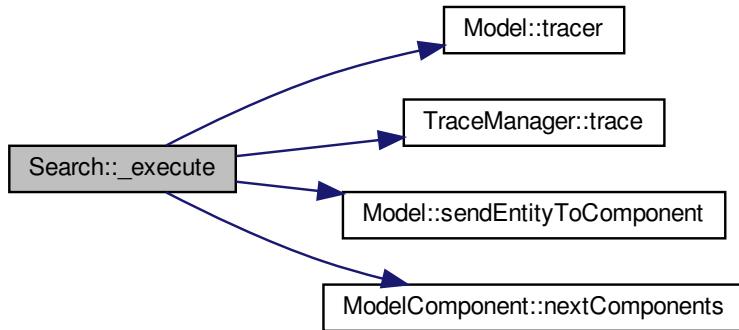
6.110.3.2 _execute()

```
void Search::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Search.cpp](#).

Here is the call graph for this function:



6.110.3.3 _initBetweenReplications()

```
void Search::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Search.cpp](#).

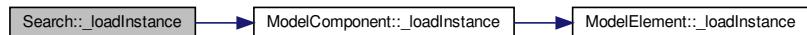
6.110.3.4 _loadInstance()

```
bool Search::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

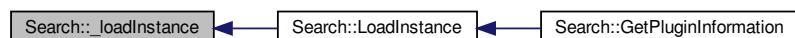
Reimplemented from [ModelComponent](#).

Definition at line 40 of file [Search.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



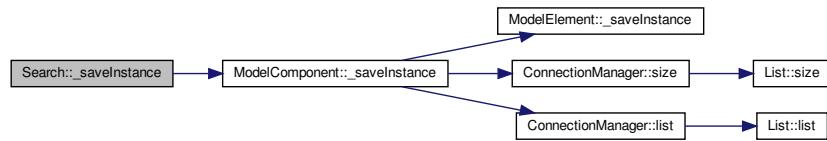
6.110.3.5 `_saveInstance()`

```
std::map< std::string, std::string > * Search::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Search.cpp](#).

Here is the call graph for this function:



6.110.3.6 `GetPluginInformation()`

```
PluginInformation * Search::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [Search.cpp](#).

Here is the call graph for this function:



6.110.3.7 `LoadInstance()`

```
ModelComponent * Search::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Search.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.110.3.8 show()

```
std::string Search::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [21](#) of file [Search.cpp](#).

Here is the call graph for this function:



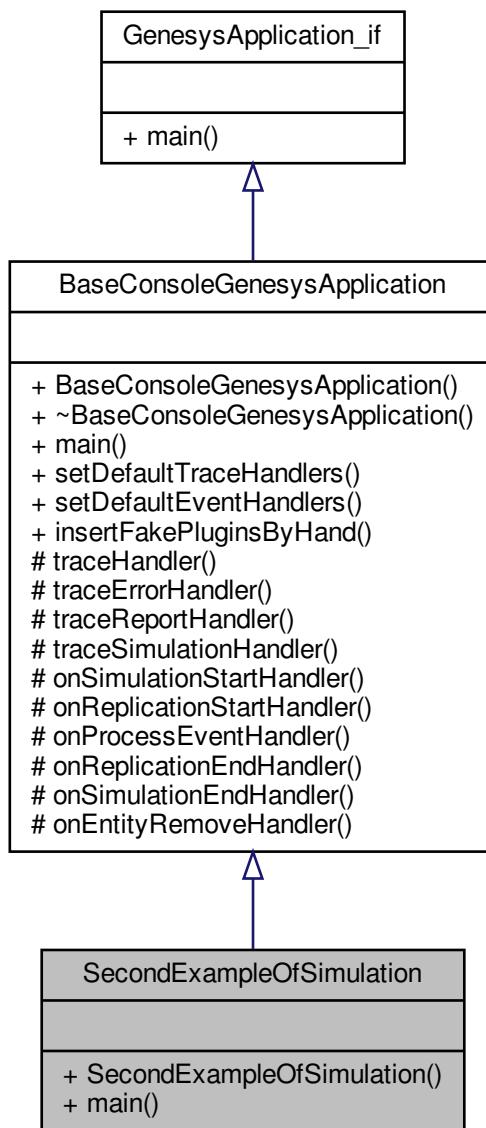
The documentation for this class was generated from the following files:

- [Search.h](#)
- [Search.cpp](#)

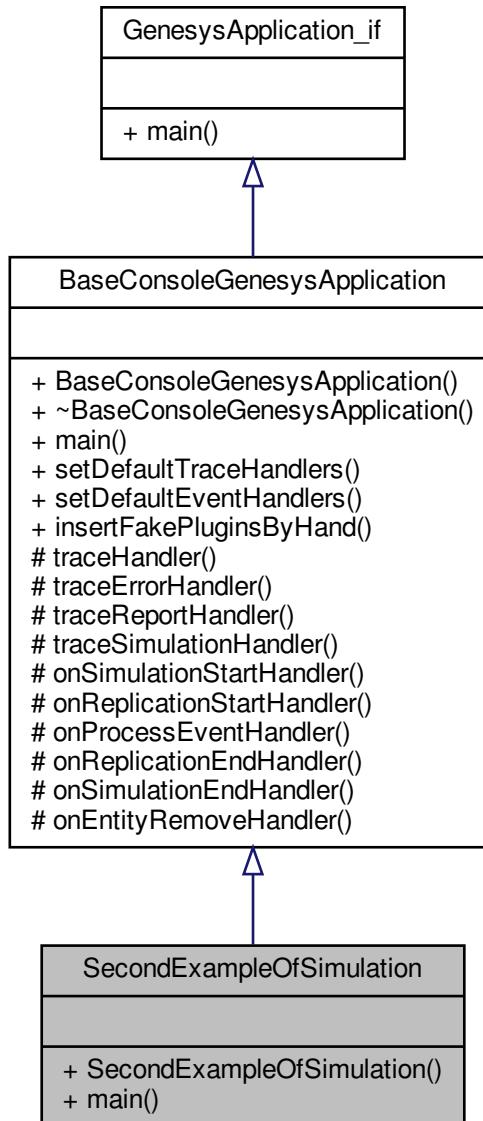
6.111 SecondExampleOfSimulation Class Reference

```
#include <SecondExampleOfSimulation.h>
```

Inheritance diagram for SecondExampleOfSimulation:



Collaboration diagram for SecondExampleOfSimulation:



Public Member Functions

- `SecondExampleOfSimulation ()`
- virtual int `main (int argc, char **argv)`

Additional Inherited Members

6.111.1 Detailed Description

Definition at line 19 of file [SecondExampleOfSimulation.h](#).

6.111.2 Constructor & Destructor Documentation

6.111.2.1 SecondExampleOfSimulation()

```
SecondExampleOfSimulation::SecondExampleOfSimulation ( )
```

Definition at line 30 of file [SecondExampleOfSimulation.cpp](#).

6.111.3 Member Function Documentation

6.111.3.1 main()

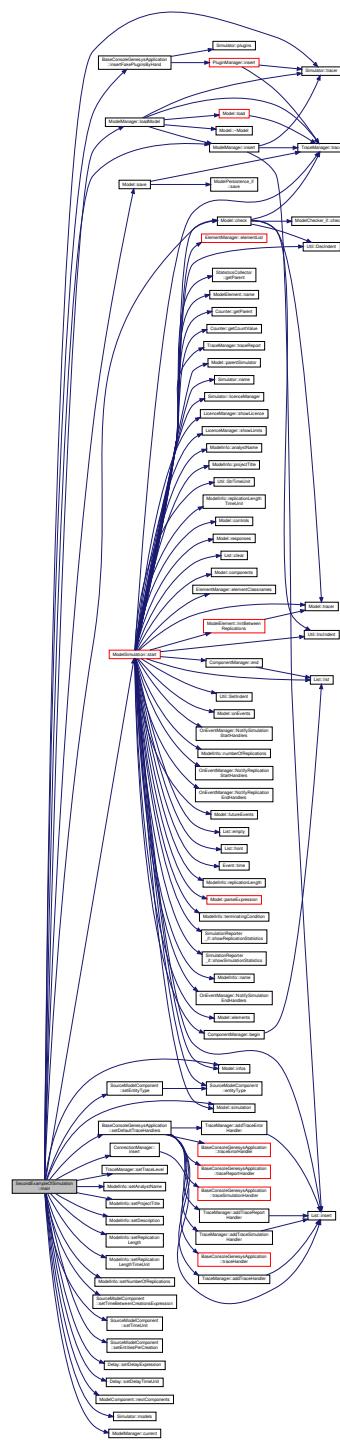
```
int SecondExampleOfSimulation::main (
    int argc,
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

Definition at line 37 of file [SecondExampleOfSimulation.cpp](#).

Here is the call graph for this function:



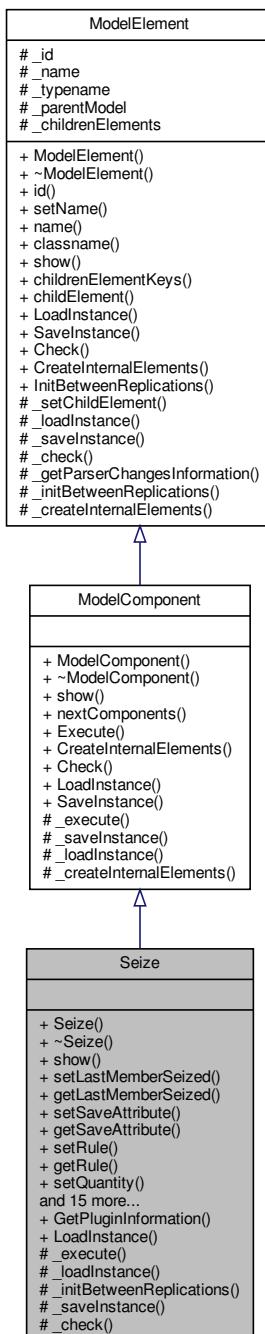
The documentation for this class was generated from the following files:

- SecondExampleOfSimulation.h
 - SecondExampleOfSimulation.cpp

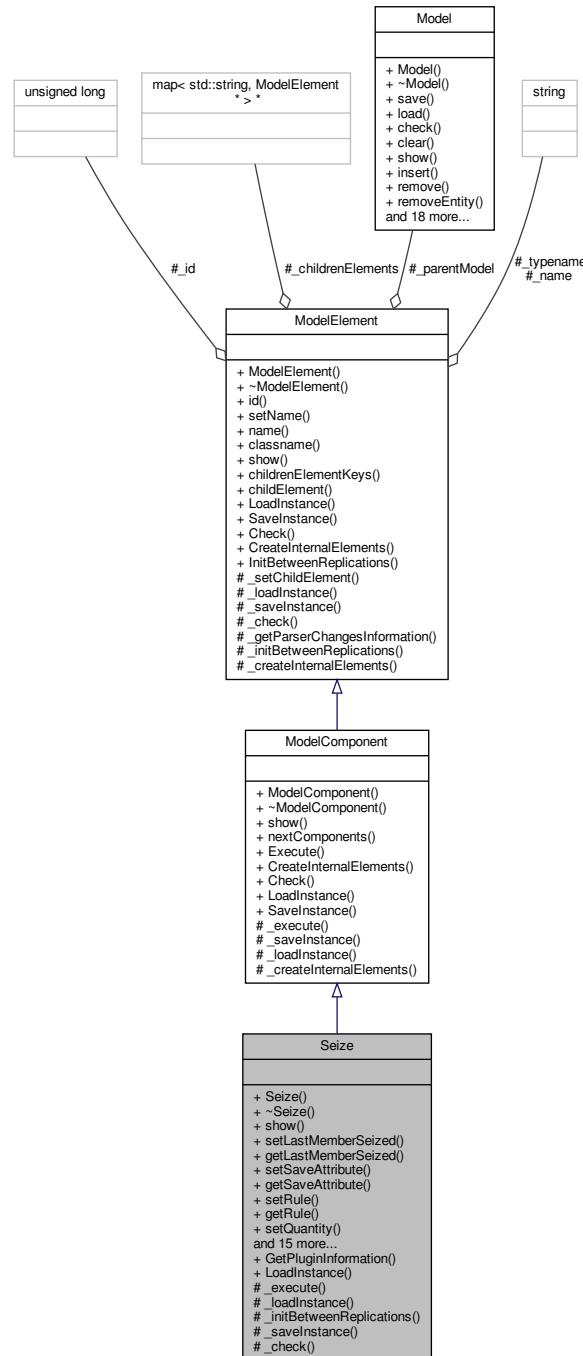
6.112 Seize Class Reference

```
#include <Seize.h>
```

Inheritance diagram for Seize:



Collaboration diagram for Seize:



Public Member Functions

- `Seize (Model *model, std::string name="")`
- `virtual ~Seize ()=default`
- `virtual std::string show ()`
- `void setLastMemberSeized (unsigned int _lastMemberSeized)`
- `unsigned int getLastMemberSeized () const`

- void `setSaveAttribute` (std::string _saveAttribute)
- std::string `getSaveAttribute` () const
- void `setRule` (Resource::ResourceRule _rule)
- Resource::ResourceRule `getRule` () const
- void `setQuantity` (std::string _quantity)
- std::string `getQuantity` () const
- void `setResourceType` (Resource::ResourceType _resourceType)
- Resource::ResourceType `getResourceType` () const
- void `setPriority` (unsigned short _priority)
- unsigned short `getPriority` () const
- void `setAllocationType` (unsigned int _allocationType)
- unsigned int `getAllocationType` () const
- void `setResourceName` (std::string _resourceName) throw ()
- std::string `getResourceName` () const
- void `setQueueName` (std::string queueName) throw ()
- std::string `getQueueName` () const
- void `setResource` (Resource *resource)
- Resource * `getResource` () const
- void `setQueue` (Queue *queue)
- Queue * `getQueue` () const

Static Public Member Functions

- static PluginInformation * `GetPluginInformation` ()
- static ModelComponent * `LoadInstance` (Model *model, std::map< std::string, std::string > *fields)

Protected Member Functions

- virtual void `_execute` (Entity *entity)
- virtual bool `_loadInstance` (std::map< std::string, std::string > *fields)
- virtual void `_initBetweenReplications` ()
- virtual std::map< std::string, std::string > * `_saveInstance` ()
- virtual bool `_check` (std::string *errorMessage)

Additional Inherited Members

6.112.1 Detailed Description

Seize module DESCRIPTION The **Seize** module allocates units of one or more resources to an entity. The **Seize** module may be used to seize units of a particular resource, a member of a resource set, or a resource as defined by an alternative method, such as an attribute or expression. When an entity enters this module, it waits in a queue (if specified) until all specified resources are available simultaneously. Allocation type for resource usage is also specified. TYPICAL USES Beginning a customer order (seize the operator) Starting a tax return (seize the accountant) Being admitted to hospital (seize the hospital room, nurse, doctor) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Allocation Determines to which category the resource usage cost will be allocated for an entity going through the **Seize** module. Priority Priority value of the entity waiting at this module for the resource(s) specified if one or more entities from other modules are waiting for the same resource(s). Type Type of resource for seizing, either specifying a particular resource, or selecting from a pool of resources (that is, a resource set). The name of the resource may also be specified as an attribute value or within an expression. Resource Name Name of the resource that will be seized. Set Name Name of the resource set from which a member will be seized. Attribute Name Name of the attribute that stores the resource name to be seized. Expression Expression that evaluates to a resource name to be seized. Quantity Number of resources of

a given name or from a given set that will be seized. For sets, this value specifies only the number of a selected resource that will be seized (based on the resource's capacity), not the number of members to be seized within the set. Selection Rule Method of selecting among available resources in a set. Cyclical will cycle through available members (for example, 1-2-3-1-2- 3). Random will randomly select a member. Preferred Order will always select the first available member (for example, 1, if available; then 2, if available; then 3). Specific Member requires an input attribute value to specify which member of the set (previously saved in the Save Attribute field). Largest Remaining Capacity and Smallest Number Busy are used for resources with multiple capacity. Save Attribute Attribute name used to store the index number into the set of the member that is chosen. This attribute can later be referenced with the Specific Member selection rule. Set Index Index value into the set that identifies the number into the set of the member requested. If an attribute name is used, the entity must have a value for the attribute before utilizing this option. Resource State State of the resource that will be assigned after the resource is seized. The resource state must be defined with the Resource module. Queue Type Determines the type of queue used to hold the entities while waiting to seize the resource(s). If Queue is selected, the queue name is specified. If Set is selected, the queue set and member in the set are specified. If Internal is selected, an internal queue is used to hold all waiting entities. Attribute and Expression are additional methods for defining the queue to be used. Queue Name This field is visible only if Queue Type is Queue, and it defines the symbol name of the queue. Set Name This field is visible only if Queue Type is Set, and it defines the queue set that contains the queue being referenced. Set Index This field is visible only if Queue Type is Set, and it defines the index into the queue set. Note that this is the index into the set and not the name of the queue in the set. For example, the only valid entries for a queue set containing three members is an expression that evaluates to 1, 2, or 3. Attribute This field is visible only if Queue Type is Attribute. The attribute entered in this field will be evaluated to indicate which queue is to be used. Expression This field is visible only if Queue Type is Expression. The expression entered in this field will be evaluated to indicate which queue is to be used.

Definition at line 125 of file [Seize.h](#).

6.112.2 Constructor & Destructor Documentation

6.112.2.1 Seize()

```
Seize::Seize (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Seize.cpp](#).

Here is the caller graph for this function:



6.112.2.2 ~Seize()

```
virtual Seize::~Seize ( ) [virtual], [default]
```

6.112.3 Member Function Documentation

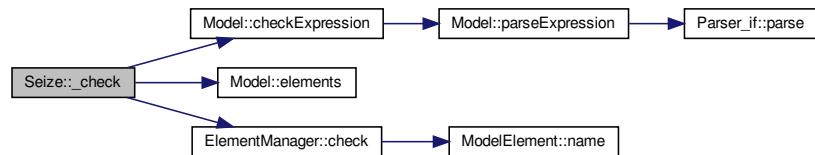
6.112.3.1 `_check()`

```
bool Seize::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 207 of file [Seize.cpp](#).

Here is the call graph for this function:

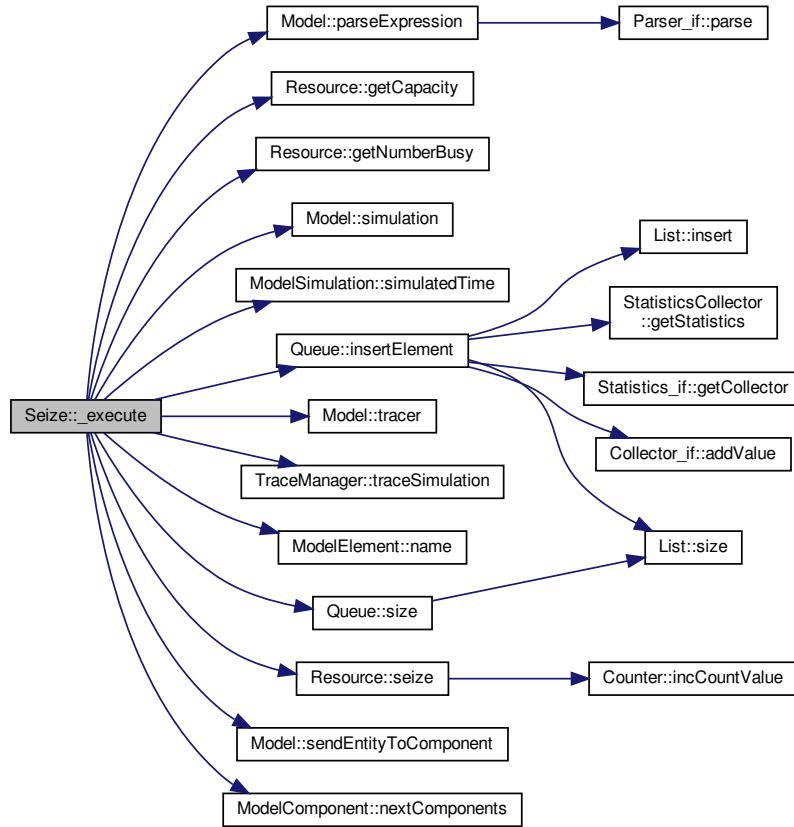
6.112.3.2 `_execute()`

```
void Seize::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 140 of file [Seize.cpp](#).

Here is the call graph for this function:



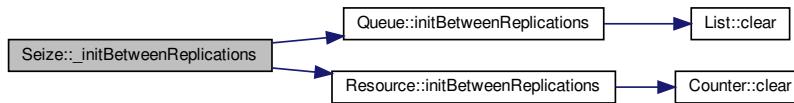
6.112.3.3 `_initBetweenReplications()`

```
void Seize::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 161 of file [Seize.cpp](#).

Here is the call graph for this function:



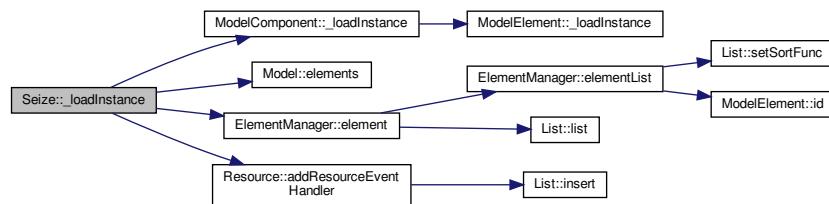
6.112.3.4 `_loadInstance()`

```
bool Seize::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 167 of file [Seize.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



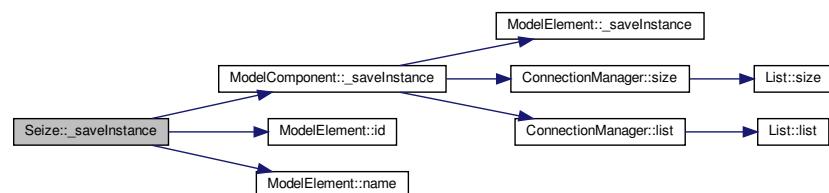
6.112.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Seize::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 192 of file [Seize.cpp](#).

Here is the call graph for this function:



6.112.3.6 getAllocationType()

```
unsigned int Seize::getAllocationType ( ) const
```

Definition at line 79 of file [Seize.cpp](#).

6.112.3.7 getLastMemberSeized()

```
unsigned int Seize::getLastMemberSeized ( ) const
```

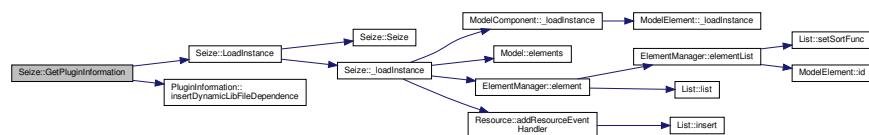
Definition at line 31 of file [Seize.cpp](#).

6.112.3.8 GetPluginInformation()

```
PluginInformation * Seize::GetPluginInformation ( ) [static]
```

Definition at line 216 of file [Seize.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.112.3.9 getPriority()

```
unsigned short Seize::getPriority ( ) const
```

Definition at line 71 of file [Seize.cpp](#).

6.112.3.10 getQuantity()

```
std::string Seize::getQuantity ( ) const
```

Definition at line 55 of file [Seize.cpp](#).

6.112.3.11 getQueue()

```
Queue * Seize::getQueue ( ) const
```

Definition at line 137 of file [Seize.cpp](#).

6.112.3.12 getQueueName()

```
std::string Seize::getQueueName ( ) const
```

Definition at line 120 of file [Seize.cpp](#).

Here is the call graph for this function:

**6.112.3.13 getResource()**

```
Resource * Seize::getResource ( ) const
```

Definition at line 129 of file [Seize.cpp](#).

6.112.3.14 getResourceName()

```
std::string Seize::getResourceName ( ) const
```

Definition at line 116 of file [Seize.cpp](#).

Here is the call graph for this function:



6.112.3.15 getResourceType()

```
Resource::ResourceType Seize::getResourceType ( ) const
```

Definition at line 63 of file [Seize.cpp](#).

6.112.3.16 getRule()

```
Resource::ResourceRule Seize::getRule ( ) const
```

Definition at line 47 of file [Seize.cpp](#).

6.112.3.17 getSaveAttribute()

```
std::string Seize::getSaveAttribute ( ) const
```

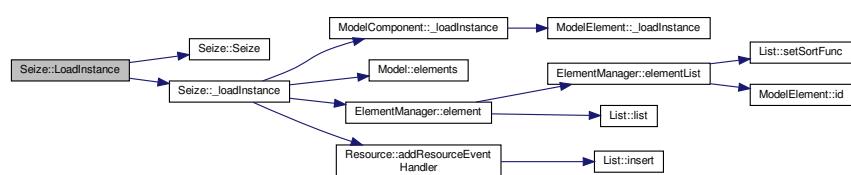
Definition at line 39 of file [Seize.cpp](#).

6.112.3.18 LoadInstance()

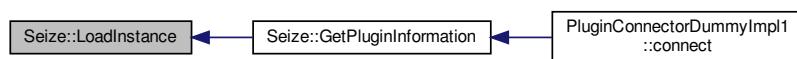
```
ModelComponent * Seize::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 223 of file [Seize.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.112.3.19 setAllocationType()

```
void Seize::setAllocationType (
    unsigned int _allocationType )
```

Definition at line 75 of file [Seize.cpp](#).

6.112.3.20 setLastMemberSeized()

```
void Seize::setLastMemberSeized (
    unsigned int _lastMemberSeized )
```

Definition at line 27 of file [Seize.cpp](#).

6.112.3.21 setPriority()

```
void Seize::setPriority (
    unsigned short _priority )
```

Definition at line 67 of file [Seize.cpp](#).

6.112.3.22 setQuantity()

```
void Seize::setQuantity (
    std::string _quantity )
```

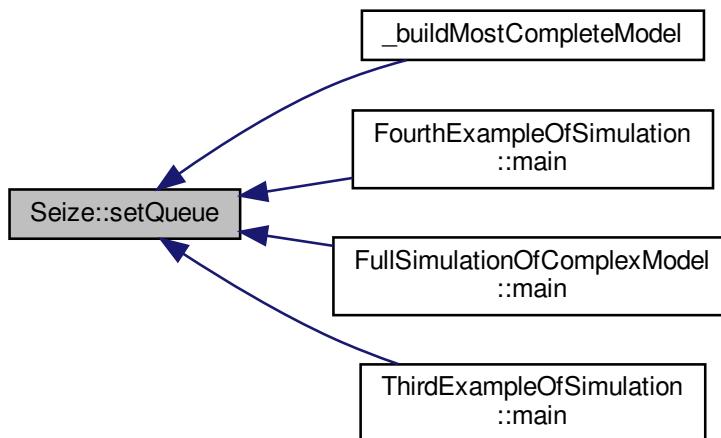
Definition at line 51 of file [Seize.cpp](#).

6.112.3.23 setQueue()

```
void Seize::setQueue (
    Queue * queue )
```

Definition at line 133 of file [Seize.cpp](#).

Here is the caller graph for this function:

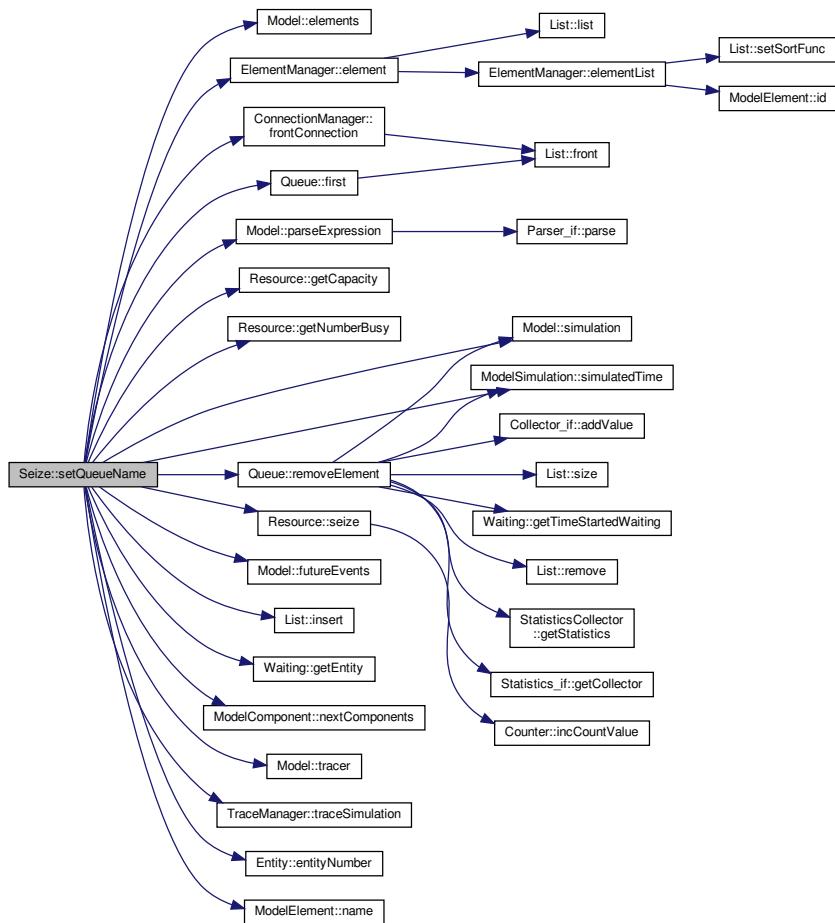


6.112.3.24 setQueueName()

```
void Seize::setQueueName (
    std::string queueName ) throw )
```

Definition at line 83 of file [Seize.cpp](#).

Here is the call graph for this function:

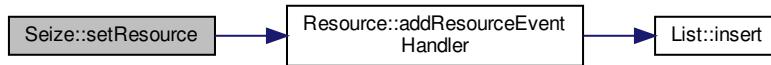


6.112.3.25 setResource()

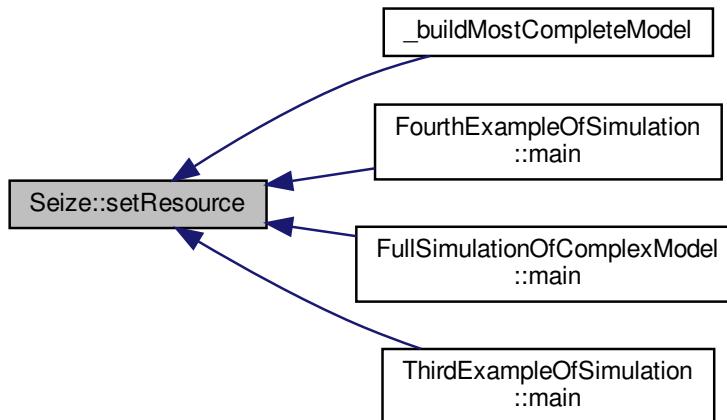
```
void Seize::setResource (
    Resource * resource )
```

Definition at line 124 of file [Seize.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

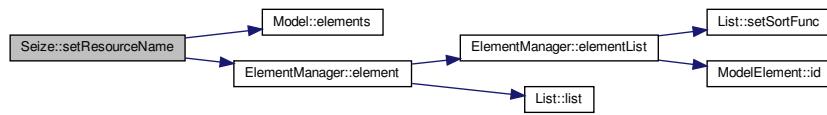


6.112.3.26 setResourceName()

```
void Seize::setResourceName (
    std::string _resourceName ) throw ()
```

Definition at line 107 of file [Seize.cpp](#).

Here is the call graph for this function:



6.112.3.27 set ResourceType()

```
void Seize::set ResourceType (
    Resource::ResourceType _resourceType )
```

Definition at line 59 of file [Seize.cpp](#).

6.112.3.28 set Rule()

```
void Seize::set Rule (
    Resource::ResourceRule _rule )
```

Definition at line 43 of file [Seize.cpp](#).

6.112.3.29 set SaveAttribute()

```
void Seize::set SaveAttribute (
    std::string _saveAttribute )
```

Definition at line 35 of file [Seize.cpp](#).

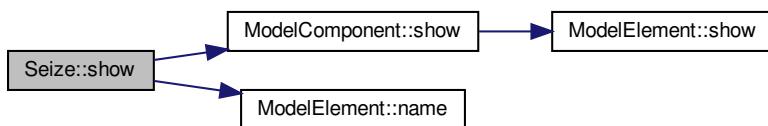
6.112.3.30 show()

```
std::string Seize::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 20 of file [Seize.cpp](#).

Here is the call graph for this function:



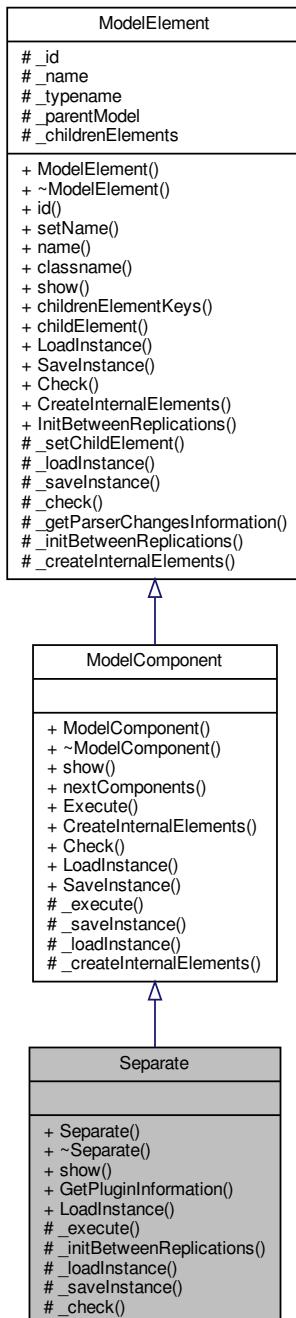
The documentation for this class was generated from the following files:

- [Seize.h](#)
- [Seize.cpp](#)

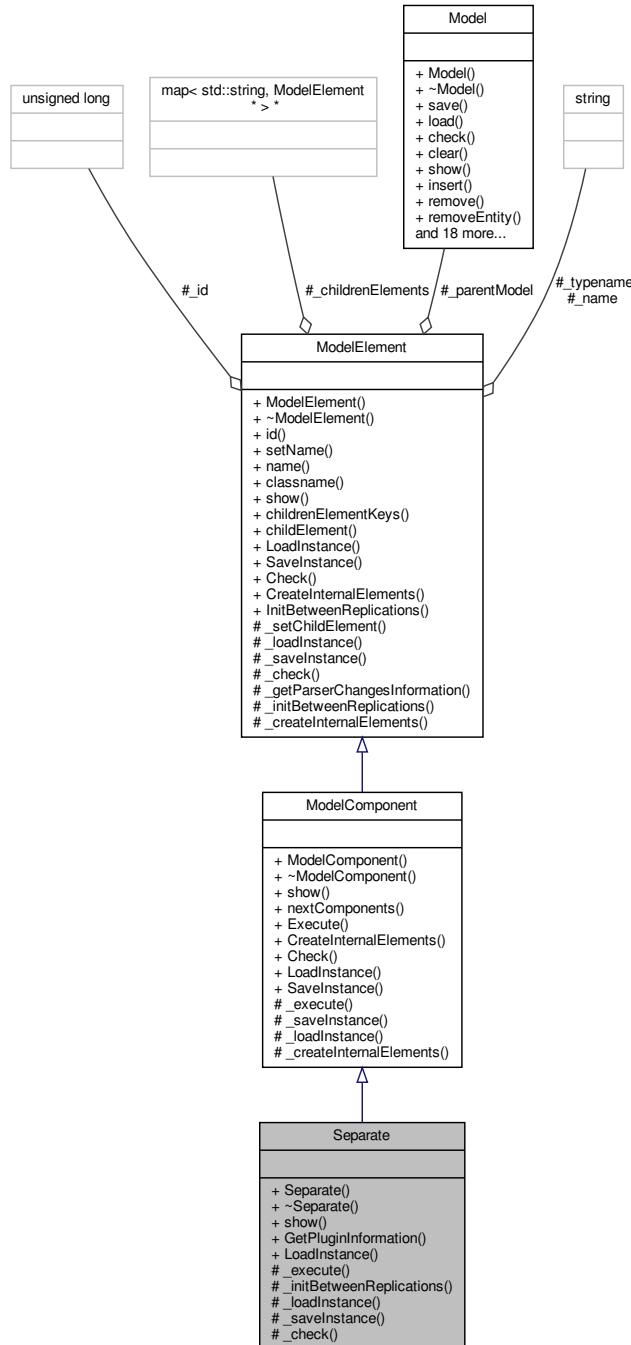
6.113 Separate Class Reference

```
#include <Separate.h>
```

Inheritance diagram for Separate:



Collaboration diagram for Separate:



Public Member Functions

- **Separate (Model *model, std::string name="")**
- virtual **~Separate ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.113.1 Detailed Description

Separate module DESCRIPTION This module can be used to either copy an incoming entity into multiple entities or to split a previously batched entity. Rules for allocating costs and times to the duplicate are also specified. Rules for attribute assignment to member entities are specified as well. When splitting existing batches, the temporary representative entity that was formed is disposed and the original entities that formed the group are recovered. The entities proceed sequentially from the module in the same order in which they originally were added to the batch. When duplicating entities, the specified number of copies is made and sent from the module. The original incoming entity also leaves the module. TYPICAL USES Send individual entities to represent boxes removed from a container Send an order both to fulfillment and billing for parallel processing **Separate** a previously batched set of documents PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of separating the incoming entity. Duplicate Original will simply take the original entity and make some number of identical duplicates. Split Existing **Batch** requires that the incoming entity be a temporarily batched entity using the **Batch** module. The original entities from the batch will be split. Percent Cost to Duplicates Allocation of costs and times of the incoming entity to the outgoing duplicates. This value is specified as a percentage of the original entity's costs and times (between 0-100). The percentage specified will be split evenly between the duplicates, while the original entity will retain any remaining cost/time percentage. Visible only when Type is Duplicate Original.

of Duplicates Number of outgoing entities that will leave the module, in

addition to the original incoming entity. Applies only when Type is Duplicate Original. Member Attributes Method of determining how to assign the representative entity attribute values to the original entities. These options relate to six of the special-purpose attributes (Entity.Type, Entity.Picture, Entity.Sequence, Entity.Station, Entity.Jobstep, and Entity.HoldCostRate) and all user-defined attributes. Applies only when Type is Split Existing **Batch**. Attribute Name Name of representative entity attribute(s) that are assigned to original entities of the group. Applies only when Member Attributes is Take Specific Representative Values.

Definition at line 65 of file [Separate.h](#).

6.113.2 Constructor & Destructor Documentation

6.113.2.1 Separate()

```
Separate::Separate (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Separate.cpp](#).

Here is the caller graph for this function:



6.113.2.2 ~Separate()

```
virtual Separate::~Separate () [virtual], [default]
```

6.113.3 Member Function Documentation

6.113.3.1 _check()

```
bool Separate::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 59 of file [Separate.cpp](#).

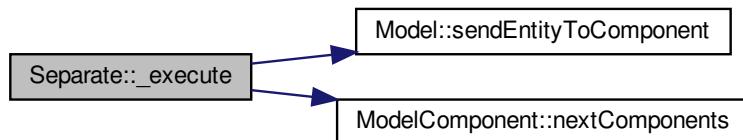
6.113.3.2 _execute()

```
void Separate::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Separate.cpp](#).

Here is the call graph for this function:



6.113.3.3 `_initBetweenReplications()`

```
void Separate::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [50](#) of file [Separate.cpp](#).

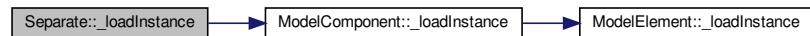
6.113.3.4 `_loadInstance()`

```
bool Separate::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

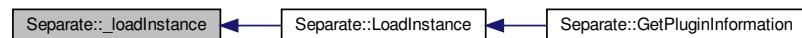
Reimplemented from [ModelComponent](#).

Definition at line [42](#) of file [Separate.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



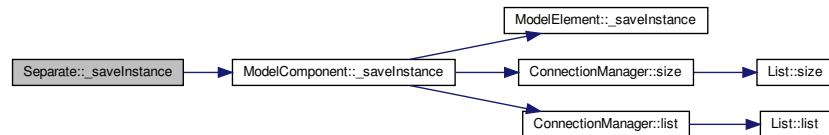
6.113.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Separate::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [53](#) of file [Separate.cpp](#).

Here is the call graph for this function:

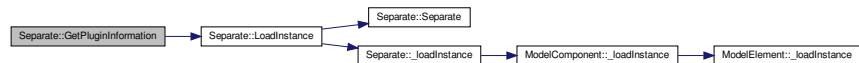


6.113.3.6 GetPluginInformation()

```
PluginInformation * Separate::GetPluginInformation ( ) [static]
```

Definition at line 65 of file [Separate.cpp](#).

Here is the call graph for this function:

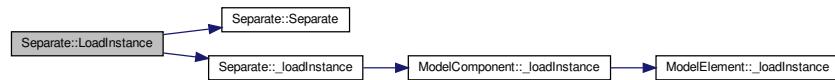


6.113.3.7 LoadInstance()

```
ModelComponent * Separate::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Separate.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



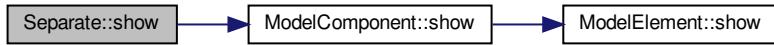
6.113.3.8 show()

```
std::string Separate::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Separate.cpp](#).

Here is the call graph for this function:



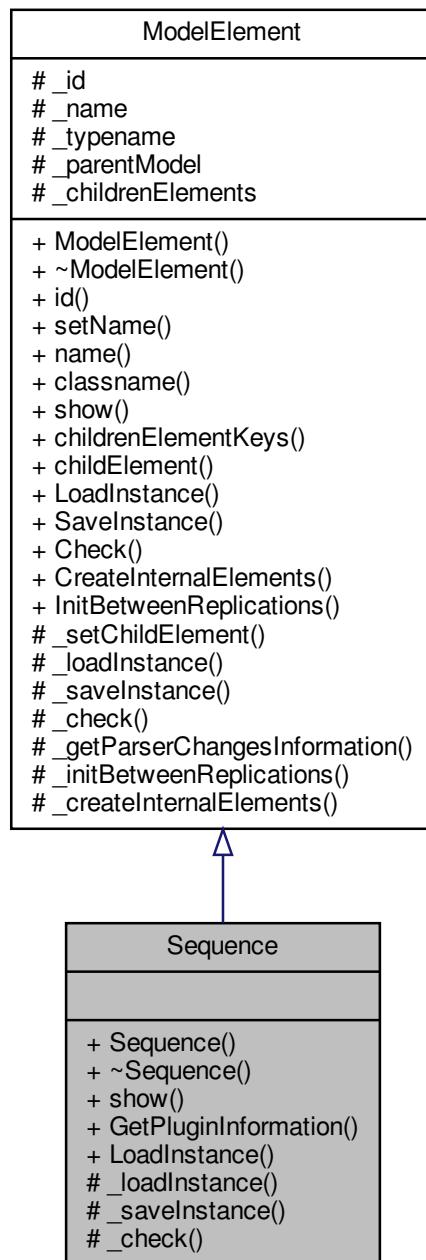
The documentation for this class was generated from the following files:

- [Separate.h](#)
- [Separate.cpp](#)

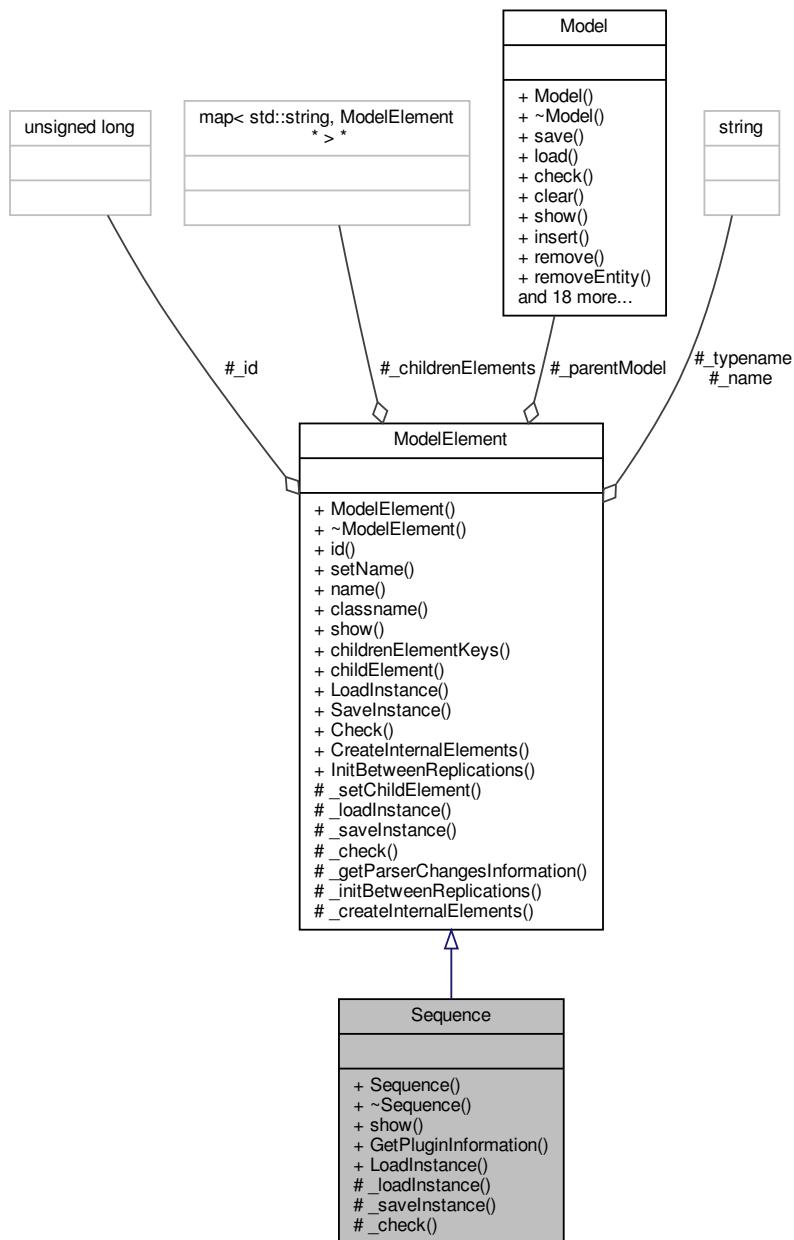
6.114 Sequence Class Reference

```
#include <Sequence.h>
```

Inheritance diagram for Sequence:



Collaboration diagram for Sequence:



Classes

- class [SequenceStep](#)

Public Member Functions

- **Sequence** (`Model *model, std::string name=""`)
- virtual **~Sequence** ()=default
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.114.1 Detailed Description

Sequence module DESCRIPTION The **Sequence** module is used to define a sequence for entity flow through the model. A sequence consists of an ordered list of stations that an entity will visit. For each station in the visitation sequence, attributes and variables may be assigned values. Each station in the visitation sequence is referred to as a step (or jobstep) in the sequence. Three special-purpose attributes are provided for all entities. The **Sequence** attribute (`Entity.Sequence`) defines the sequence that an entity is to follow; a value of 0 indicates that the entity is not following any sequence. In order for an entity to follow a sequence, its **Sequence** attribute must be assigned a value (for example, in the **Assign** module). The **Jobstep** attribute (`Entity.Jobstep`) stores the entity's current step number in the sequence. This value is updated automatically each time an entity is transferred. You typically do not need to assign explicitly a value to **Jobstep** in the model. The **PlannedStation** attribute (`Entity.PlannedStation`) stores the number of the station associated with the next jobstep in the sequence. This attribute is not user-assignable. It is automatically updated whenever `Entity.Sequence` or `Entity.JobStep` changes, or whenever the entity enters a station. **Jobstep** names must be globally unique. TYPICAL USES Define a routing path for part processing Define a sequence of steps patients must take upon arrival at an emergency room

Definition at line 47 of file [Sequence.h](#).

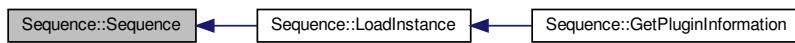
6.114.2 Constructor & Destructor Documentation

6.114.2.1 Sequence()

```
Sequence::Sequence (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Sequence.cpp](#).

Here is the caller graph for this function:



6.114.2.2 ~Sequence()

```
virtual Sequence::~Sequence ( ) [virtual], [default]
```

6.114.3 Member Function Documentation

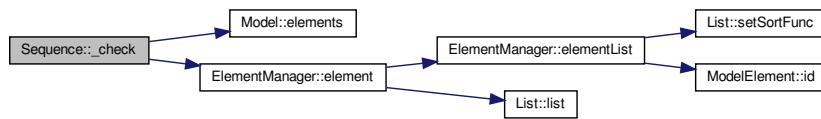
6.114.3.1 _check()

```
bool Sequence::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Sequence.cpp](#).

Here is the call graph for this function:



6.114.3.2 _loadInstance()

```
bool Sequence::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

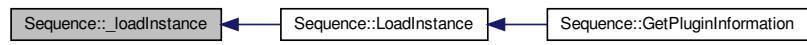
Reimplemented from [ModelElement](#).

Definition at line 43 of file [Sequence.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.114.3.3 `_saveInstance()`

```
std::map< std::string, std::string > * Sequence::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 53 of file [Sequence.cpp](#).

Here is the call graph for this function:

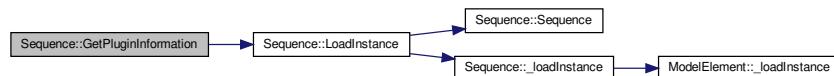


6.114.3.4 `GetPluginInformation()`

```
PluginInformation * Sequence::GetPluginInformation ( ) [static]
```

Definition at line 28 of file [Sequence.cpp](#).

Here is the call graph for this function:

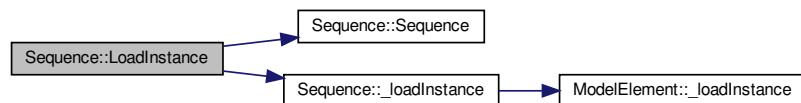


6.114.3.5 `LoadInstance()`

```
ModelElement * Sequence::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 33 of file [Sequence.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.114.3.6 show()

```
std::string Sequence::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [22](#) of file [Sequence.cpp](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [Sequence.h](#)
- [Sequence.cpp](#)

6.115 Sequence::SequenceStep Class Reference

```
#include <Sequence.h>
```

Collaboration diagram for Sequence::SequenceStep:



Public Attributes

- `Station * _station`
- `std::list< std::string > * _assignments`

6.115.1 Detailed Description

Definition at line 49 of file [Sequence.h](#).

6.115.2 Member Data Documentation

6.115.2.1 `_assignments`

```
std::list<std::string>* Sequence::SequenceStep::_assignments
```

Definition at line 52 of file [Sequence.h](#).

6.115.2.2 `_station`

```
Station* Sequence::SequenceStep::_station
```

Definition at line 51 of file [Sequence.h](#).

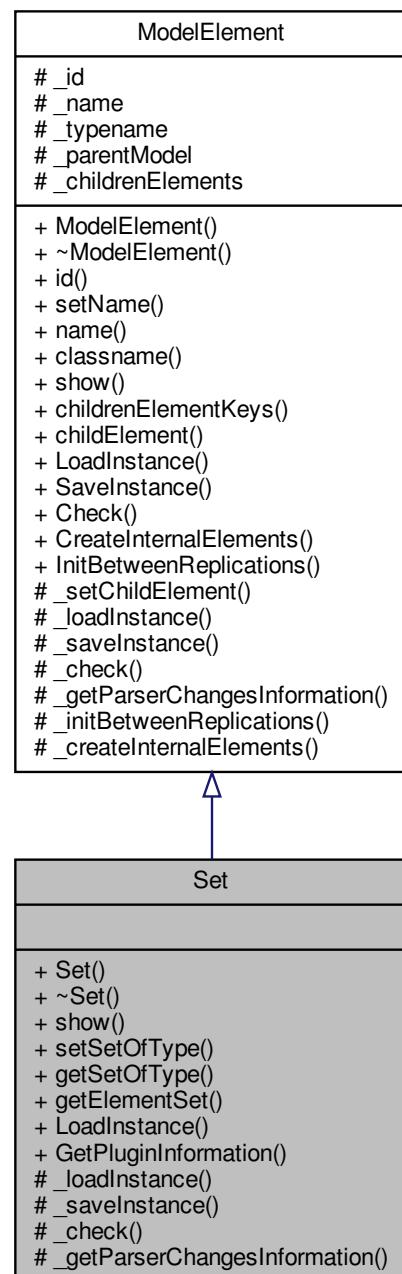
The documentation for this class was generated from the following file:

- [Sequence.h](#)

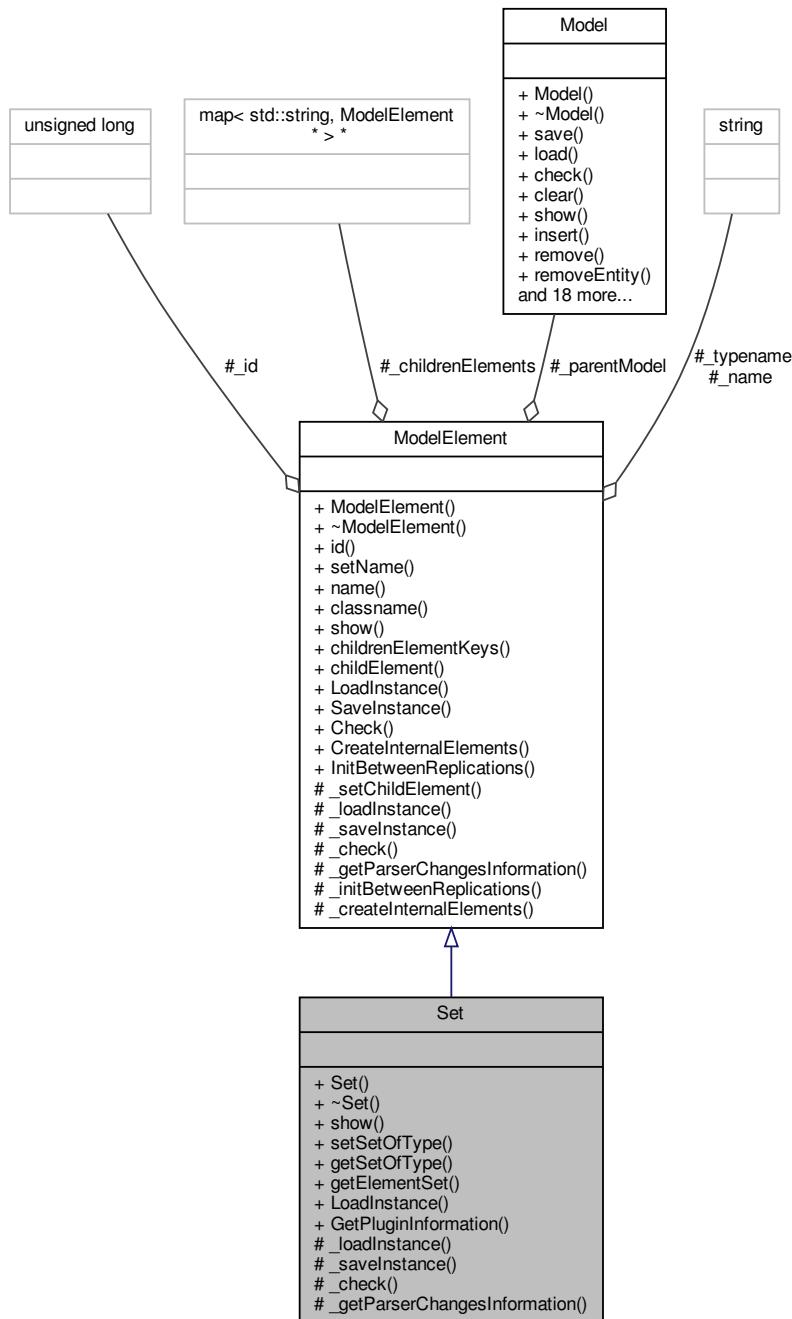
6.116 Set Class Reference

```
#include <Set.h>
```

Inheritance diagram for Set:



Collaboration diagram for Set:



Public Member Functions

- `Set (Model *model, std::string name="")`
- `virtual ~Set ()=default`
- `virtual std::string show ()`
- `void setSetOfType (std::string _setOfType)`
- `std::string getSetOfType () const`
- `List< ModelElement * > * getElementSet () const`

Static Public Member Functions

- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`
- static `PluginInformation * GetPluginInformation ()`

Protected Member Functions

- virtual `bool _loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual `bool _check (std::string *errorMessage)`
- virtual `ParserChangesInformation * _getParserChangesInformation ()`

Additional Inherited Members

6.116.1 Detailed Description

Set module DESCRIPTION This data module defines various types of sets, including resource, counter, tally, entity type, and entity picture. **Resource** sets can be used in the Process modules (and **Seize**, **Release**, **Enter**, and **Leave** of the Advanced Process and Advanced Transfer panels). **Counter** and **Tally** sets can be used in the **Record** module. **Queue** sets can be used with the **Seize**, **Hold**, **Access**, **Request**, **Leave**, and **Allocate** modules of the Advanced Process and Advanced Transfer panels. TYPICAL USES Machines that can perform the same operations in a manufacturing facility Supervisors, check-out clerks in a store Shipping clerks, receptionists in an office **Set** of pictures corresponding to a set of entity types PROMPTS Prompt Description Name The unique name of the set being defined. Type Type of set being defined. Members Repeat group that specifies the resource members with the set. The order of listing the members within the repeat group is important when using selection rules such as Preferred Order and Cyclical. **Resource** Name Name of the resource to include in the resource set. Applies only when Type is **Resource**. **Tally** Name Name of the tally within the tally set. Applies only when Type is **Tally**. **Counter** Name Name of the counter within the counter set. Applies only when Type is **Counter**. **Entity** Type Name of the entity type within the entity type set. Applies only when Type is **Entity**. Picture Name Name of the picture within the picture set. Applies only when Type is **Entity** Picture.

Definition at line 55 of file [Set.h](#).

6.116.2 Constructor & Destructor Documentation

6.116.2.1 Set()

```
Set::Set (
    Model * model,
    std::string name = "" )
```

Definition at line 16 of file [Set.cpp](#).

Here is the caller graph for this function:



6.116.2.2 ~Set()

```
virtual Set::~Set ( ) [virtual], [default]
```

6.116.3 Member Function Documentation

6.116.3.1 _check()

```
bool Set::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 71 of file [Set.cpp](#).

6.116.3.2 _getParserChangesInformation()

```
ParserChangesInformation * Set::_getParserChangesInformation ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 77 of file [Set.cpp](#).

6.116.3.3 _loadInstance()

```
bool Set::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 52 of file [Set.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.116.3.4 `_saveInstance()`

```
std::map< std::string, std::string > * Set::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 64 of file [Set.cpp](#).

Here is the call graph for this function:

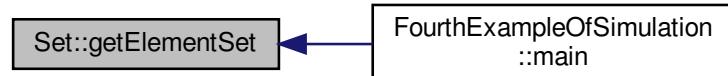


6.116.3.5 `getElementSet()`

```
List< ModelElement * > * Set::getElementSet ( ) const
```

Definition at line 33 of file [Set.cpp](#).

Here is the caller graph for this function:

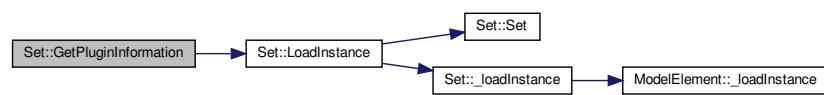


6.116.3.6 `GetPluginInformation()`

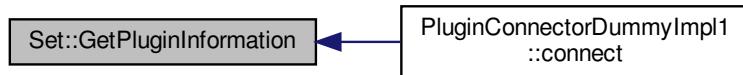
```
PluginInformation * Set::GetPluginInformation ( ) [static]
```

Definition at line 37 of file [Set.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.116.3.7 getSetOfType()

```
std::string Set::getSetOfType () const
```

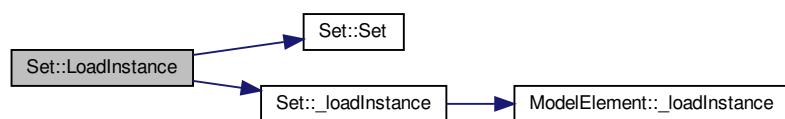
Definition at line 29 of file [Set.cpp](#).

6.116.3.8 LoadInstance()

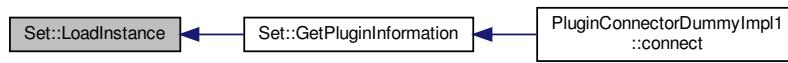
```
ModelElement * Set::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 42 of file [Set.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.116.3.9 `setSetOfType()`

```
void Set::setSetOfType ( std::string _setOfType )
```

Definition at line [25](#) of file [Set.cpp](#).

Here is the caller graph for this function:



6.116.3.10 `show()`

```
std::string Set::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [20](#) of file [Set.cpp](#).

Here is the call graph for this function:



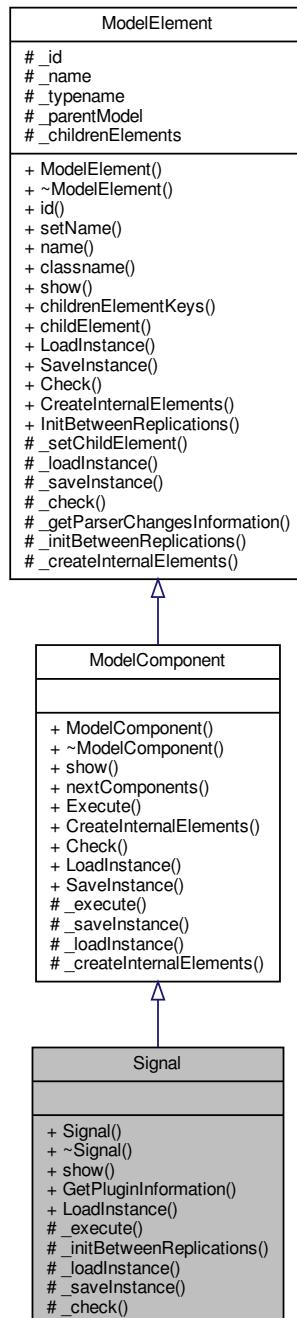
The documentation for this class was generated from the following files:

- [Set.h](#)
- [Set.cpp](#)

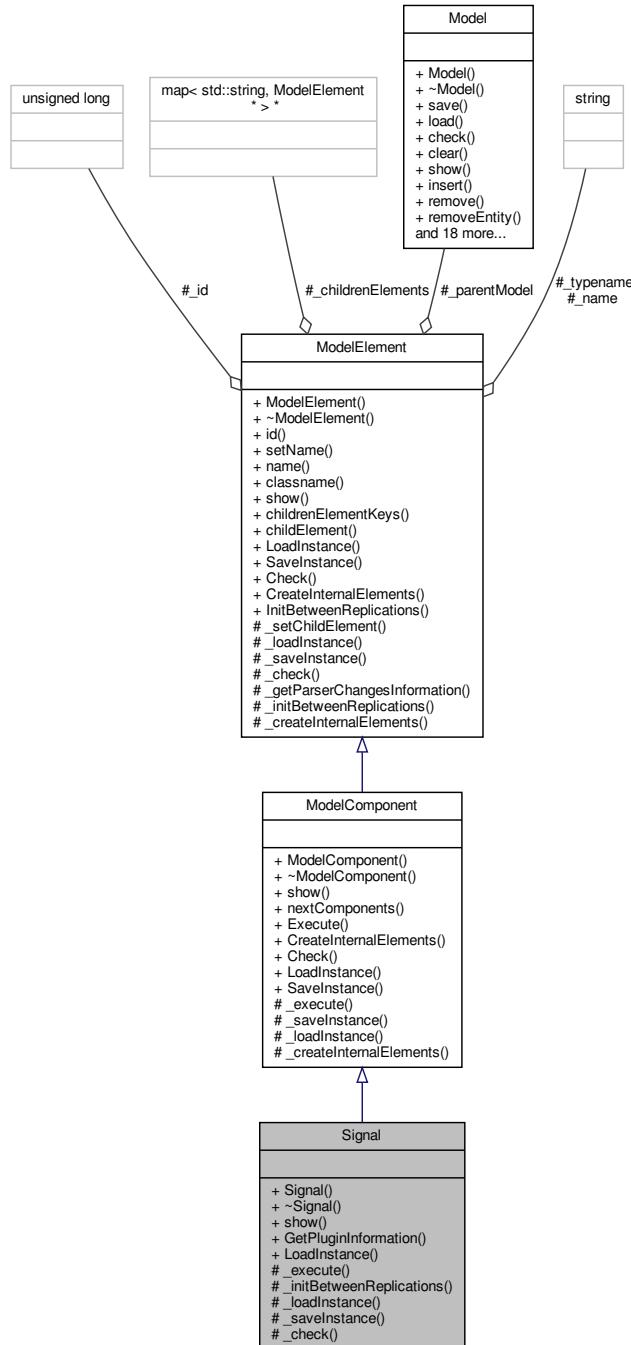
6.117 Signal Class Reference

```
#include <Signal.h>
```

Inheritance diagram for Signal:



Collaboration diagram for Signal:



Public Member Functions

- **Signal (Model *model, std::string name="")**
- virtual **~Signal ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.117.1 Detailed Description

Signal module DESCRIPTION The **Signal** module sends a signal value to each **Hold** module in the model set to Wait for **Signal** and releases the maximum specified number of entities. When an entity arrives at a **Signal** module, the signal is evaluated and the signal code is sent. At this time, entities at **Hold** modules that are waiting for the same signal are removed from their queues. The entity sending the signal continues processing until it encounters a delay, enters a queue, or is disposed. TYPICAL USES Analyzing traffic patterns at an intersection (signal when the light turns green) Signaling an operator to complete an order that was waiting for a component part PROMPT TS Prompt Description Name Unique module identifier displayed on the module shape. **Signal** Value Value of the signal to be sent to entities in **Hold** modules. Limit Maximum number of entities that are to be released from any **Hold** modules when the signal is received.

Definition at line 38 of file [Signal.h](#).

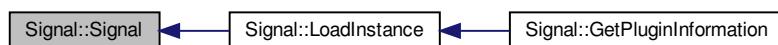
6.117.2 Constructor & Destructor Documentation

6.117.2.1 Signal()

```
Signal::Signal (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Signal.cpp](#).

Here is the caller graph for this function:



6.117.2.2 ~Signal()

```
virtual Signal::~Signal ( ) [virtual], [default]
```

6.117.3 Member Function Documentation

6.117.3.1 _check()

```
bool Signal::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Signal.cpp](#).

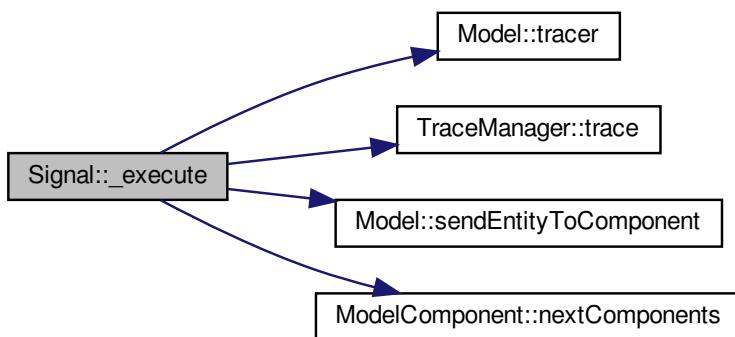
6.117.3.2 _execute()

```
void Signal::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Signal.cpp](#).

Here is the call graph for this function:



6.117.3.3 `_initBetweenReplications()`

```
void Signal::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Signal.cpp](#).

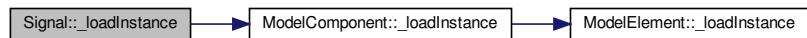
6.117.3.4 `_loadInstance()`

```
bool Signal::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

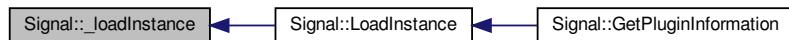
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Signal.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



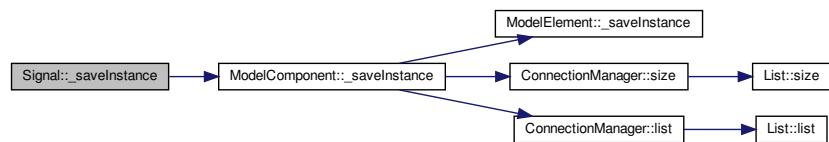
6.117.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Signal::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Signal.cpp](#).

Here is the call graph for this function:

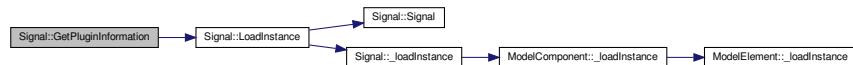


6.117.3.6 GetPluginInformation()

```
PluginInformation * Signal::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Signal.cpp](#).

Here is the call graph for this function:



6.117.3.7 LoadInstance()

```
ModelComponent * Signal::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Signal.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



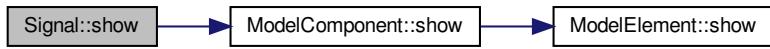
6.117.3.8 show()

```
std::string Signal::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Signal.cpp](#).

Here is the call graph for this function:



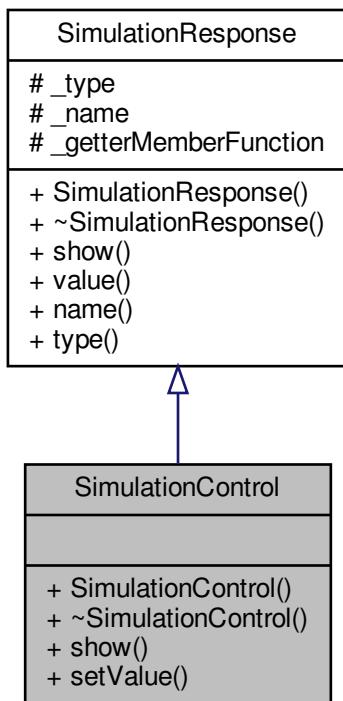
The documentation for this class was generated from the following files:

- [Signal.h](#)
- [Signal.cpp](#)

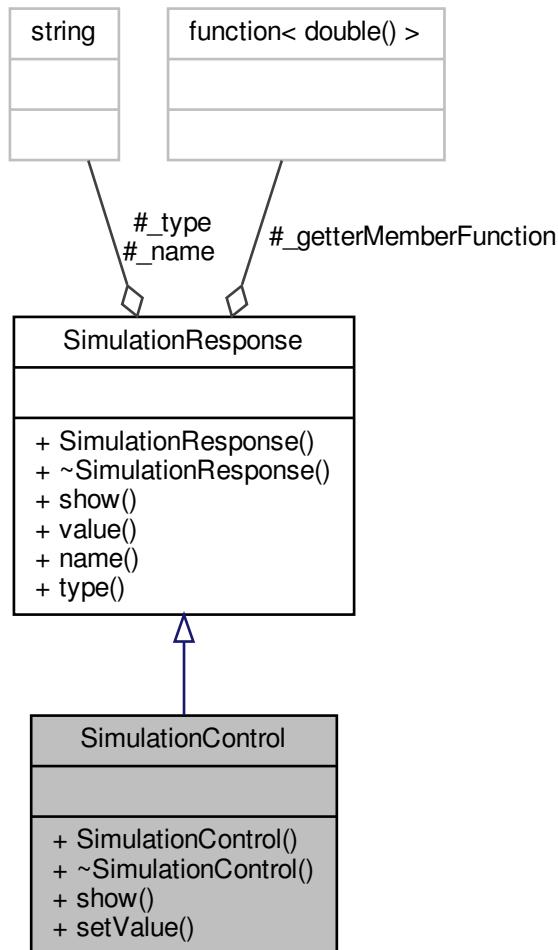
6.118 SimulationControl Class Reference

```
#include <SimulationControl.h>
```

Inheritance diagram for SimulationControl:



Collaboration diagram for SimulationControl:



Public Member Functions

- `SimulationControl (std::string type, std::string name, GetterMember getterMember, SetterMember setterMember)`
- `virtual ~SimulationControl ()=default`
- `std::string show ()`
- `void setValue (double value)`

Additional Inherited Members

6.118.1 Detailed Description

Represents any possible parameter or control for a simulation. Any element or event the model can declare one of its own attribute as a simulation control. It just have to create a `SimulationControl` object, passing the access to the methods that gets and sets the control value and including this `SimulationControl` in the corresponding list of the model

Definition at line 23 of file [SimulationControl.h](#).

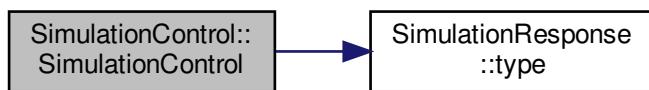
6.118.2 Constructor & Destructor Documentation

6.118.2.1 SimulationControl()

```
SimulationControl::SimulationControl (
    std::string type,
    std::string name,
    GetterMember getterMember,
    SetterMember setterMember )
```

Definition at line 16 of file [SimulationControl.cpp](#).

Here is the call graph for this function:



6.118.2.2 ~SimulationControl()

```
virtual SimulationControl::~SimulationControl () [virtual], [default]
```

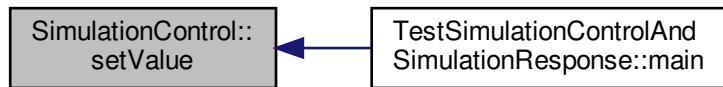
6.118.3 Member Function Documentation

6.118.3.1 setValue()

```
void SimulationControl::setValue (
    double value )
```

Definition at line 26 of file [SimulationControl.cpp](#).

Here is the caller graph for this function:



6.118.3.2 show()

```
std::string SimulationControl::show ( )
```

Definition at line 22 of file [SimulationControl.cpp](#).

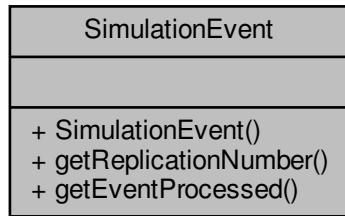
The documentation for this class was generated from the following files:

- [SimulationControl.h](#)
- [SimulationControl.cpp](#)

6.119 SimulationEvent Class Reference

```
#include <OnEventManager.h>
```

Collaboration diagram for SimulationEvent:



Public Member Functions

- [SimulationEvent](#) (unsigned int replicationNumber, [Event](#) *event)
- unsigned int [getReplicationNumber](#) () const
- [Event](#) * [getEventProcessed](#) () const

6.119.1 Detailed Description

Definition at line 27 of file [OnEventManager.h](#).

6.119.2 Constructor & Destructor Documentation

6.119.2.1 SimulationEvent()

```
SimulationEvent::SimulationEvent (
    unsigned int replicationNumber,
    Event * event ) [inline]
```

Definition at line 30 of file [OnEventManager.h](#).

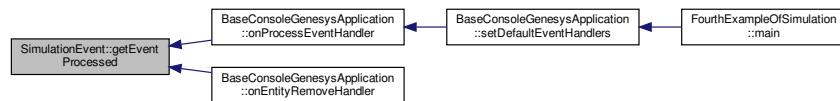
6.119.3 Member Function Documentation

6.119.3.1 getEventProcessed()

```
Event* SimulationEvent::getEventProcessed () const [inline]
```

Definition at line 40 of file [OnEventManager.h](#).

Here is the caller graph for this function:

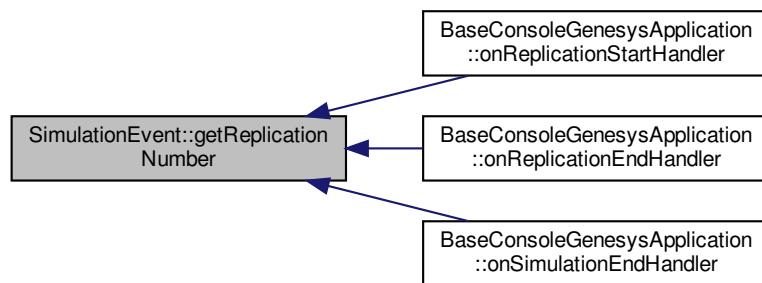


6.119.3.2 getReplicationNumber()

```
unsigned int SimulationEvent::getReplicationNumber () const [inline]
```

Definition at line 36 of file [OnEventManager.h](#).

Here is the caller graph for this function:



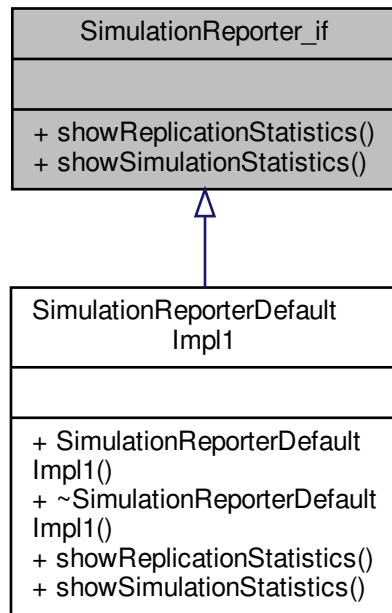
The documentation for this class was generated from the following file:

- [OnEventManager.h](#)

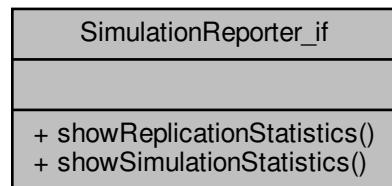
6.120 SimulationReporter_if Class Reference

```
#include <SimulationReporter_if.h>
```

Inheritance diagram for SimulationReporter_if:



Collaboration diagram for SimulationReporter_if:



Public Member Functions

- virtual void `showReplicationStatistics ()=0`
- virtual void `showSimulationStatistics ()=0`

6.120.1 Detailed Description

Definition at line 20 of file [SimulationReporter_if.h](#).

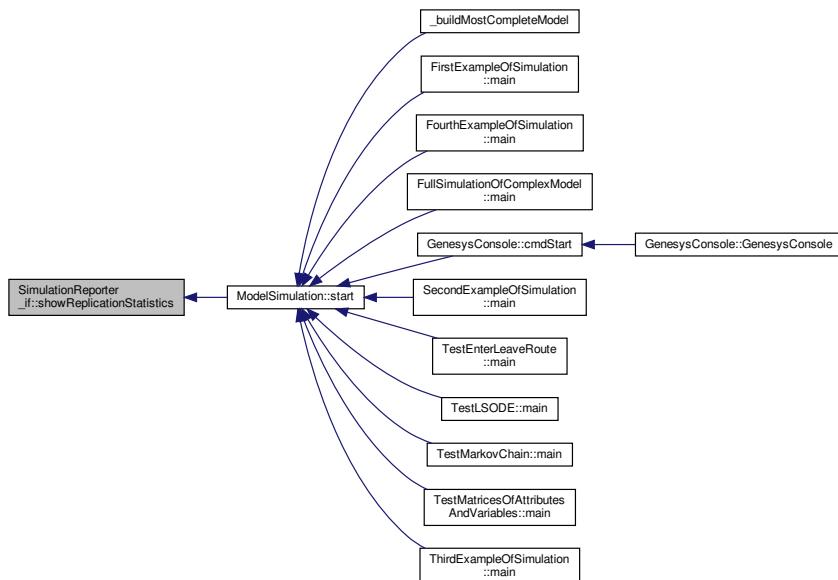
6.120.2 Member Function Documentation

6.120.2.1 showReplicationStatistics()

```
virtual void SimulationReporter_if::showReplicationStatistics ( ) [pure virtual]
```

Implemented in [SimulationReporterDefaultImpl1](#).

Here is the caller graph for this function:

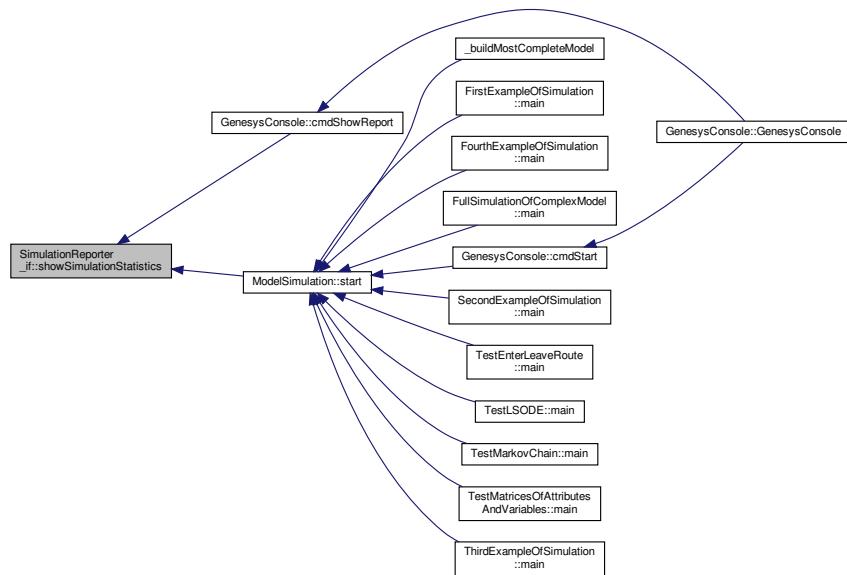


6.120.2.2 showSimulationStatistics()

```
virtual void SimulationReporter_if::showSimulationStatistics ( ) [pure virtual]
```

Implemented in [SimulationReporterDefaultImpl1](#).

Here is the caller graph for this function:



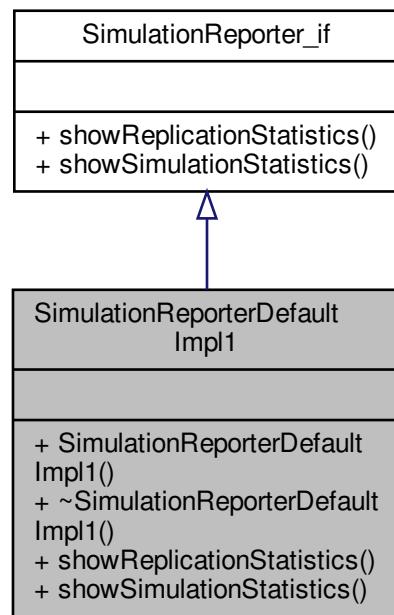
The documentation for this class was generated from the following file:

- [SimulationReporter_if.h](#)

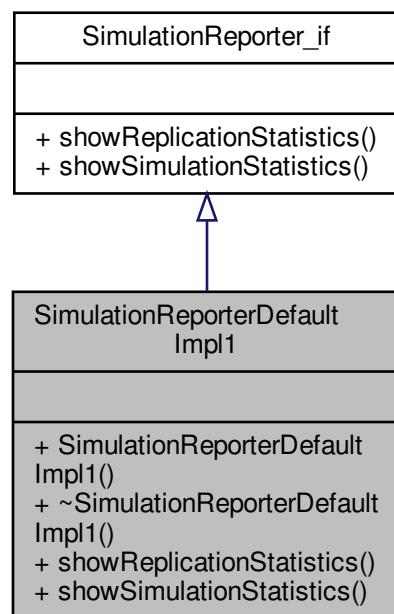
6.121 SimulationReporterDefaultImpl1 Class Reference

```
#include <SimulationReporterDefaultImpl1.h>
```

Inheritance diagram for SimulationReporterDefaultImpl1:



Collaboration diagram for SimulationReporterDefaultImpl1:



Public Member Functions

- `SimulationReporterDefaultImpl1 (ModelSimulation *simulation, Model *model, List< ModelElement *> *statsCountersSimulation)`
- `virtual ~SimulationReporterDefaultImpl1 ()=default`
- `virtual void showReplicationStatistics ()`
- `virtual void showSimulationStatistics ()`

6.121.1 Detailed Description

Class that implements `SimulationReporter_if` interface and is responsible for building and showing replication and simulation reports

Definition at line 24 of file `SimulationReporterDefaultImpl1.h`.

6.121.2 Constructor & Destructor Documentation

6.121.2.1 `SimulationReporterDefaultImpl1()`

```
SimulationReporterDefaultImpl1::SimulationReporterDefaultImpl1 (
    ModelSimulation * simulation,
    Model * model,
    List< ModelElement *> * statsCountersSimulation )
```

Definition at line 19 of file `SimulationReporterDefaultImpl1.cpp`.

6.121.2.2 `~SimulationReporterDefaultImpl1()`

```
virtual SimulationReporterDefaultImpl1::~SimulationReporterDefaultImpl1 ( ) [virtual], [default]
```

6.121.3 Member Function Documentation

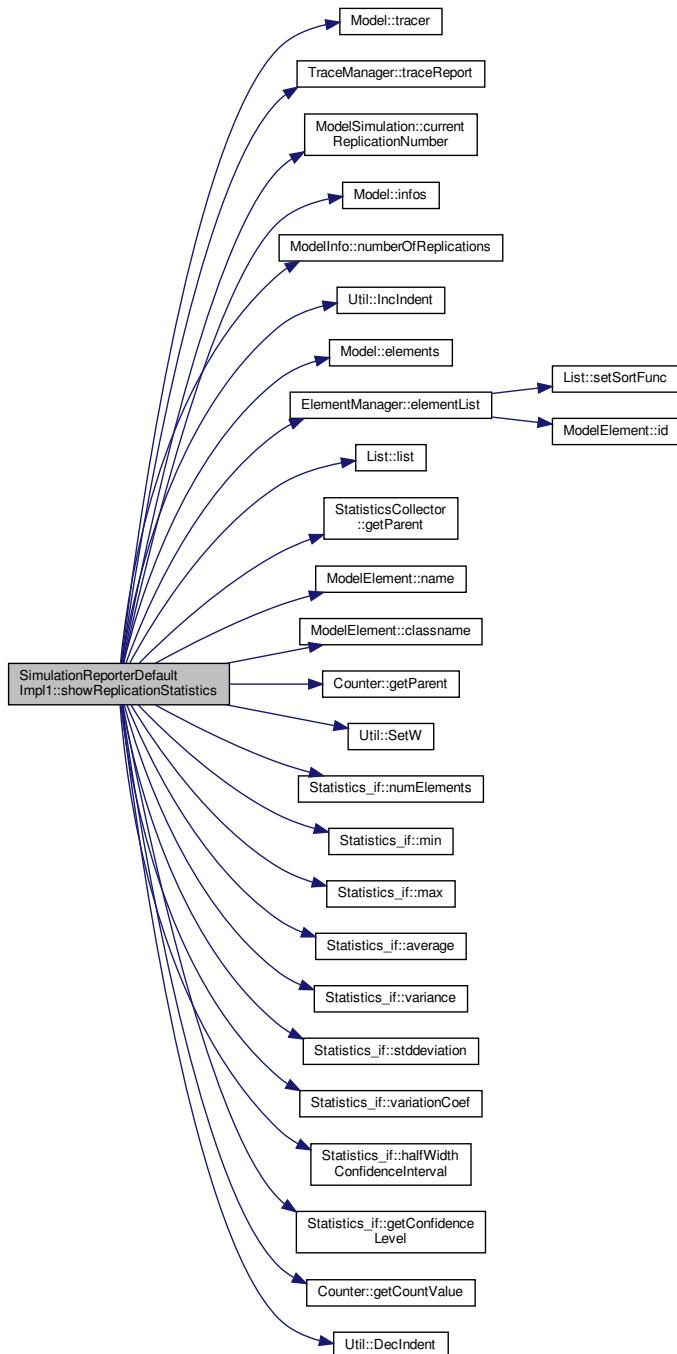
6.121.3.1 showReplicationStatistics()

```
void SimulationReporterDefaultImpl1::showReplicationStatistics () [virtual]
```

Implements [SimulationReporter_if](#).

Definition at line 26 of file [SimulationReporterDefaultImpl1.cpp](#).

Here is the call graph for this function:



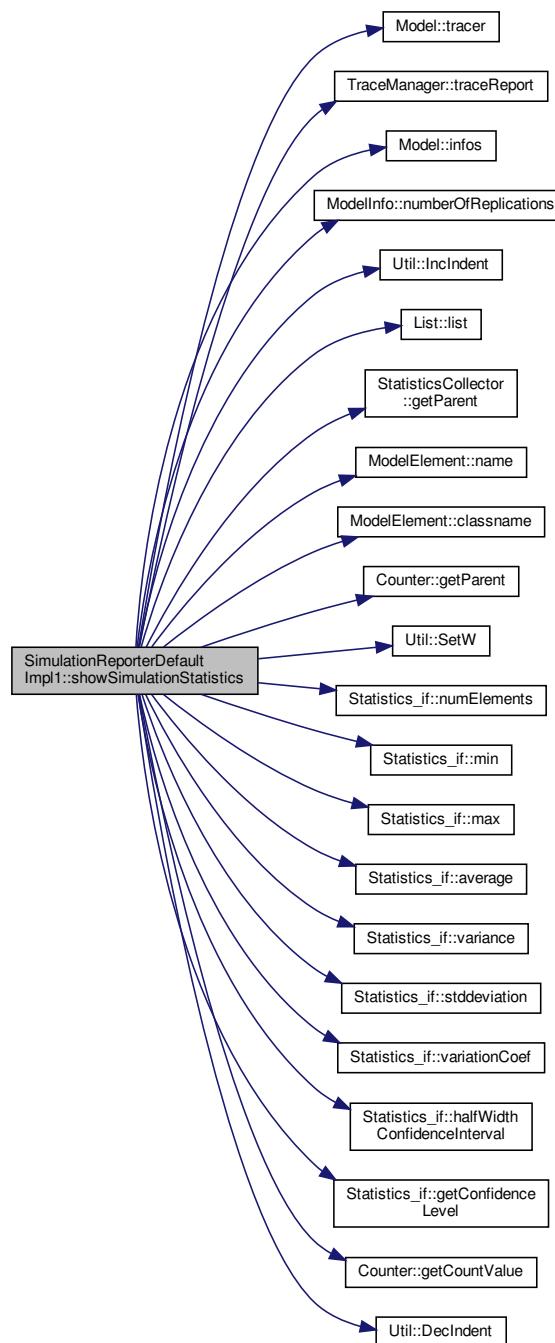
6.121.3.2 showSimulationStatistics()

```
void SimulationReporterDefaultImpl1::showSimulationStatistics ( ) [virtual]
```

Implements [SimulationReporter_if](#).

Definition at line 131 of file [SimulationReporterDefaultImpl1.cpp](#).

Here is the call graph for this function:



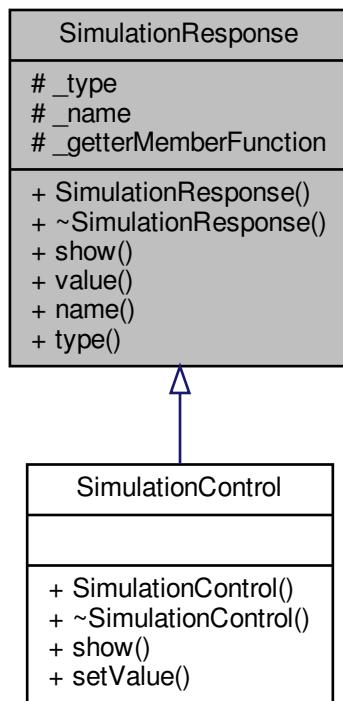
The documentation for this class was generated from the following files:

- [SimulationReporterDefaultImpl1.h](#)
- [SimulationReporterDefaultImpl1.cpp](#)

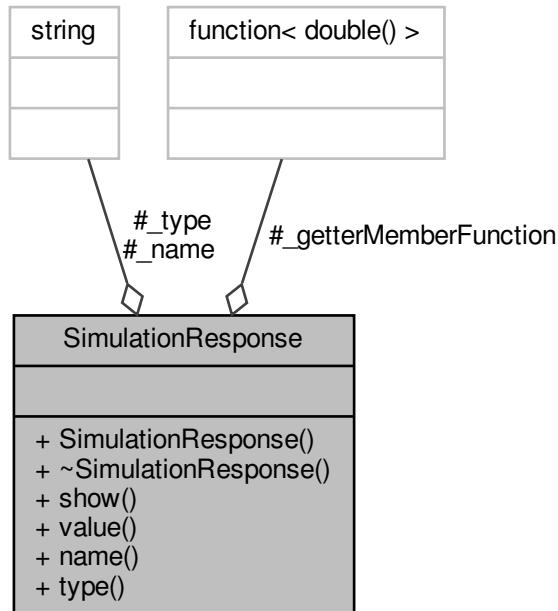
6.122 SimulationResponse Class Reference

```
#include <SimulationResponse.h>
```

Inheritance diagram for SimulationResponse:



Collaboration diagram for SimulationResponse:



Public Member Functions

- `SimulationResponse (std::string type, std::string name, GetterMember getterMember)`
- virtual `~SimulationResponse ()=default`
- `std::string show ()`
- `double value ()`
- `std::string name () const`
- `std::string type () const`

Protected Attributes

- `std::string _type`
- `std::string _name`
- `GetterMember _getterMemberFunction`

6.122.1 Detailed Description

Represents any possible response of a simulation. Any element or event the model can declare one of its own attribute as a simulation response. It just have to create a `SimulationResponse` object, passing the access to the method that gets the response value and including this `SimulationResponse` in the corresponding list of the model

Definition at line 23 of file `SimulationResponse.h`.

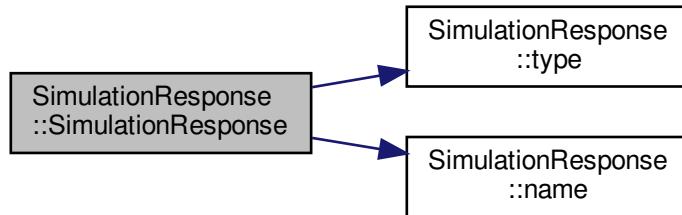
6.122.2 Constructor & Destructor Documentation

6.122.2.1 SimulationResponse()

```
SimulationResponse::SimulationResponse (   
    std::string type,  
    std::string name,  
    GetterMember getterMember )
```

Definition at line 17 of file [SimulationResponse.cpp](#).

Here is the call graph for this function:



6.122.2.2 ~SimulationResponse()

```
virtual SimulationResponse::~SimulationResponse ( ) [virtual], [default]
```

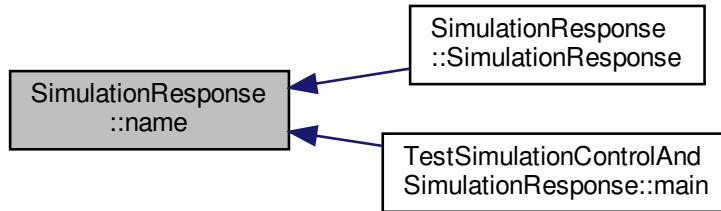
6.122.3 Member Function Documentation

6.122.3.1 name()

```
std::string SimulationResponse::name ( ) const
```

Definition at line 28 of file [SimulationResponse.cpp](#).

Here is the caller graph for this function:



6.122.3.2 show()

```
std::string SimulationResponse::show ( )
```

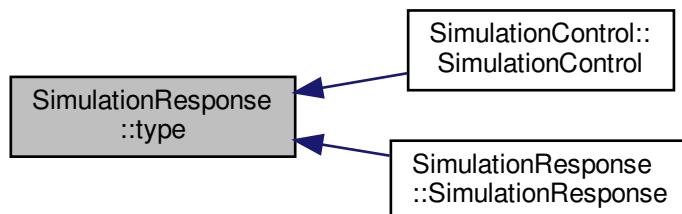
Definition at line 24 of file [SimulationResponse.cpp](#).

6.122.3.3 type()

```
std::string SimulationResponse::type ( ) const
```

Definition at line 32 of file [SimulationResponse.cpp](#).

Here is the caller graph for this function:



6.122.3.4 value()

```
double SimulationResponse::value ()
```

Definition at line 36 of file [SimulationResponse.cpp](#).

Here is the caller graph for this function:



6.122.4 Member Data Documentation

6.122.4.1 _getterMemberFunction

```
GetterMember SimulationResponse::_getterMemberFunction [protected]
```

Definition at line 36 of file [SimulationResponse.h](#).

6.122.4.2 _name

```
std::string SimulationResponse::_name [protected]
```

Definition at line 35 of file [SimulationResponse.h](#).

6.122.4.3 _type

```
std::string SimulationResponse::_type [protected]
```

Definition at line 34 of file [SimulationResponse.h](#).

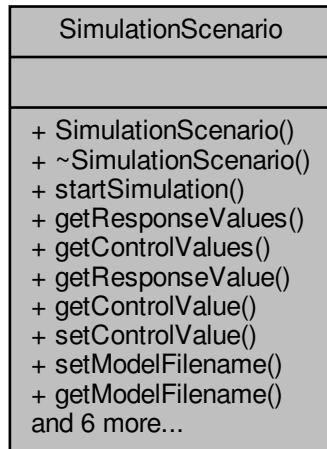
The documentation for this class was generated from the following files:

- [SimulationResponse.h](#)
- [SimulationResponse.cpp](#)

6.123 SimulationScenario Class Reference

```
#include <SimulationScenario.h>
```

Collaboration diagram for SimulationScenario:



Public Member Functions

- `SimulationScenario ()`
- virtual `~SimulationScenario ()=default`
- `bool startSimulation (std::string *errorMessage)`
- `std::list< std::pair< std::string, double > * > * getResponseValues () const`
- `std::list< std::pair< std::string, double > * > * getControlValues () const`
- `double getResponseBody (std::string responseName)`
- `double getControlBody (std::string controlName)`
- `void setControlValue (std::string controlName, double value)`
- `void setModelFilename (std::string _modelFilename)`
- `std::string getModelFilename () const`
- `void setScenarioName (std::string _name)`
- `std::string getScenarioName () const`
- `std::list< SimulationResponse * > * getSelectedResponses () const`
- `std::list< SimulationControl * > * getSelectedControls () const`
- `void setScenarioDescription (std::string _scenarioDescription)`
- `std::string getScenarioDescription () const`

6.123.1 Detailed Description

Represents a scenario where a specific model (defined by ModelFilename) will be simulated. To each scenario will be associated a set of `SimulationControl` and `SimulationResponse`, and their values are set to the scenario by the ProcessAnalysyer.

Definition at line 25 of file [SimulationScenario.h](#).

6.123.2 Constructor & Destructor Documentation

6.123.2.1 SimulationScenario()

```
SimulationScenario::SimulationScenario ( )
```

Definition at line 16 of file [SimulationScenario.cpp](#).

6.123.2.2 ~SimulationScenario()

```
virtual SimulationScenario::~SimulationScenario ( ) [virtual], [default]
```

6.123.3 Member Function Documentation

6.123.3.1 getControlValue()

```
double SimulationScenario::getControlValue ( std::string controlName )
```

6.123.3.2 getControlValues()

```
std::list<std::pair<std::string, double>*>* SimulationScenario::getControlValues ( ) const
```

6.123.3.3 getModelFilename()

```
std::string SimulationScenario::getModelFilename ( ) const
```

Definition at line 43 of file [SimulationScenario.cpp](#).

6.123.3.4 getResponseValue()

```
double SimulationScenario::getResponseValue ( std::string responseName )
```

6.123.3.5 getResponseValues()

```
std::list<std::pair<std::string, double>*>* SimulationScenario::getResponseValues ( ) const
```

The final result of the simulationScenario

6.123.3.6 getScenarioDescription()

```
std::string SimulationScenario::getScenarioDescription ( ) const
```

Definition at line [60](#) of file [SimulationScenario.cpp](#).

6.123.3.7 getScenarioName()

```
std::string SimulationScenario::getScenarioName ( ) const
```

Definition at line [34](#) of file [SimulationScenario.cpp](#).

6.123.3.8 getSelectedControls()

```
std::list< SimulationControl * > * SimulationScenario::getSelectedControls ( ) const
```

Definition at line [52](#) of file [SimulationScenario.cpp](#).

6.123.3.9 getSelectedResponses()

```
std::list< SimulationResponse * > * SimulationScenario::getSelectedResponses ( ) const
```

Definition at line [48](#) of file [SimulationScenario.cpp](#).

6.123.3.10 setControlValue()

```
void SimulationScenario::setControlValue (
    std::string controlName,
    double value )
```

6.123.3.11 setModelFilename()

```
void SimulationScenario::setModelFilename (
    std::string _modelFilename )
```

Definition at line [39](#) of file [SimulationScenario.cpp](#).

6.123.3.12 setScenarioDescription()

```
void SimulationScenario::setScenarioDescription (
    std::string _scenarioDescription )
```

Definition at line [56](#) of file [SimulationScenario.cpp](#).

6.123.3.13 setScenarioName()

```
void SimulationScenario::setScenarioName ( std::string _name )
```

Definition at line 30 of file [SimulationScenario.cpp](#).

6.123.3.14 startSimulation()

```
bool SimulationScenario::startSimulation ( std::string * errorMessage )
```

Definition at line 20 of file [SimulationScenario.cpp](#).

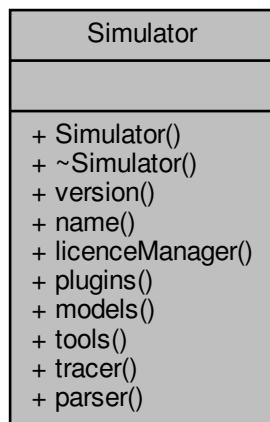
The documentation for this class was generated from the following files:

- [SimulationScenario.h](#)
- [SimulationScenario.cpp](#)

6.124 Simulator Class Reference

```
#include <Simulator.h>
```

Collaboration diagram for Simulator:



Public Member Functions

- `Simulator ()`
- `virtual ~Simulator ()=default`
- `std::string version () const`
- `std::string name () const`
- `LicenceManager * licenceManager () const`
- `PluginManager * plugins () const`
- `ModelManager * models () const`
- `ToolManager * tools () const`
- `TraceManager * tracer () const`
- `ParserManager * parser () const`

6.124.1 Detailed Description

The main class of the ReGenesys KERNEL simulation. It gives access to simulation models and tools. Simulation is the top level class and is supposed to be available to application as a dynamic linked library.

Definition at line 32 of file [Simulator.h](#).

6.124.2 Constructor & Destructor Documentation

6.124.2.1 Simulator()

```
Simulator::Simulator ( )
```

Definition at line 18 of file [Simulator.cpp](#).

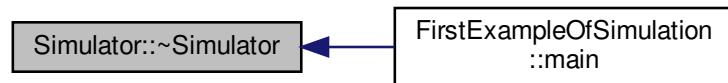
Here is the call graph for this function:



6.124.2.2 ~Simulator()

```
virtual Simulator::~Simulator ( ) [virtual], [default]
```

Here is the caller graph for this function:



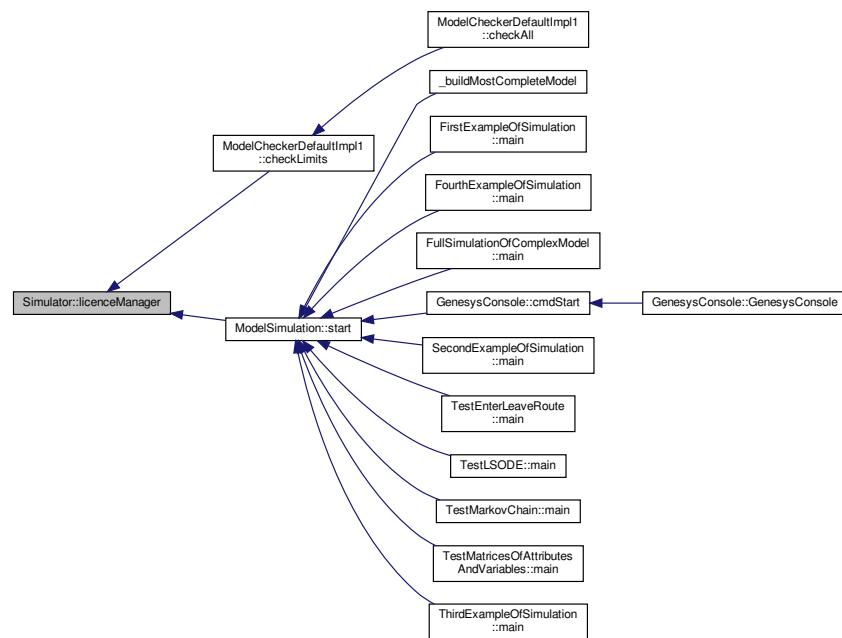
6.124.3 Member Function Documentation

6.124.3.1 licenceManager()

```
LicenceManager * Simulator::licenceManager ( ) const
```

Definition at line 59 of file [Simulator.cpp](#).

Here is the caller graph for this function:

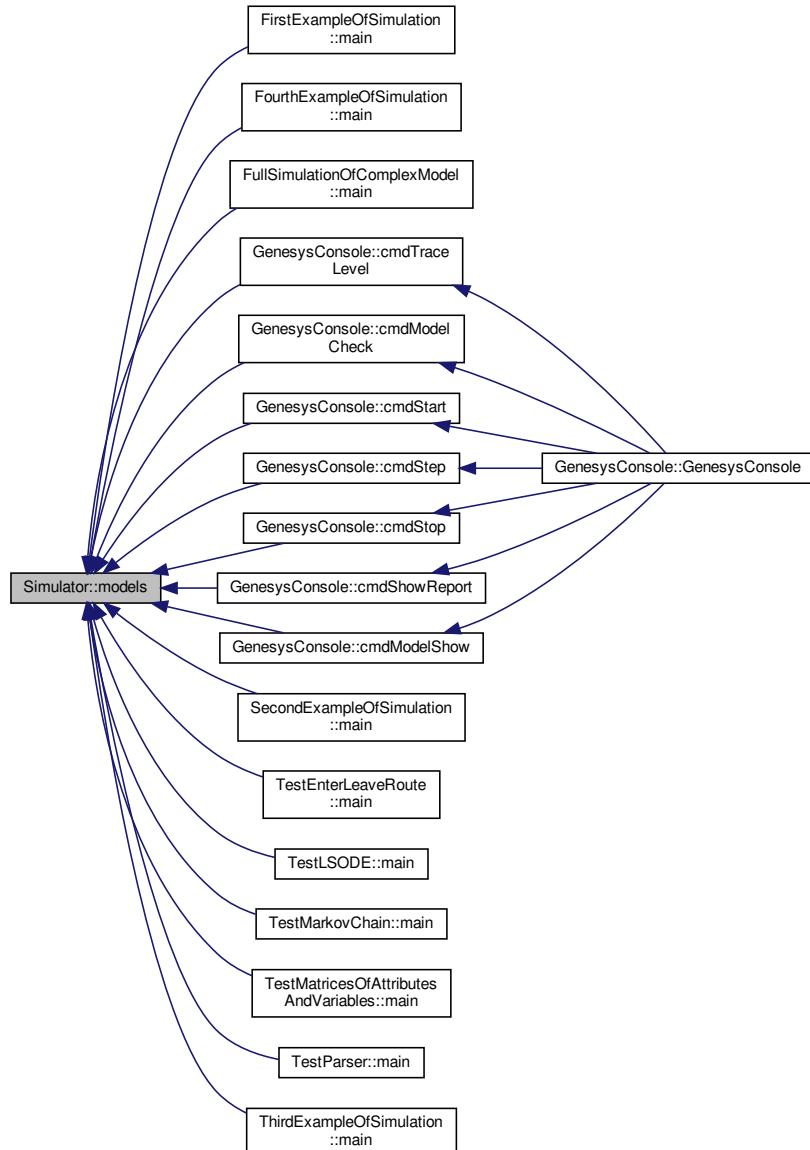


6.124.3.2 models()

```
ModelManager * Simulator::models ( ) const
```

Definition at line 35 of file [Simulator.cpp](#).

Here is the caller graph for this function:

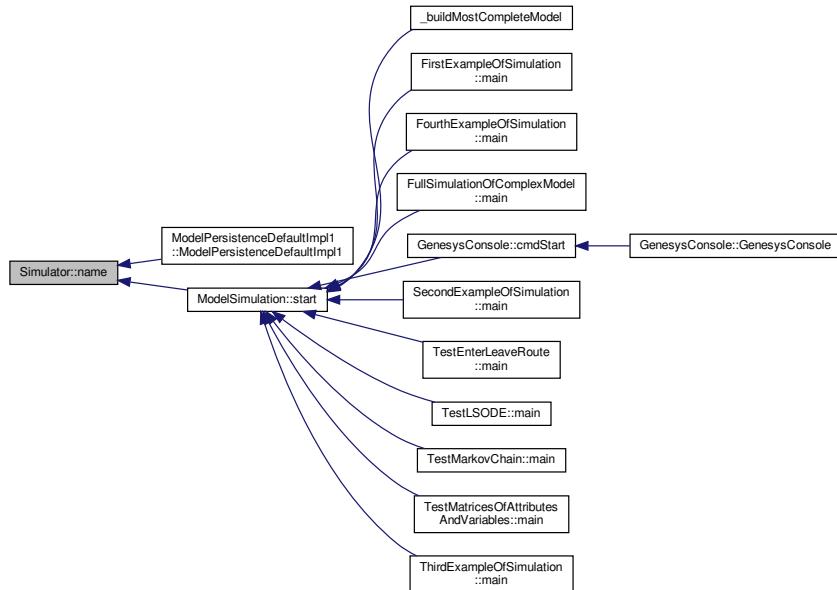


6.124.3.3 name()

```
std::string Simulator::name ( ) const
```

Definition at line 55 of file [Simulator.cpp](#).

Here is the caller graph for this function:



6.124.3.4 `parser()`

```
ParserManager * Simulator::parser ( ) const
```

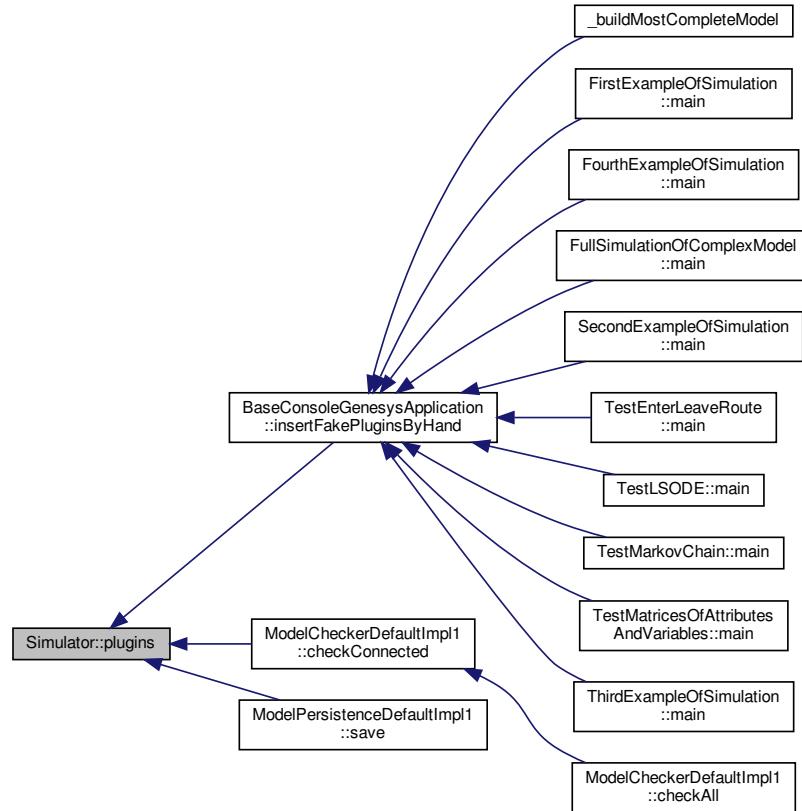
Definition at line 47 of file [Simulator.cpp](#).

6.124.3.5 `plugins()`

```
PluginManager * Simulator::plugins ( ) const
```

Definition at line 31 of file [Simulator.cpp](#).

Here is the caller graph for this function:

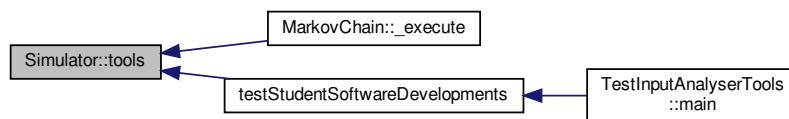


6.124.3.6 tools()

```
ToolManager * Simulator::tools( ) const
```

Definition at line 39 of file [Simulator.cpp](#).

Here is the caller graph for this function:

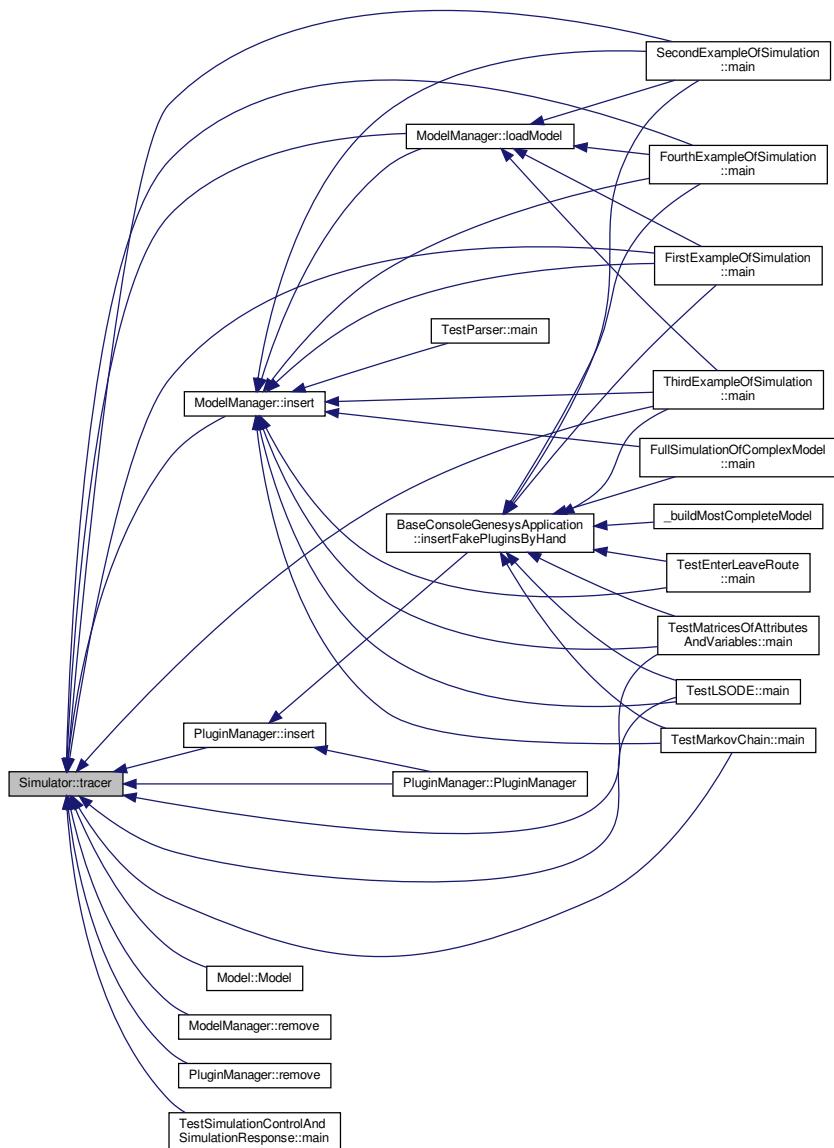


6.124.3.7 tracer()

```
TraceManager * Simulator::tracer ( ) const
```

Definition at line 43 of file [Simulator.cpp](#).

Here is the caller graph for this function:

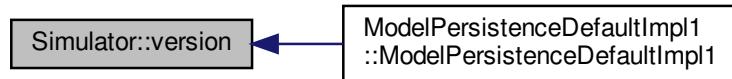


6.124.3.8 version()

```
std::string Simulator::version ( ) const
```

Definition at line 51 of file [Simulator.cpp](#).

Here is the caller graph for this function:



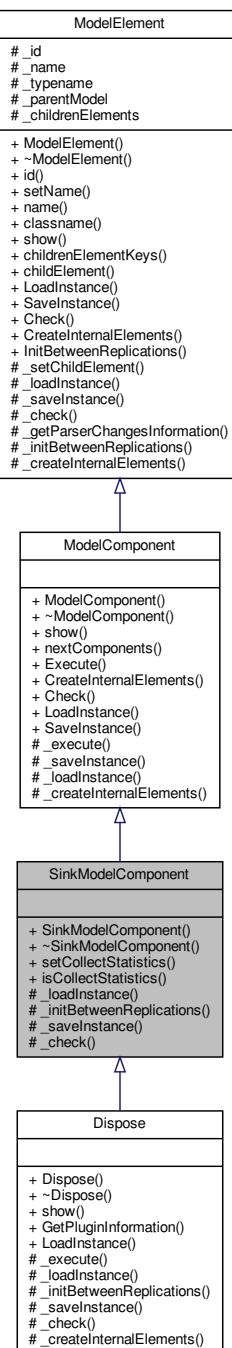
The documentation for this class was generated from the following files:

- [Simulator.h](#)
- [Simulator.cpp](#)

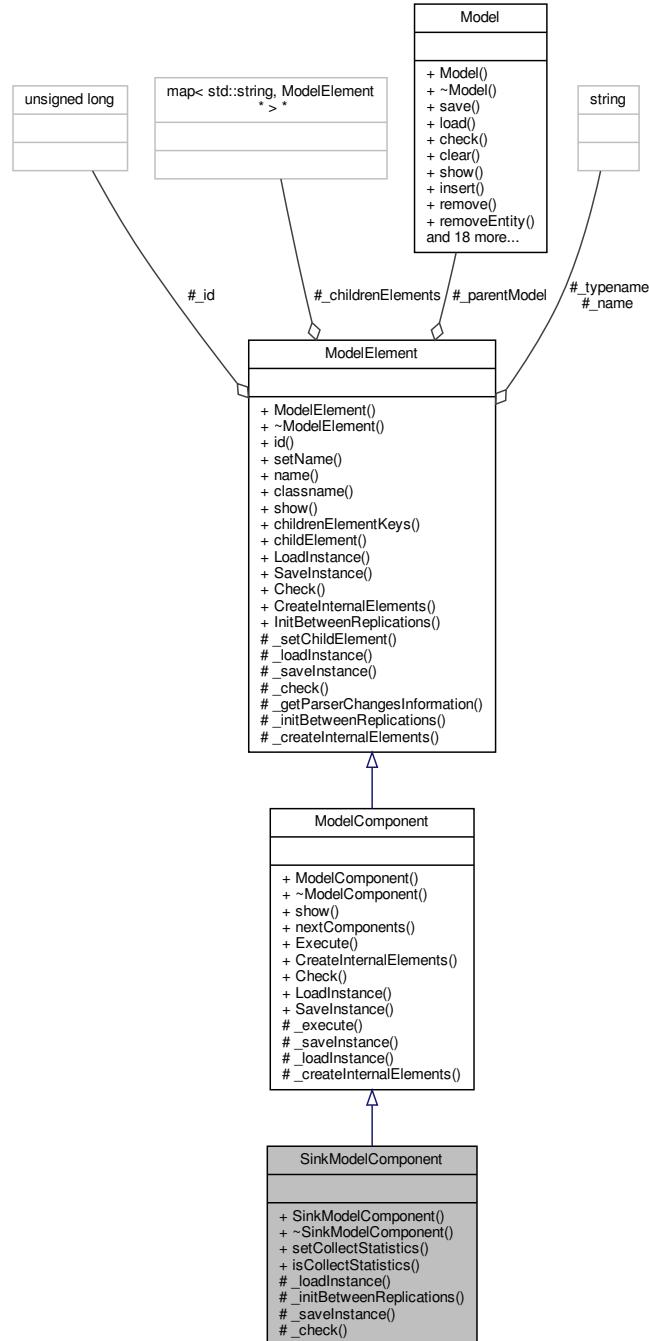
6.125 SinkModelComponent Class Reference

```
#include <SinkModelComponent.h>
```

Inheritance diagram for SinkModelComponent:



Collaboration diagram for SinkModelComponent:



Public Member Functions

- `SinkModelComponent (Model *model, std::string componentTypename, std::string name="")`
- `virtual ~SinkModelComponent ()=default`
- `void setCollectStatistics (bool _collectStatistics)`
- `bool isCollectStatistics () const`

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual void [_initBetweenReplications](#) ()
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)

Additional Inherited Members

6.125.1 Detailed Description

This class is the basis for any component representing the end of a process flow, such as a [Dispose](#). It can remove entities from the system and collect statistics.

Definition at line [23](#) of file [SinkModelComponent.h](#).

6.125.2 Constructor & Destructor Documentation

6.125.2.1 SinkModelComponent()

```
SinkModelComponent::SinkModelComponent (
    Model * model,
    std::string componentTypename,
    std::string name = "")
```

Definition at line [16](#) of file [SinkModelComponent.cpp](#).

6.125.2.2 ~SinkModelComponent()

```
virtual SinkModelComponent::~SinkModelComponent () [virtual], [default]
```

6.125.3 Member Function Documentation

6.125.3.1 _check()

```
bool SinkModelComponent::_check (
    std::string * errorMessage) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Reimplemented in [Dispose](#).

Definition at line [45](#) of file [SinkModelComponent.cpp](#).

6.125.3.2 `_initBetweenReplications()`

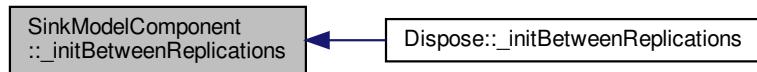
```
void SinkModelComponent::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Reimplemented in [Dispose](#).

Definition at line [36](#) of file [SinkModelComponent.cpp](#).

Here is the caller graph for this function:



6.125.3.3 `_loadInstance()`

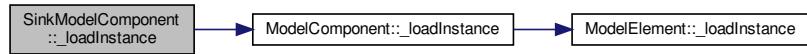
```
bool SinkModelComponent::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Reimplemented in [Dispose](#).

Definition at line [28](#) of file [SinkModelComponent.cpp](#).

Here is the call graph for this function:



6.125.3.4 `_saveInstance()`

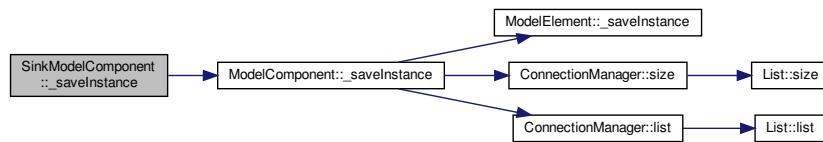
```
std::map< std::string, std::string > * SinkModelComponent::_saveInstance ( ) [protected],  
[virtual]
```

Reimplemented from [ModelComponent](#).

Reimplemented in [Dispose](#).

Definition at line 39 of file [SinkModelComponent.cpp](#).

Here is the call graph for this function:

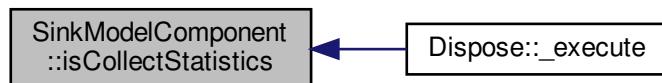


6.125.3.5 `isCollectStatistics()`

```
bool SinkModelComponent::isCollectStatistics ( ) const
```

Definition at line 24 of file [SinkModelComponent.cpp](#).

Here is the caller graph for this function:



6.125.3.6 `setCollectStatistics()`

```
void SinkModelComponent::setCollectStatistics (   
    bool _collectStatistics )
```

Definition at line 20 of file [SinkModelComponent.cpp](#).

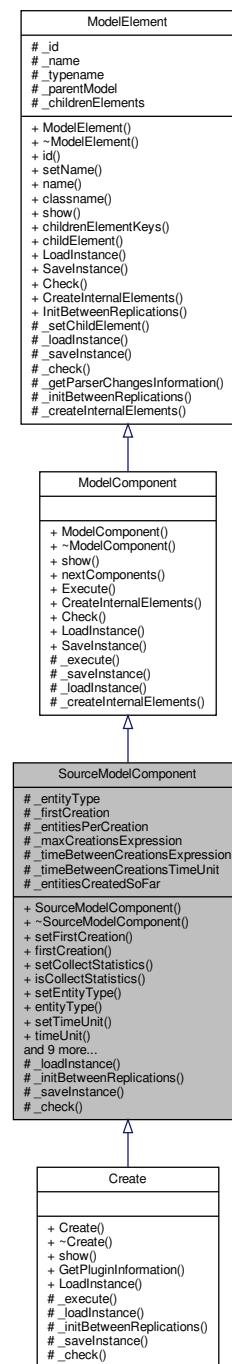
The documentation for this class was generated from the following files:

- [SinkModelComponent.h](#)
- [SinkModelComponent.cpp](#)

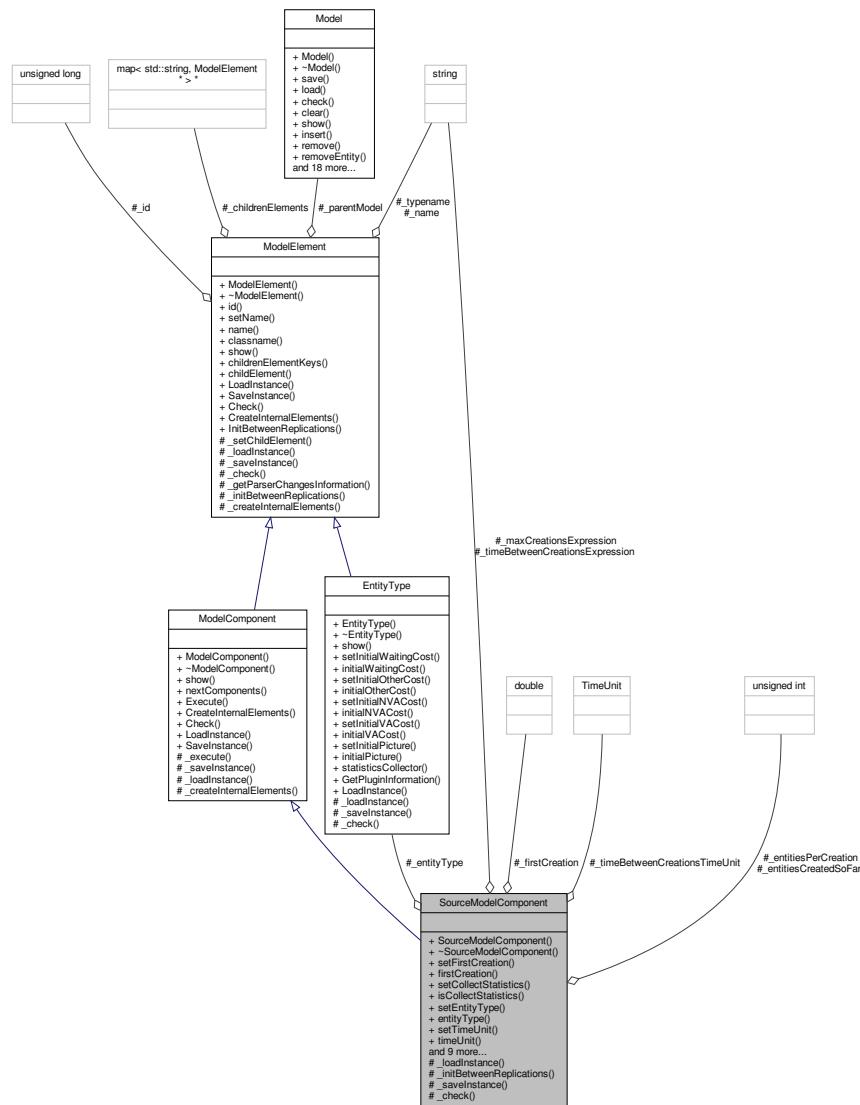
6.126 SourceModelComponent Class Reference

```
#include <SourceModelComponent.h>
```

Inheritance diagram for SourceModelComponent:



Collaboration diagram for SourceModelComponent:



Public Member Functions

- **SourceModelComponent (Model *model, std::string componentTypename, std::string name="")**
- virtual **~SourceModelComponent ()=default**
- void **setFirstCreation (double _firstCreation)**
- double **firstCreation () const**
- void **setCollectStatistics (bool _collectStatistics)**
- bool **isCollectStatistics () const**
- void **setEntityType (EntityType *_entityType)**
- **EntityType * entityType () const**
- void **setTimeUnit (Util::TimeUnit _timeUnit)**
- **Util::TimeUnit timeUnit () const**
- void **setTimeBetweenCreationsExpression (std::string _timeBetweenCreations)**
- std::string **timeBetweenCreationsExpression () const**
- void **setMaxCreations (std::string _maxCreationsExpression)**

- std::string `maxCreations () const`
- unsigned int `entitiesCreated () const`
- void `setEntitiesCreated (unsigned int _entitiesCreated)`
- void `setEntitiesPerCreation (unsigned int _entitiesPerCreation)`
- unsigned int `entitiesPerCreation () const`
- virtual std::string `show ()`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual void `_initBetweenReplications ()`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Protected Attributes

- `EntityType * _entityType`
- double `_firstCreation = 0.0`
- unsigned int `_entitiesPerCreation = 1`
- std::string `_maxCreationsExpression = std::to_string(std::numeric_limits<unsigned int>::max())`
- std::string `_timeBetweenCreationsExpression = "EXPO(1)"`
- `Util::TimeUnit _timeBetweenCreationsTimeUnit = Util::TimeUnit::second`
- unsigned int `_entitiesCreatedSoFar = 0`

Additional Inherited Members

6.126.1 Detailed Description

A source component implements the base for inserting entities into the model when its simulation is initialized. During the initialization, the new and empty future events list is populated by events of creating entities and sending them to the source components existing in the model

Definition at line 27 of file [SourceModelComponent.h](#).

6.126.2 Constructor & Destructor Documentation

6.126.2.1 SourceModelComponent()

```
SourceModelComponent::SourceModelComponent (
    Model * model,
    std::string componentTypename,
    std::string name = "" )
```

Definition at line 18 of file [SourceModelComponent.cpp](#).

6.126.2.2 ~SourceModelComponent()

```
virtual SourceModelComponent::~SourceModelComponent ( ) [virtual], [default]
```

6.126.3 Member Function Documentation

6.126.3.1 _check()

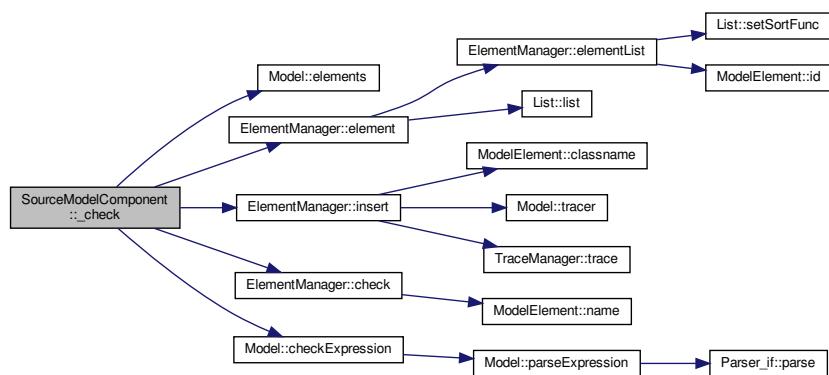
```
bool SourceModelComponent::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

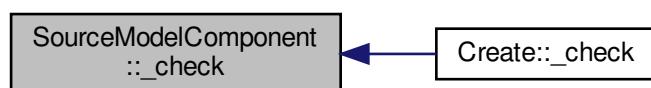
Reimplemented in [Create](#).

Definition at line 59 of file [SourceModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.126.3.2 `_initBetweenReplications()`

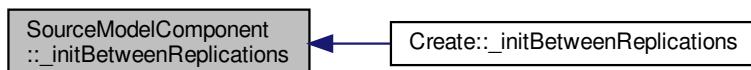
```
void SourceModelComponent::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Reimplemented in [Create](#).

Definition at line 43 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:



6.126.3.3 `_loadInstance()`

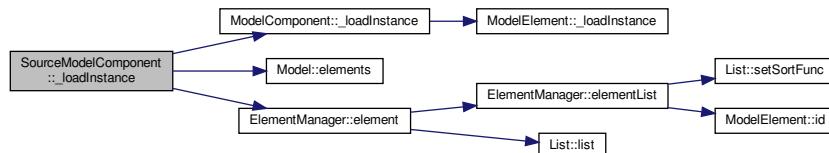
```
bool SourceModelComponent::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

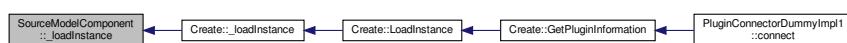
Reimplemented in [Create](#).

Definition at line 29 of file [SourceModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.126.3.4 `_saveInstance()`

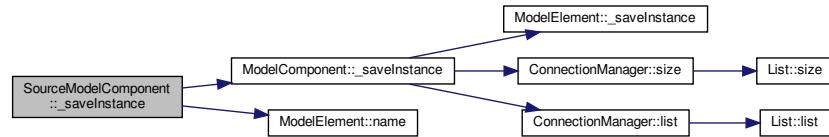
```
std::map< std::string, std::string > * SourceModelComponent::_saveInstance ( ) [protected],  
[virtual]
```

Reimplemented from [ModelComponent](#).

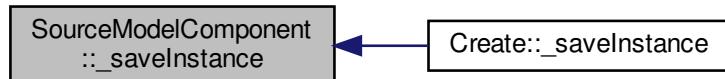
Reimplemented in [Create](#).

Definition at line [47](#) of file [SourceModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.126.3.5 `entitiesCreated()`

```
unsigned int SourceModelComponent::entitiesCreated ( ) const
```

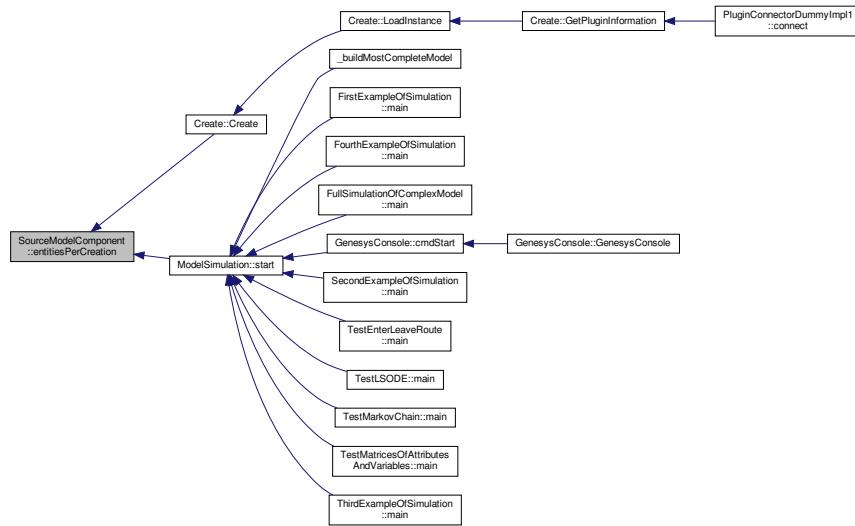
Definition at line [125](#) of file [SourceModelComponent.cpp](#).

6.126.3.6 entitiesPerCreation()

```
unsigned int SourceModelComponent::entitiesPerCreation ( ) const
```

Definition at line 138 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:

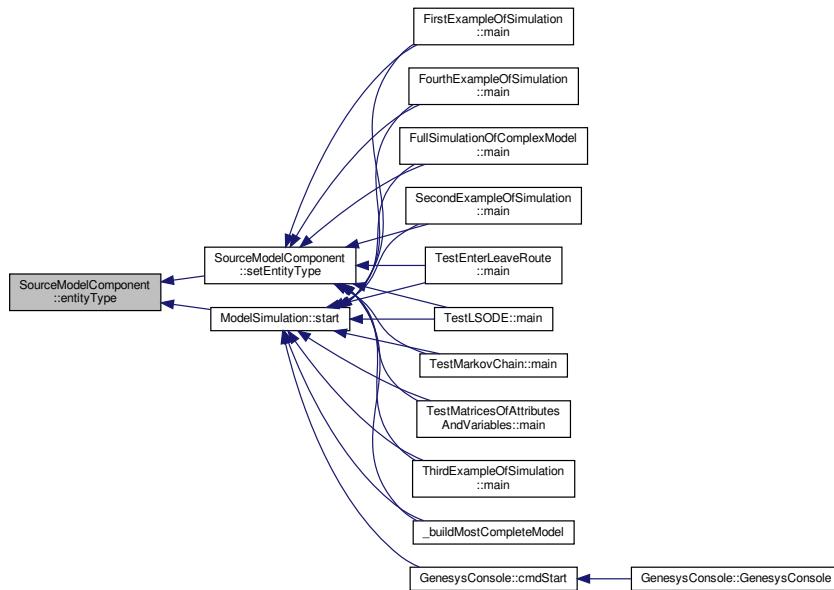


6.126.3.7 entityType()

```
EntityType * SourceModelComponent::entityType ( ) const
```

Definition at line 97 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:

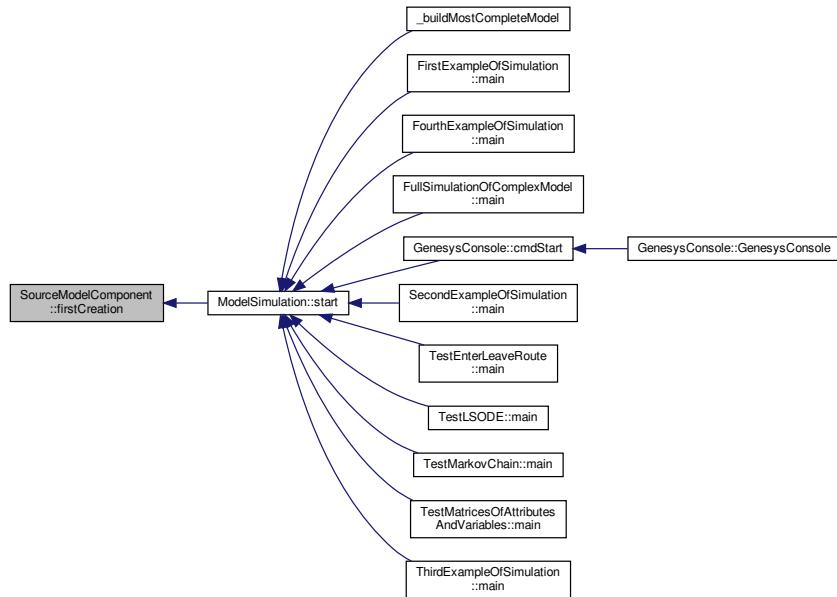


6.126.3.8 firstCreation()

```
double SourceModelComponent::firstCreation ( ) const
```

Definition at line 81 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:



6.126.3.9 isCollectStatistics()

```
bool SourceModelComponent::isCollectStatistics ( ) const
```

6.126.3.10 maxCreations()

```
std::string SourceModelComponent::maxCreations ( ) const
```

Definition at line 121 of file [SourceModelComponent.cpp](#).

6.126.3.11 setCollectStatistics()

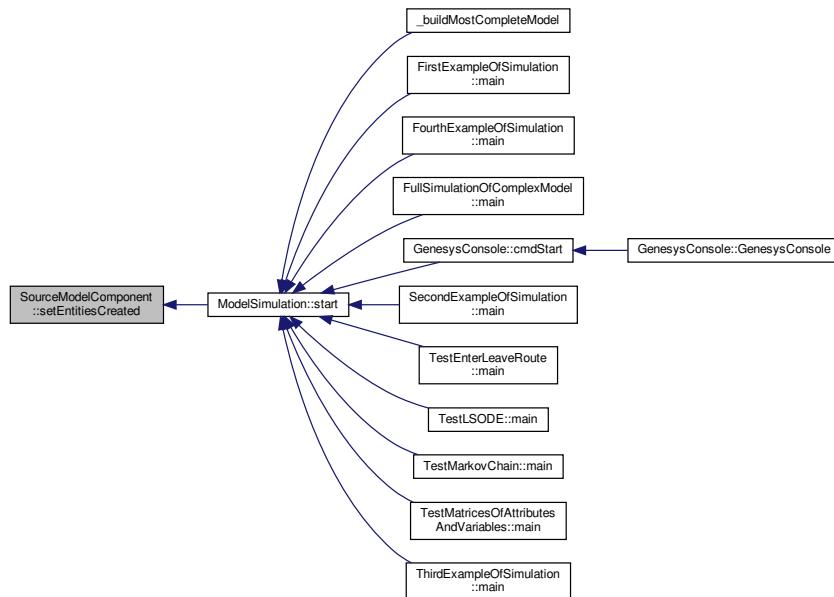
```
void SourceModelComponent::setCollectStatistics (
    bool _collectStatistics )
```

6.126.3.12 setEntitiesCreated()

```
void SourceModelComponent::setEntitiesCreated (
    unsigned int _entitiesCreated )
```

Definition at line 129 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:

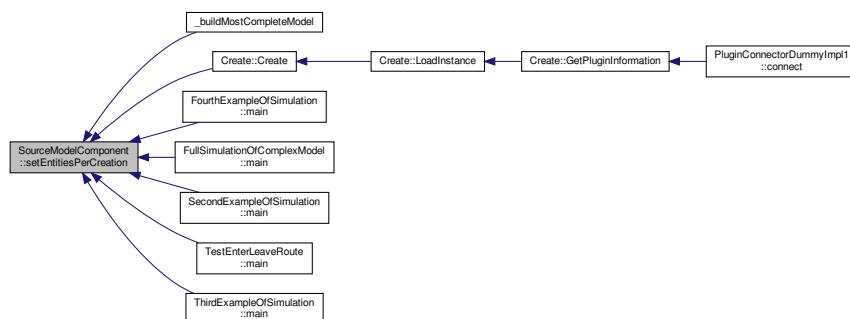


6.126.3.13 setEntitiesPerCreation()

```
void SourceModelComponent::setEntitiesPerCreation (
    unsigned int _entitiesPerCreation )
```

Definition at line 133 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:

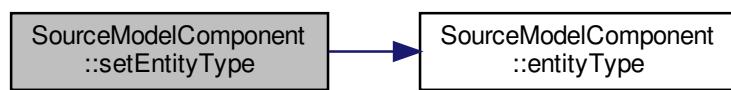


6.126.3.14 setEntityType()

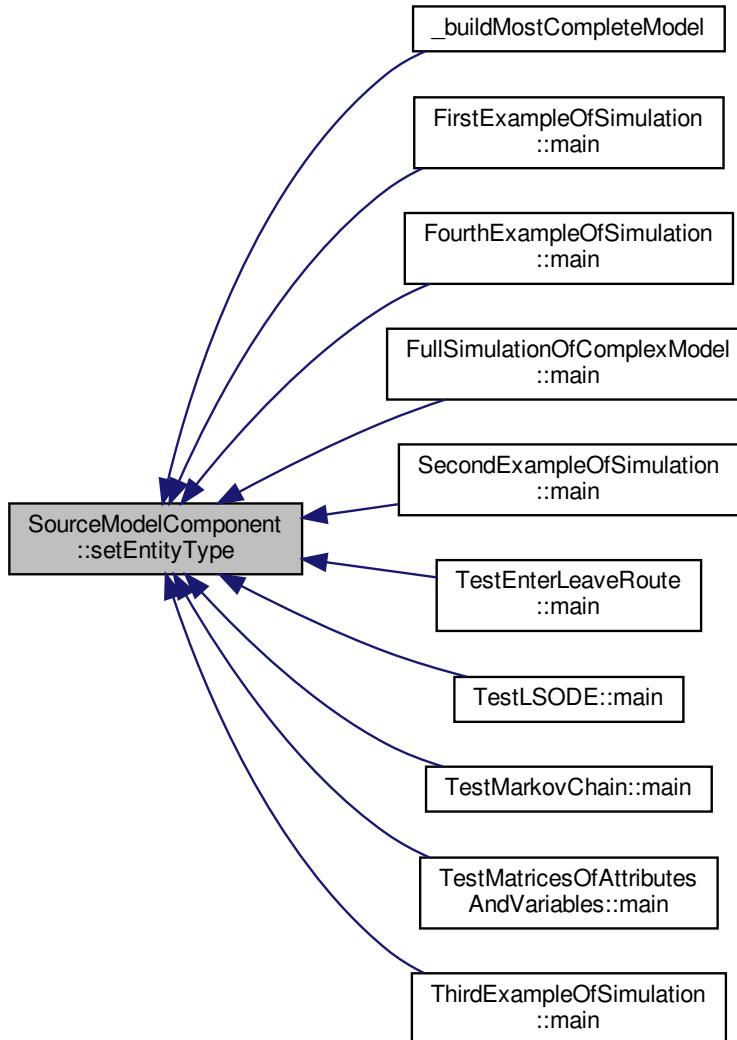
```
void SourceModelComponent::setEntityType (
    EntityType * _entityType )
```

Definition at line 93 of file [SourceModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.126.3.15 setFirstCreation()

```
void SourceModelComponent::setFirstCreation ( double _firstCreation )
```

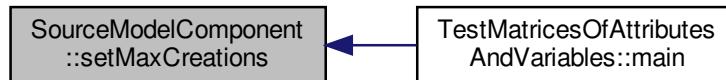
Definition at line 77 of file [SourceModelComponent.cpp](#).

6.126.3.16 setMaxCreations()

```
void SourceModelComponent::setMaxCreations (
    std::string _maxCreationsExpression )
```

Definition at line 117 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:

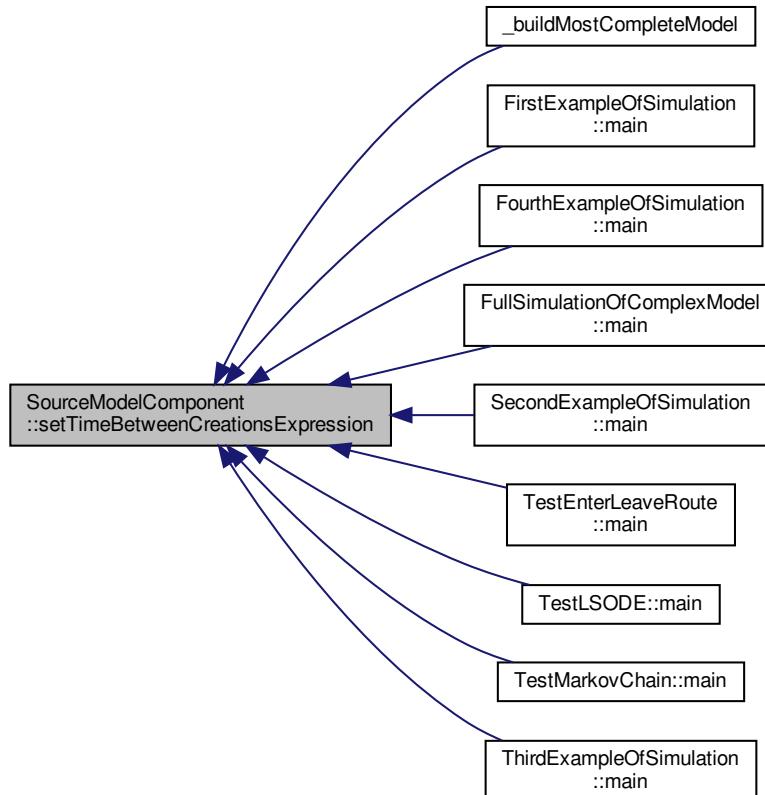


6.126.3.17 setTimeBetweenCreationsExpression()

```
void SourceModelComponent::setTimeBetweenCreationsExpression (
    std::string _timeBetweenCreations )
```

Definition at line 109 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:

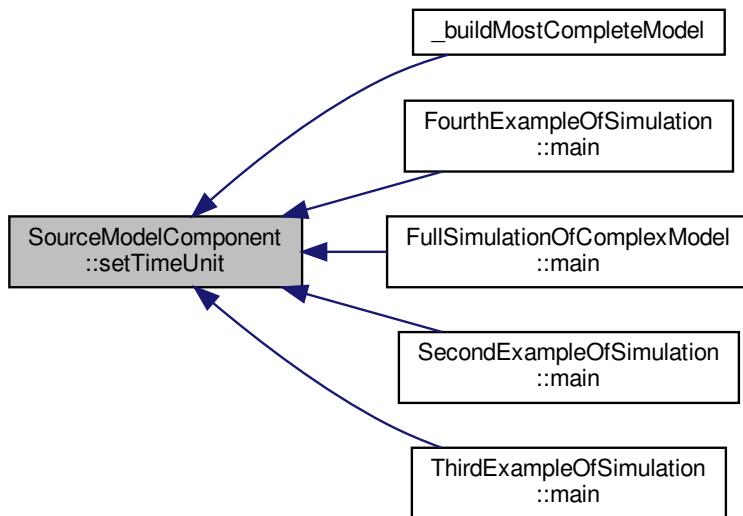


6.126.3.18 setTimeUnit()

```
void SourceModelComponent::setTimeUnit (
    Util::TimeUnit _timeUnit )
```

Definition at line 101 of file [SourceModelComponent.cpp](#).

Here is the caller graph for this function:



6.126.3.19 show()

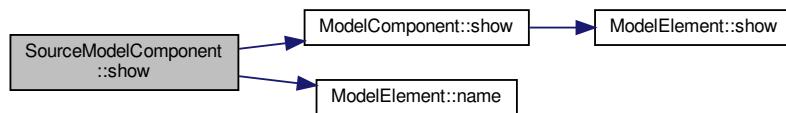
```
std::string SourceModelComponent::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Reimplemented in [Create](#).

Definition at line 22 of file [SourceModelComponent.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.126.3.20 timeBetweenCreationsExpression()

```
std::string SourceModelComponent::timeBetweenCreationsExpression ( ) const
```

Definition at line 113 of file [SourceModelComponent.cpp](#).

6.126.3.21 timeUnit()

```
Util::TimeUnit SourceModelComponent::timeUnit ( ) const
```

Definition at line 105 of file [SourceModelComponent.cpp](#).

6.126.4 Member Data Documentation

6.126.4.1 _entitiesCreatedSoFar

```
unsigned int SourceModelComponent::_entitiesCreatedSoFar = 0 [protected]
```

Definition at line 63 of file [SourceModelComponent.h](#).

6.126.4.2 _entitiesPerCreation

```
unsigned int SourceModelComponent::_entitiesPerCreation = 1 [protected]
```

Definition at line 58 of file [SourceModelComponent.h](#).

6.126.4.3 _entityType

```
EntityType* SourceModelComponent::_entityType [protected]
```

Definition at line 56 of file [SourceModelComponent.h](#).

6.126.4.4 _firstCreation

```
double SourceModelComponent::_firstCreation = 0.0 [protected]
```

Definition at line 57 of file [SourceModelComponent.h](#).

6.126.4.5 _maxCreationsExpression

```
std::string SourceModelComponent::_maxCreationsExpression = std::to_string(std::numeric_limits<unsigned int>::max()) [protected]
```

Definition at line 59 of file [SourceModelComponent.h](#).

6.126.4.6 _timeBetweenCreationsExpression

```
std::string SourceModelComponent::_timeBetweenCreationsExpression = "EXPO(1)" [protected]
```

Definition at line 60 of file [SourceModelComponent.h](#).

6.126.4.7 _timeBetweenCreationsTimeUnit

```
Util::TimeUnit SourceModelComponent::_timeBetweenCreationsTimeUnit = Util::TimeUnit::second [protected]
```

Definition at line 61 of file [SourceModelComponent.h](#).

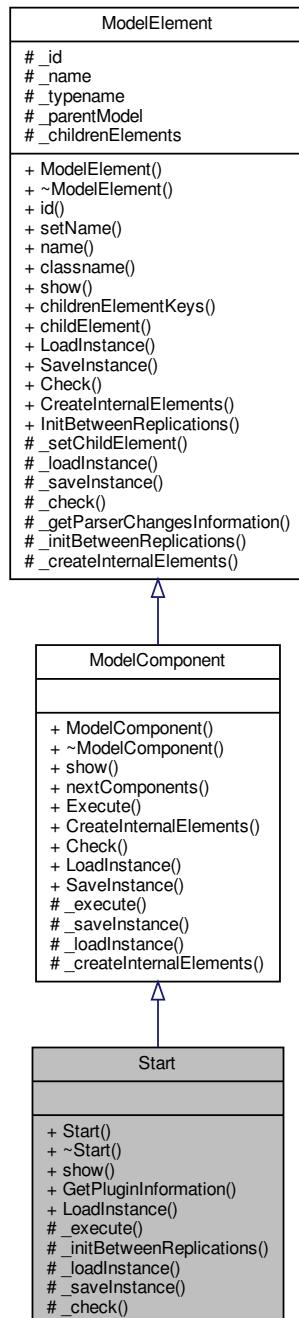
The documentation for this class was generated from the following files:

- [SourceModelComponent.h](#)
- [SourceModelComponent.cpp](#)

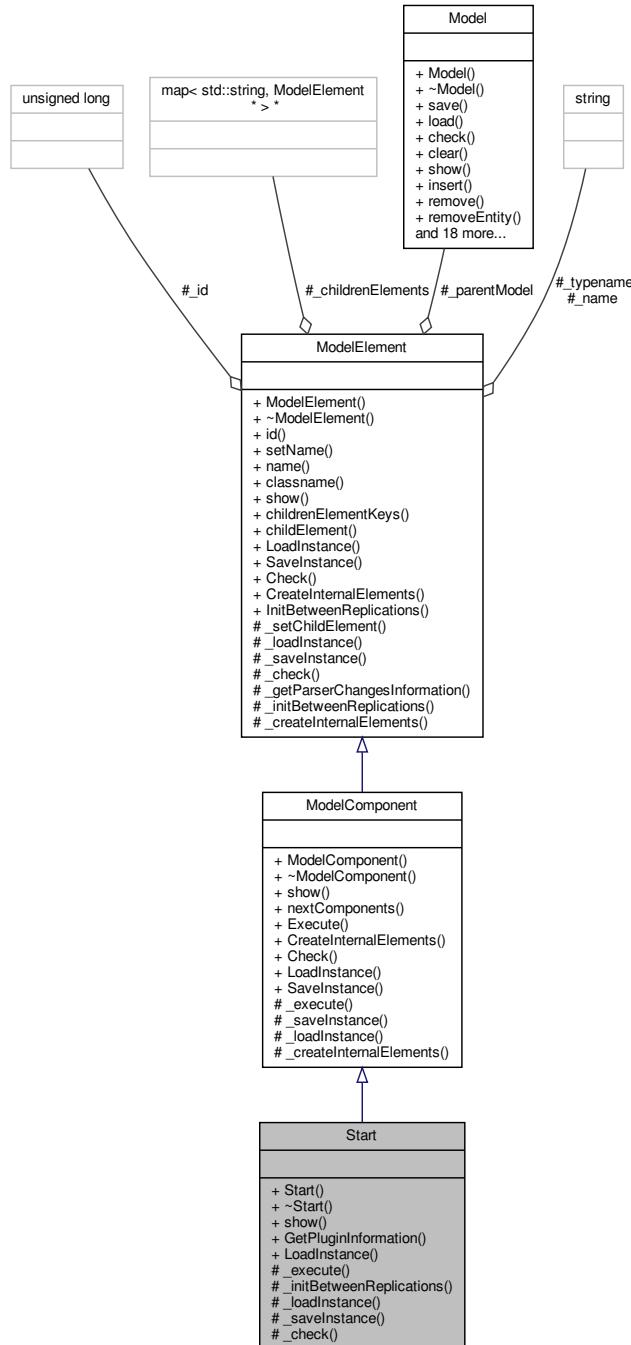
6.127 Start Class Reference

```
#include <Start.h>
```

Inheritance diagram for Start:



Collaboration diagram for Start:



Public Member Functions

- `Start (Model *model, std::string name="")`
- virtual `~Start ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.127.1 Detailed Description

Start module DESCRIPTION The **Start** module changes the status of a conveyor from inactive to active. The conveyor may have been deactivated from either the **Stop** module or by initially being set to inactive at the start of the simulation. The velocity of the conveyor may be changed permanently when the conveyor is started. TYPICAL USES **Start** a bottling conveyor after scheduled maintenance **Start** a baggage claim conveyor when bags have arrived PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor to start. Velocity Speed of the conveyor once it begins to operate. This value will change the speed of the conveyor permanently, until it is changed in another module. Units Velocity time units.

Definition at line 39 of file [Start.h](#).

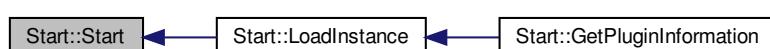
6.127.2 Constructor & Destructor Documentation

6.127.2.1 Start()

```
Start::Start (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Start.cpp](#).

Here is the caller graph for this function:



6.127.2.2 ~Start()

```
virtual Start::~Start ( ) [virtual], [default]
```

6.127.3 Member Function Documentation

6.127.3.1 _check()

```
bool Start::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Start.cpp](#).

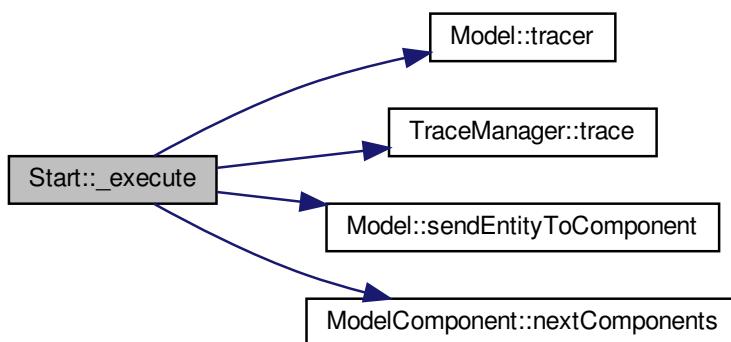
6.127.3.2 _execute()

```
void Start::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Start.cpp](#).

Here is the call graph for this function:



6.127.3.3 `_initBetweenReplications()`

```
void Start::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Start.cpp](#).

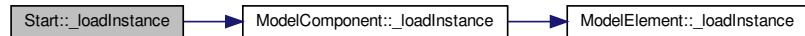
6.127.3.4 `_loadInstance()`

```
bool Start::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

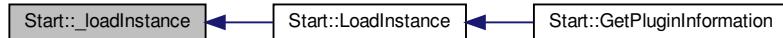
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Start.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



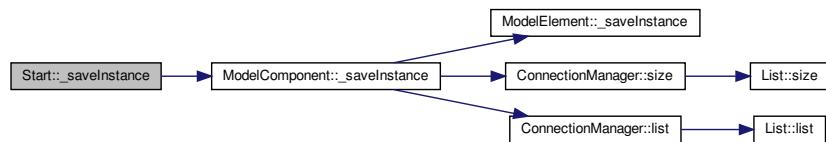
6.127.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Start::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Start.cpp](#).

Here is the call graph for this function:



6.127.3.6 GetPluginInformation()

```
PluginInformation * Start::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Start.cpp](#).

Here is the call graph for this function:

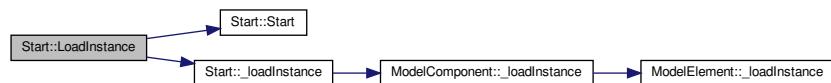


6.127.3.7 LoadInstance()

```
ModelComponent * Start::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Start.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



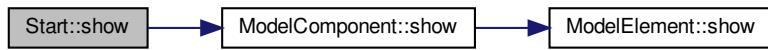
6.127.3.8 show()

```
std::string Start::show () [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Start.cpp](#).

Here is the call graph for this function:



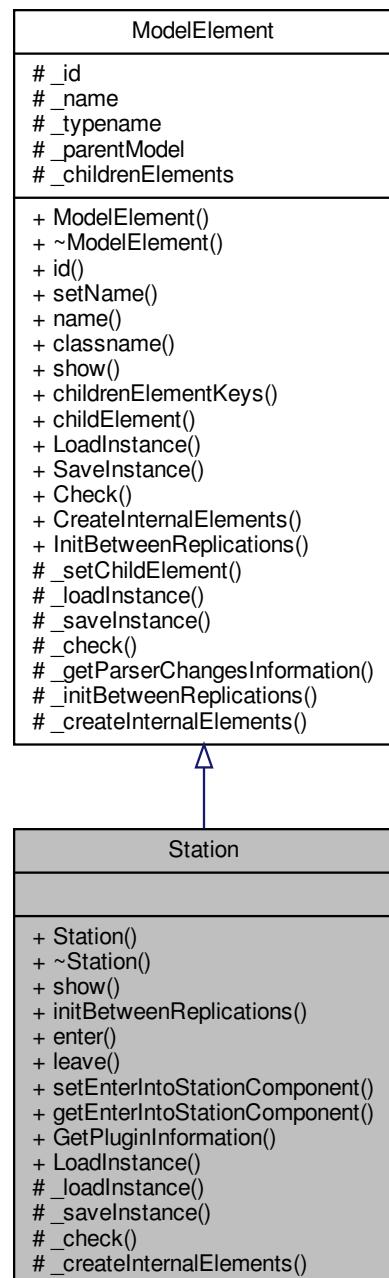
The documentation for this class was generated from the following files:

- [Start.h](#)
- [Start.cpp](#)

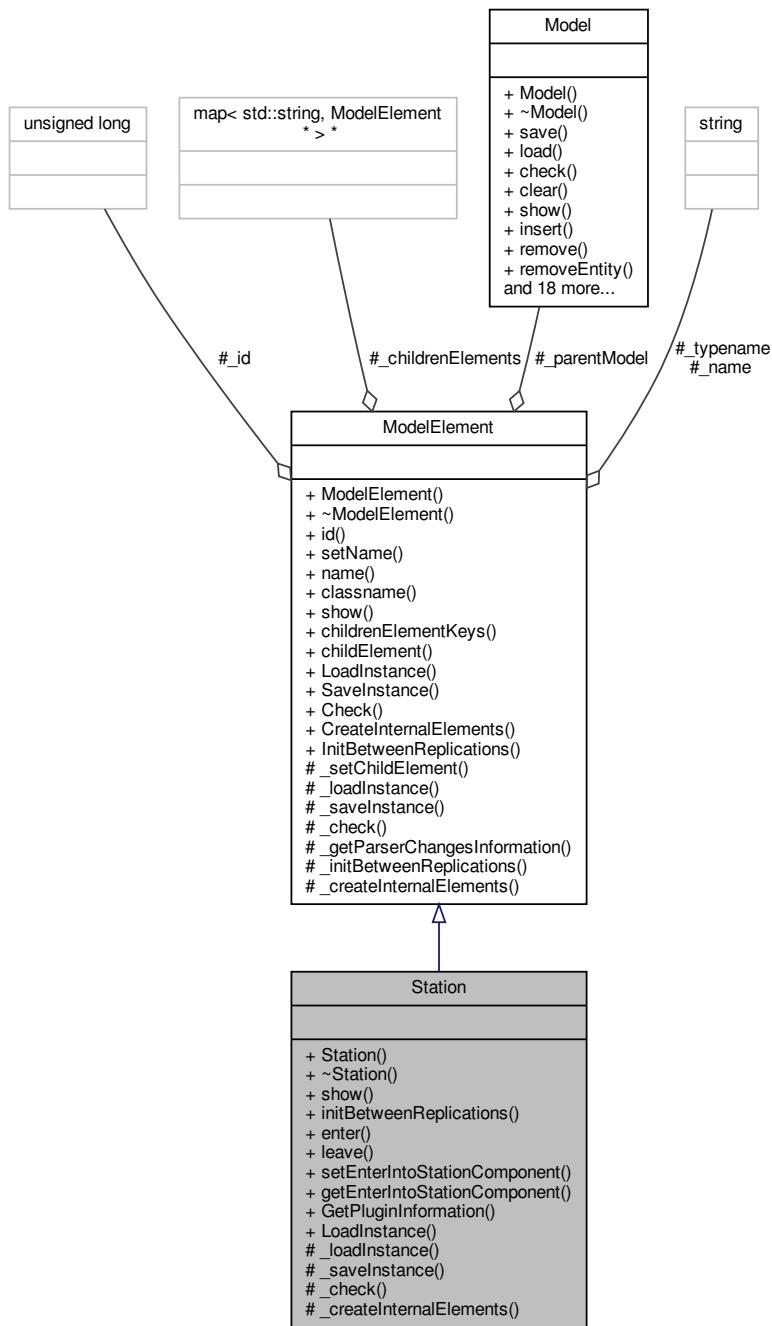
6.128 Station Class Reference

```
#include <Station.h>
```

Inheritance diagram for Station:



Collaboration diagram for Station:



Public Member Functions

- `Station (Model *model, std::string name="")`
- `virtual ~Station ()`
- `virtual std::string show ()`
- `void initBetweenReplications ()`
- `void enter (Entity *entity)`

- void `leave (Entity *entity)`
- void `setEnterIntoStationComponent (ModelComponent *_enterIntoStationComponent)`
- `ModelComponent * getEnterIntoStationComponent () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual `std::map< std::string, std::string > * _saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_createInternalElements ()`

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Additional Inherited Members

6.128.1 Detailed Description

Definition at line 64 of file [Station.h](#).

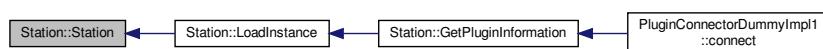
6.128.2 Constructor & Destructor Documentation

6.128.2.1 Station()

```
Station::Station (
    Model * model,
    std::string name = "" )
```

Definition at line 19 of file [Station.cpp](#).

Here is the caller graph for this function:



6.128.2.2 ~Station()

```
Station::~Station ( ) [virtual]
```

Definition at line 30 of file [Station.cpp](#).

6.128.3 Member Function Documentation

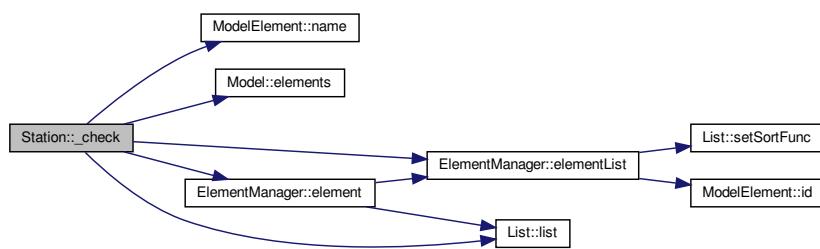
6.128.3.1 _check()

```
bool Station::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 107 of file [Station.cpp](#).

Here is the call graph for this function:



6.128.3.2 _createInternalElements()

```
void Station::_createInternalElements ( ) [protected], [virtual]
```

This method is necessary only for those components that instantiate internal elements that must exist before simulation starts and even before model checking. That's the case of components that have internal StatisticsCollectors, since others components may refer to them as expressions (as in "TVAG(ThisCSTAT)") and therefore the element must exist when checking such expression.

Reimplemented from [ModelElement](#).

Definition at line 128 of file [Station.cpp](#).

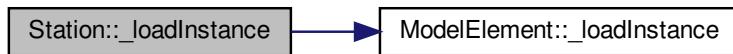
6.128.3.3 `_loadInstance()`

```
bool Station::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [92](#) of file [Station.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



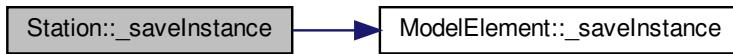
6.128.3.4 `_saveInstance()`

```
std::map< std::string, std::string * > * Station::_saveInstance () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [102](#) of file [Station.cpp](#).

Here is the call graph for this function:

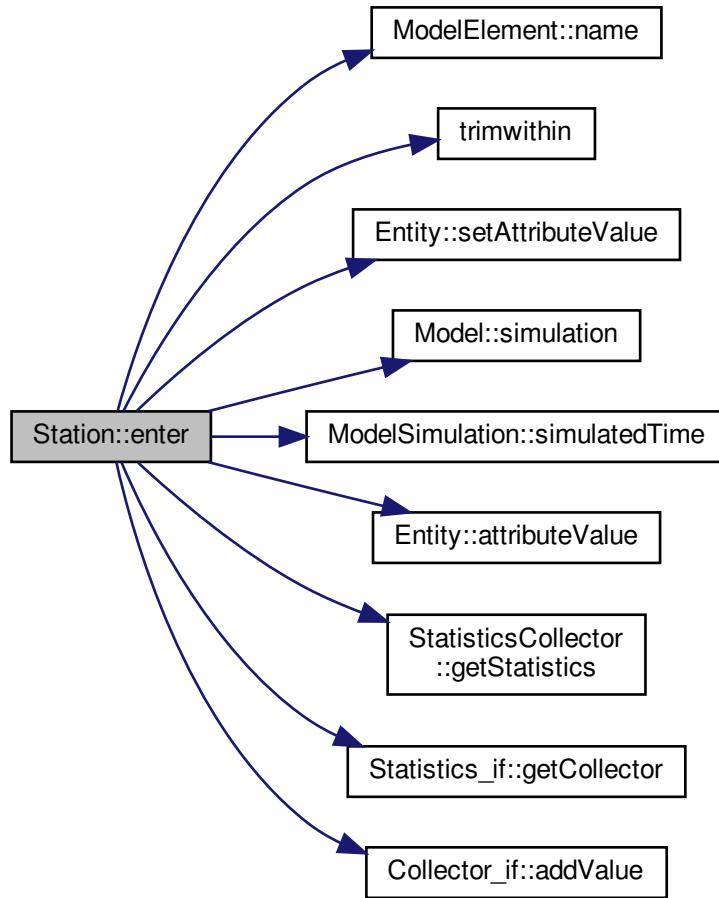


6.128.3.5 enter()

```
void Station::enter (
    Entity * entity )
```

Definition at line 48 of file [Station.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

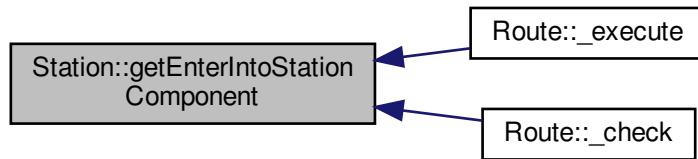


6.128.3.6 getEnterIntoStationComponent()

```
ModelComponent * Station::getEnterIntoStationComponent ( ) const
```

Definition at line 73 of file [Station.cpp](#).

Here is the caller graph for this function:

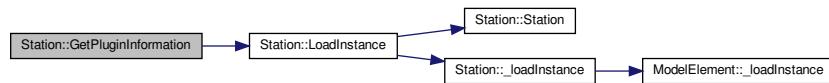


6.128.3.7 GetPluginInformation()

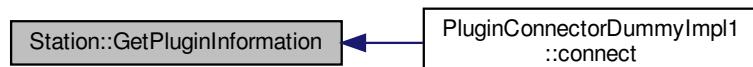
```
PluginInformation * Station::GetPluginInformation ( ) [static]
```

Definition at line 77 of file [Station.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.128.3.8 initBetweenReplications()

```
void Station::initBetweenReplications ( )
```

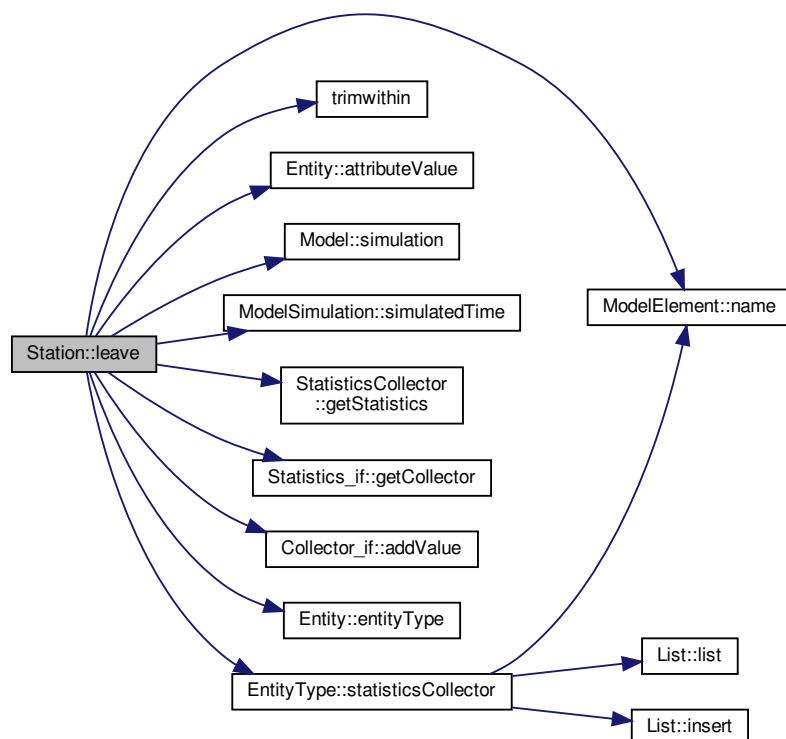
Definition at line 44 of file [Station.cpp](#).

6.128.3.9 leave()

```
void Station::leave ( Entity * entity )
```

Definition at line 57 of file [Station.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

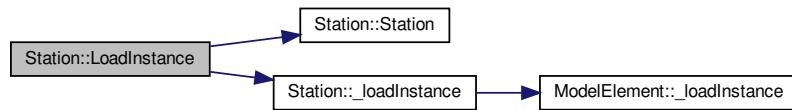


6.128.3.10 LoadInstance()

```
ModelElement * Station::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 82 of file [Station.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

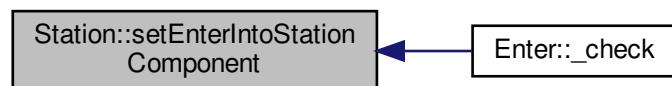


6.128.3.11 setEnterIntoStationComponent()

```
void Station::setEnterIntoStationComponent (
    ModelComponent * _enterIntoStationComponent )
```

Definition at line 69 of file [Station.cpp](#).

Here is the caller graph for this function:



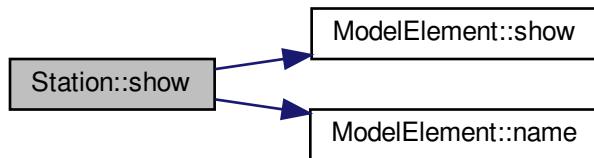
6.128.3.12 show()

```
std::string Station::show () [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 35 of file [Station.cpp](#).

Here is the call graph for this function:



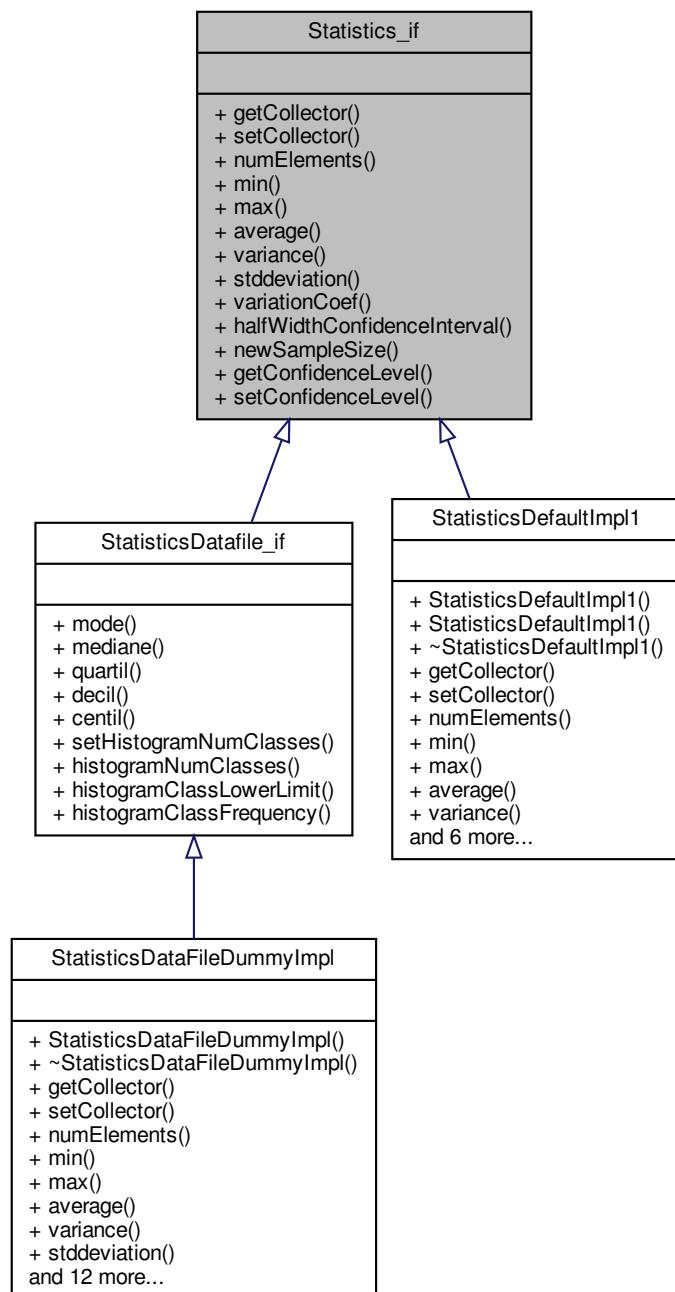
The documentation for this class was generated from the following files:

- [Station.h](#)
- [Station.cpp](#)

6.129 Statistics_if Class Reference

```
#include <Statistics_if.h>
```

Inheritance diagram for Statistics_if:



Collaboration diagram for Statistics_if:

Statistics_if
+ getCollector() + setCollector() + numElements() + min() + max() + average() + variance() + stddeviation() + variationCoef() + halfWidthConfidenceInterval() + newSampleSize() + getConfidenceLevel() + setConfidenceLevel()

Public Member Functions

- virtual `Collector_if * getCollector ()=0`
- virtual void `setCollector (Collector_if *collector)=0`
- virtual unsigned int `numElements ()=0`
- virtual double `min ()=0`
- virtual double `max ()=0`
- virtual double `average ()=0`
- virtual double `variance ()=0`
- virtual double `stddeviation ()=0`
- virtual double `variationCoef ()=0`
- virtual double `halfWidthConfidenceInterval ()=0`
- virtual unsigned int `newSampleSize (double halfWidth)=0`
- virtual double `getConfidenceLevel ()=0`
- virtual void `setConfidenceLevel (double confidencelevel)=0`

6.129.1 Detailed Description

Interface for statistic synthesis of a stochastic variable collected by a `Collector_if`. The statistics generated may be updated based only on the previous statistics and the single newest added value or they may be updated based on a datafile, depending on the Collector implementation.

Definition at line 23 of file `Statistics_if.h`.

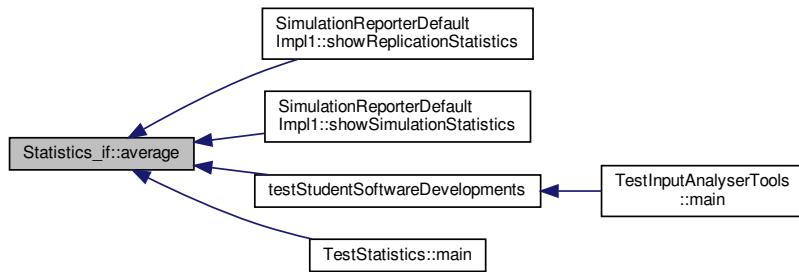
6.129.2 Member Function Documentation

6.129.2.1 average()

```
virtual double Statistics_if::average ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:

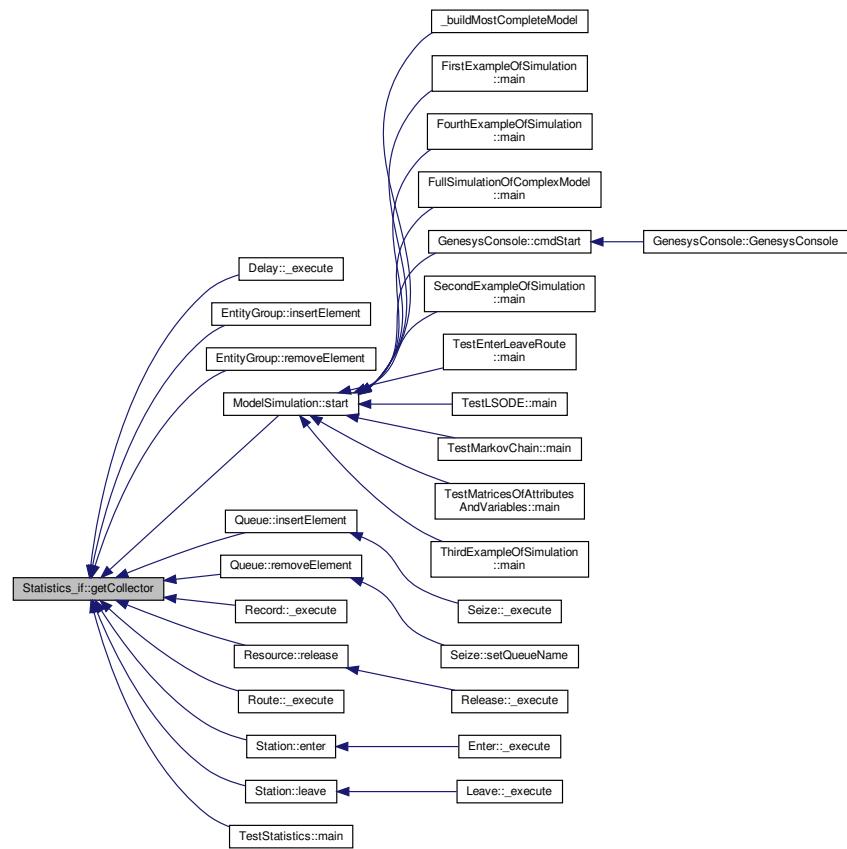


6.129.2.2 getCollector()

```
virtual Collector_if* Statistics_if::getCollector ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:

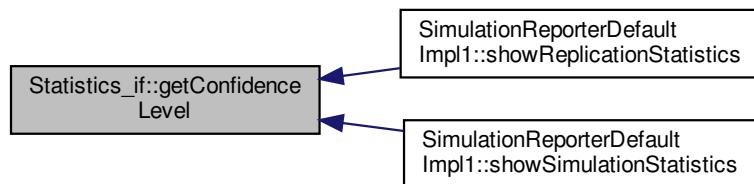


6.129.2.3 getConfidenceLevel()

```
virtual double Statistics_if::getConfidenceLevel ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#).

Here is the caller graph for this function:

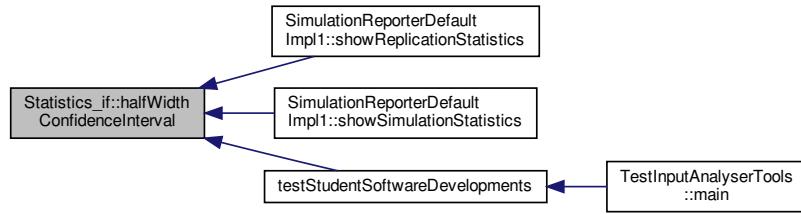


6.129.2.4 halfWidthConfidenceInterval()

```
virtual double Statistics_if::halfWidthConfidenceInterval () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#).

Here is the caller graph for this function:

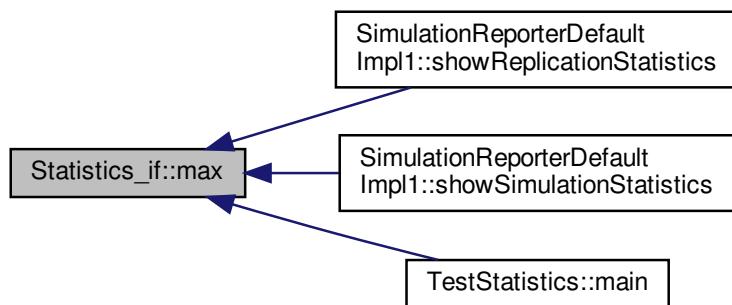


6.129.2.5 max()

```
virtual double Statistics_if::max () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:

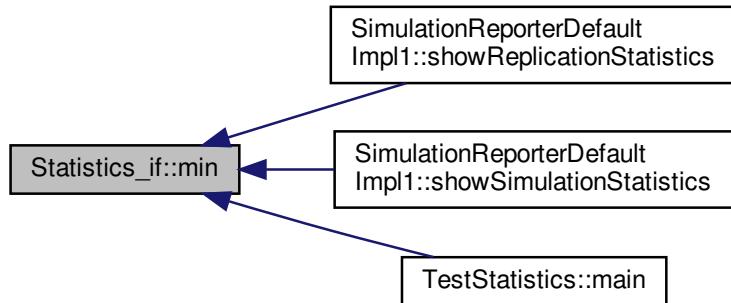


6.129.2.6 min()

```
virtual double Statistics_if::min () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



6.129.2.7 newSampleSize()

```
virtual unsigned int Statistics_if::newSampleSize (
    double halfWidth ) [pure virtual]
```

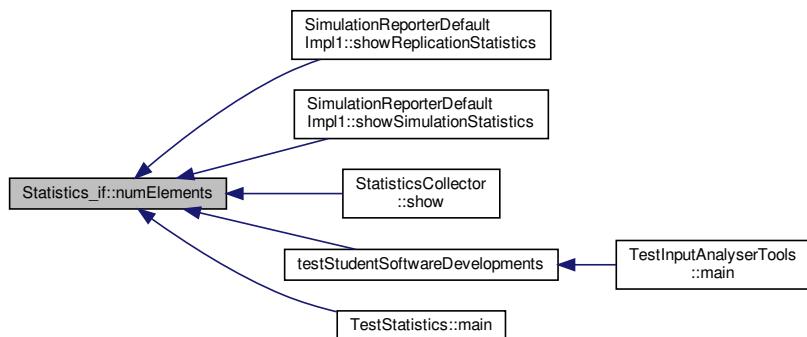
Implemented in [StatisticsDefaultImpl1](#).

6.129.2.8 numElements()

```
virtual unsigned int Statistics_if::numElements () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:

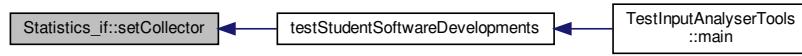


6.129.2.9 setCollector()

```
virtual void Statistics_if::setCollector (
    Collector_if * collector ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



6.129.2.10 setConfidenceLevel()

```
virtual void Statistics_if::setConfidenceLevel (
    double confidencelevel ) [pure virtual]
```

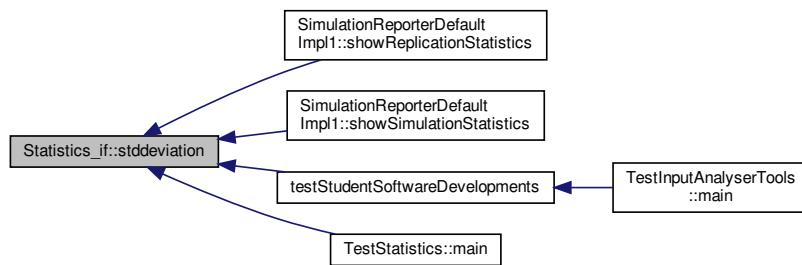
Implemented in [StatisticsDefaultImpl1](#).

6.129.2.11 stddeviation()

```
virtual double Statistics_if::stddeviation () [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:

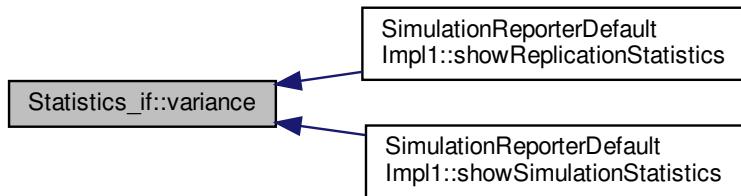


6.129.2.12 variance()

```
virtual double Statistics_if::variance ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:

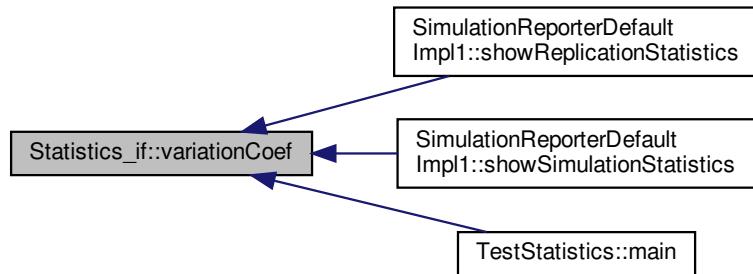


6.129.2.13 variationCoef()

```
virtual double Statistics_if::variationCoef ( ) [pure virtual]
```

Implemented in [StatisticsDefaultImpl1](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



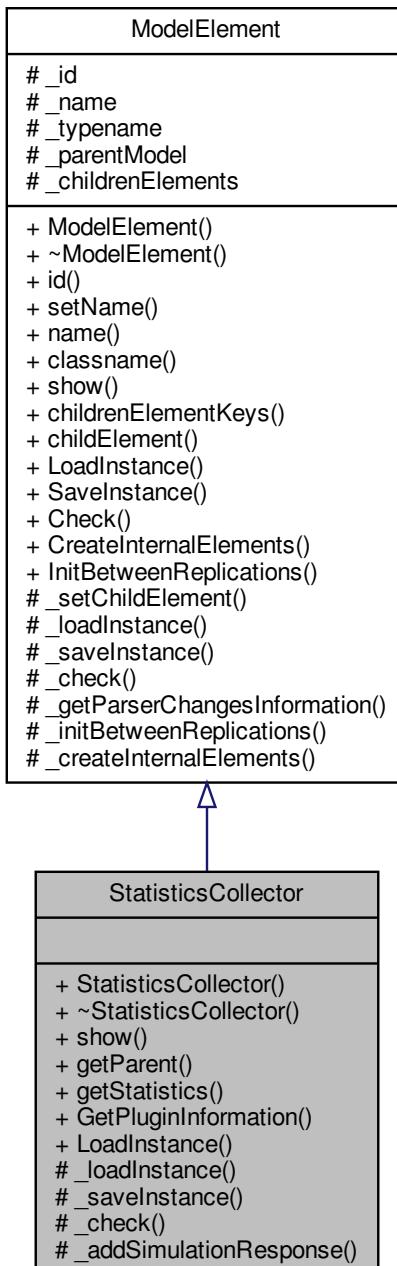
The documentation for this class was generated from the following file:

- [Statistics_if.h](#)

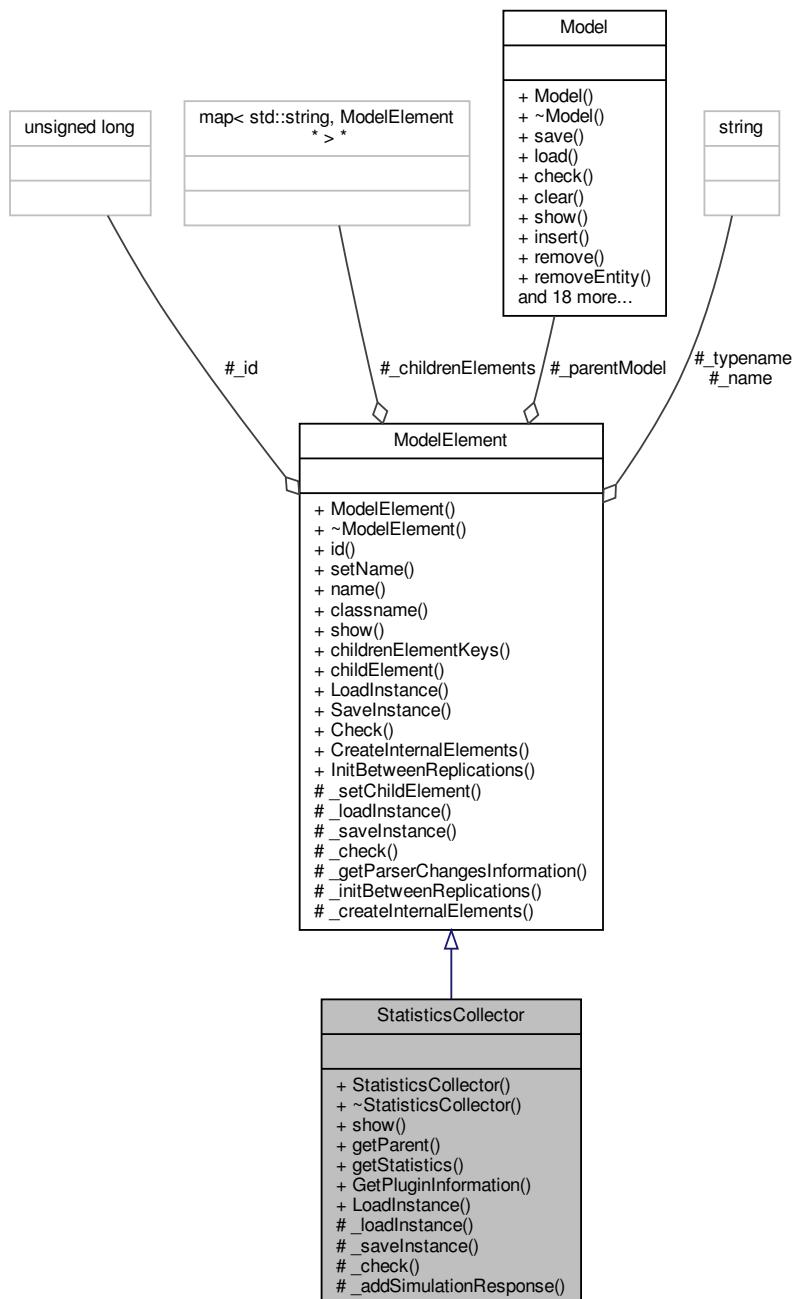
6.130 StatisticsCollector Class Reference

```
#include <StatisticsCollector.h>
```

Inheritance diagram for StatisticsCollector:



Collaboration diagram for StatisticsCollector:



Public Member Functions

- **StatisticsCollector (Model *model, std::string name="", ModelElement *parent=nullptr, bool insertIntoModel=true)**
- virtual **~StatisticsCollector ()=default**
- virtual std::string **show ()**
- **ModelElement * getParent () const**
- **Statistics_if * getStatistics () const**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- void `_addSimulationResponse ()`

Additional Inherited Members

6.130.1 Detailed Description

The `StatisticsCollector` is the `ModelElement` responsible for collecting data from the model (using the Collector) and simultaneously keeping statistics updated (using the Statistics)

Definition at line 25 of file `StatisticsCollector.h`.

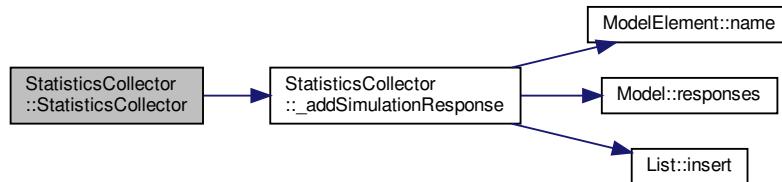
6.130.2 Constructor & Destructor Documentation

6.130.2.1 `StatisticsCollector()`

```
StatisticsCollector::StatisticsCollector (
    Model * model,
    std::string name = "",
    ModelElement * parent = nullptr,
    bool insertIntoModel = true )
```

Definition at line 19 of file `StatisticsCollector.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.130.2.2 ~StatisticsCollector()

```
virtual StatisticsCollector::~StatisticsCollector() [virtual], [default]
```

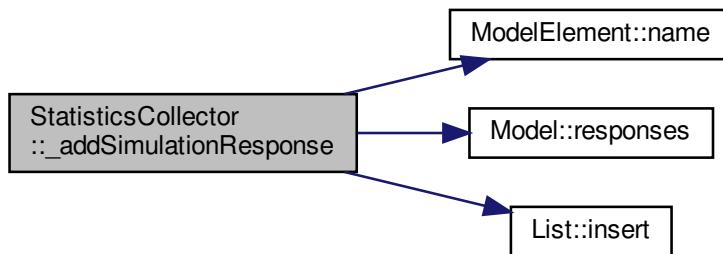
6.130.3 Member Function Documentation

6.130.3.1 _addSimulationResponse()

```
void StatisticsCollector::_addSimulationResponse() [protected]
```

Definition at line 25 of file [StatisticsCollector.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.130.3.2 _check()

```
bool StatisticsCollector::_check(
    std::string * errorMessage) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 98 of file [StatisticsCollector.cpp](#).

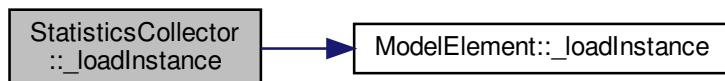
6.130.3.3 `_loadInstance()`

```
bool StatisticsCollector::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 79 of file [StatisticsCollector.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



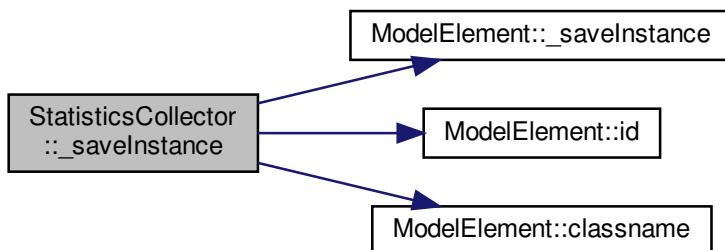
6.130.3.4 `_saveInstance()`

```
std::map< std::string, std::string > * StatisticsCollector::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 86 of file [StatisticsCollector.cpp](#).

Here is the call graph for this function:

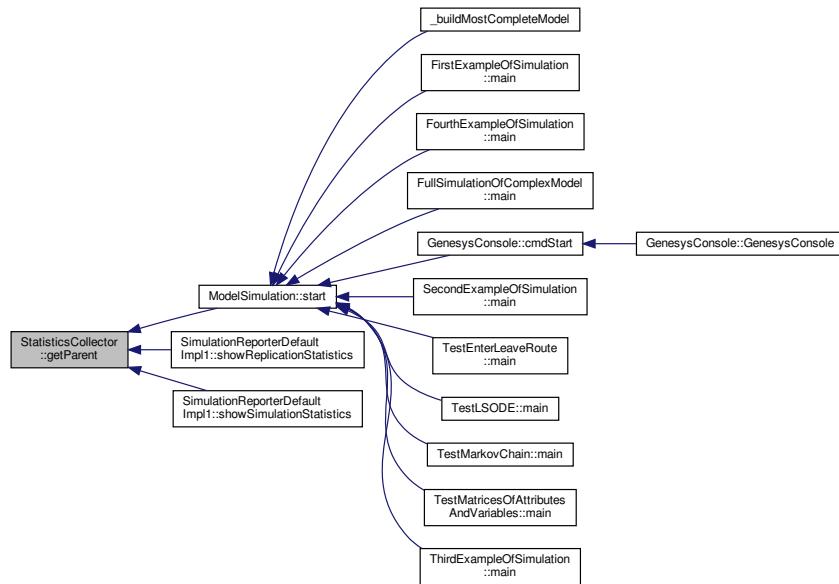


6.130.3.5 getParent()

```
ModelElement * StatisticsCollector::getParent ( ) const
```

Definition at line 55 of file [StatisticsCollector.cpp](#).

Here is the caller graph for this function:

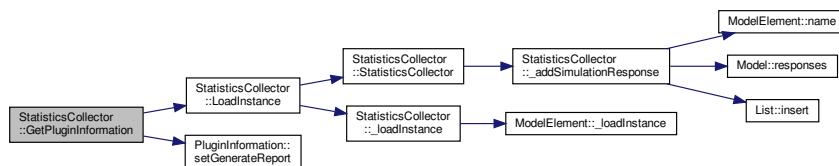


6.130.3.6 GetPluginInformation()

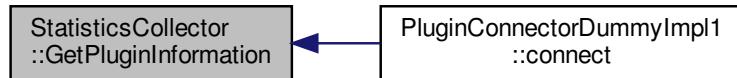
```
PluginInformation * StatisticsCollector::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [StatisticsCollector.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

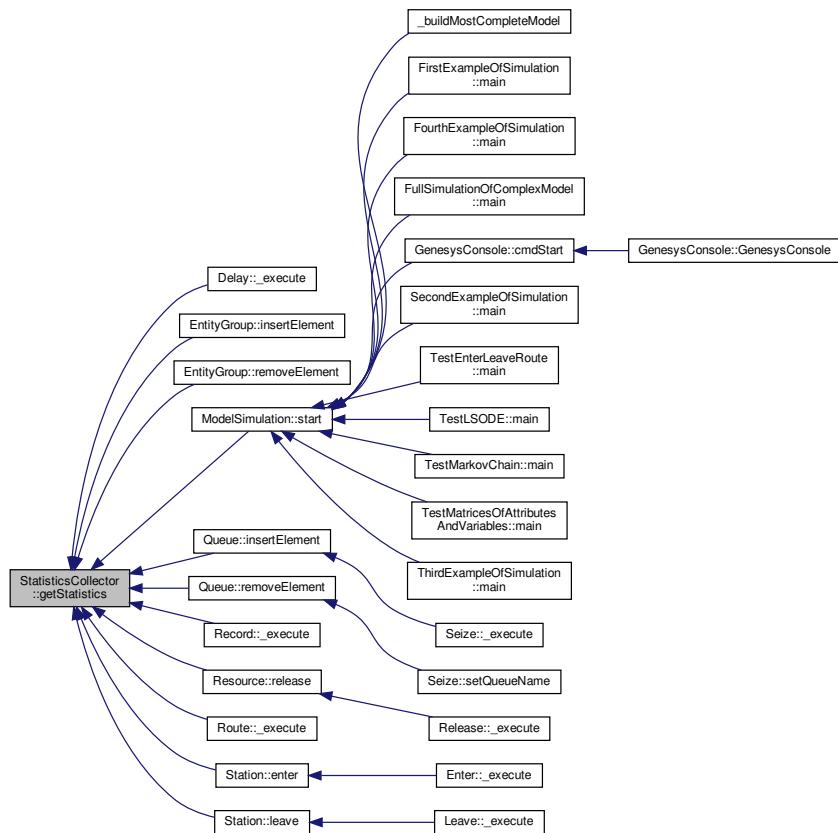


6.130.3.7 getStatistics()

```
Statistics_if * StatisticsCollector::getStatistics ( ) const
```

Definition at line 59 of file [StatisticsCollector.cpp](#).

Here is the caller graph for this function:

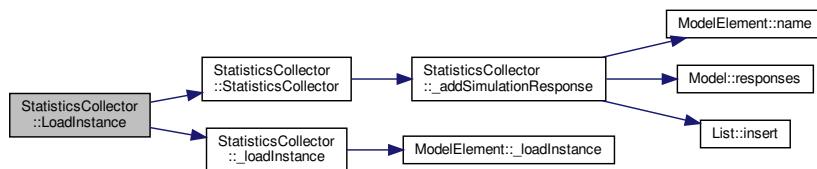


6.130.3.8 LoadInstance()

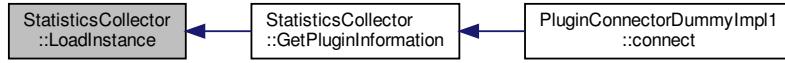
```
ModelElement * StatisticsCollector::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 69 of file [StatisticsCollector.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



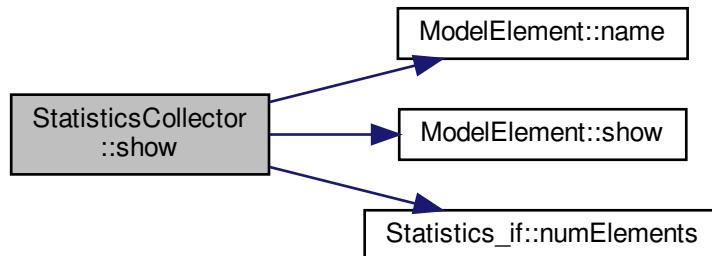
6.130.3.9 show()

```
std::string StatisticsCollector::show () [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 41 of file [StatisticsCollector.cpp](#).

Here is the call graph for this function:



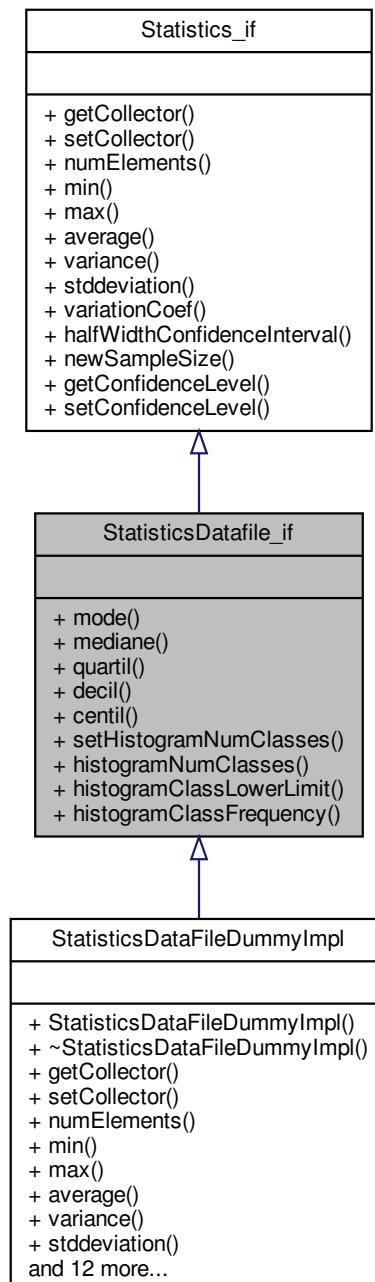
The documentation for this class was generated from the following files:

- [StatisticsCollector.h](#)
- [StatisticsCollector.cpp](#)

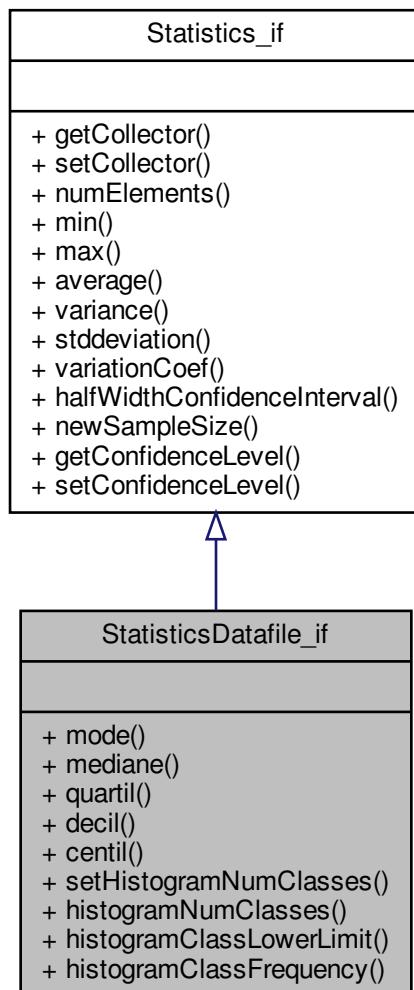
6.131 StatisticsDatafile_if Class Reference

```
#include <StatisticsDataFile_if.h>
```

Inheritance diagram for StatisticsDatafile_if:



Collaboration diagram for StatisticsDatafile_if:



Public Member Functions

- virtual double `mode ()=0`
- virtual double `mediane ()=0`
- virtual double `quartil (unsigned short num)=0`
- virtual double `decil (unsigned short num)=0`
- virtual double `centil (unsigned short num)=0`
- virtual void `setHistogramNumClasses (unsigned short num)=0`
- virtual unsigned short `histogramNumClasses ()=0`
- virtual double `histogramClassLowerLimit (unsigned short classNum)=0`
- virtual unsigned int `histogramClassFrequency (unsigned short classNum)=0`

6.131.1 Detailed Description

Definition at line 20 of file [StatisticsDataFile_if.h](#).

6.131.2 Member Function Documentation

6.131.2.1 centil()

```
virtual double StatisticsDatafile_if::centil (
    unsigned short num ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.2 decil()

```
virtual double StatisticsDatafile_if::decil (
    unsigned short num ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.3 histogramClassFrequency()

```
virtual unsigned int StatisticsDatafile_if::histogramClassFrequency (
    unsigned short classNum ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.4 histogramClassLowerLimit()

```
virtual double StatisticsDatafile_if::histogramClassLowerLimit (
    unsigned short classNum ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.5 histogramNumClasses()

```
virtual unsigned short StatisticsDatafile_if::histogramNumClasses ( ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.6 mediane()

```
virtual double StatisticsDatafile_if::mediane ( ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.7 mode()

```
virtual double StatisticsDatafile_if::mode ( ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.8 quartil()

```
virtual double StatisticsDatafile_if::quartil ( unsigned short num ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

6.131.2.9 setHistogramNumClasses()

```
virtual void StatisticsDatafile_if::setHistogramNumClasses ( unsigned short num ) [pure virtual]
```

Implemented in [StatisticsDataFileDummyImpl](#).

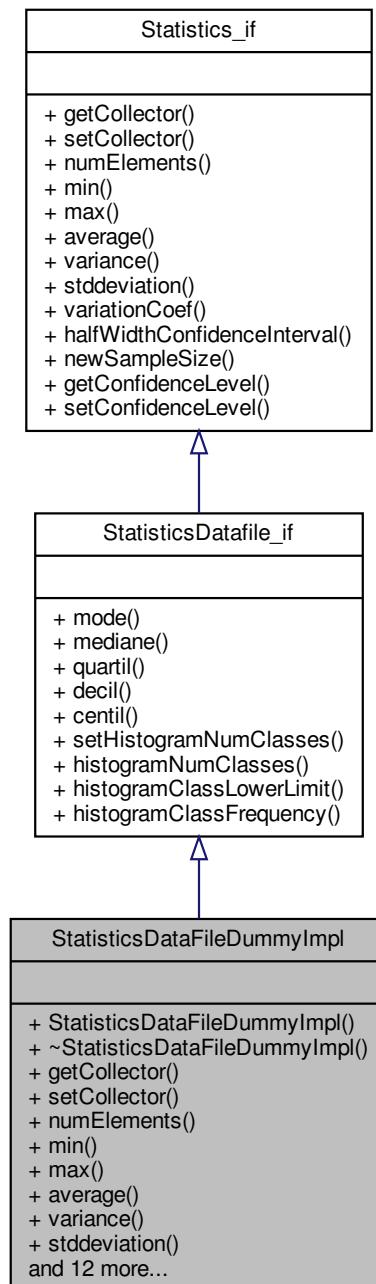
The documentation for this class was generated from the following file:

- [StatisticsDataFile_if.h](#)

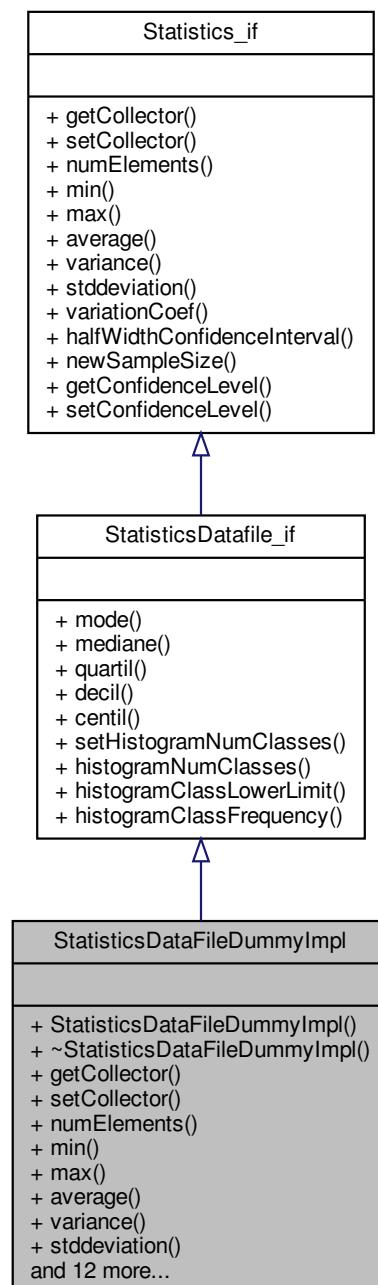
6.132 StatisticsDataFileDummyImpl Class Reference

```
#include <StatisticsDataFileDefaultImpl.h>
```

Inheritance diagram for StatisticsDataFileDummyImpl:



Collaboration diagram for StatisticsDataFileDummyImpl:



Public Member Functions

- `StatisticsDataFileDummyImpl ()`
- virtual `~StatisticsDataFileDummyImpl ()=default`
- virtual `Collector_if * getCollector ()`
- void `setCollector (Collector_if *collector)`
- virtual unsigned int `numElements ()`

- virtual double `min ()`
- virtual double `max ()`
- virtual double `average ()`
- virtual double `variance ()`
- virtual double `stddeviation ()`
- virtual double `variationCoef ()`
- virtual double `halfWidthConfidenceInterval` (double confidencelevel)
- virtual unsigned int `newSampleSize` (double confidencelevel, double halfWidth)
- virtual double `mode ()`
- virtual double `mediane ()`
- virtual double `quartil` (unsigned short num)
- virtual double `decil` (unsigned short num)
- virtual double `centil` (unsigned short num)
- virtual void `setHistogramNumClasses` (unsigned short num)
- virtual unsigned short `histogramNumClasses ()`
- virtual double `histogramClassLowerLimit` (unsigned short classNum)
- virtual unsigned int `histogramClassFrequency` (unsigned short classNum)

6.132.1 Detailed Description

Definition at line 19 of file [StatisticsDataFileDefaultImpl.h](#).

6.132.2 Constructor & Destructor Documentation

6.132.2.1 `StatisticsDataFileDummyImpl()`

```
StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl ( )
```

Definition at line 18 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.2.2 `~StatisticsDataFileDummyImpl()`

```
virtual StatisticsDataFileDummyImpl::~StatisticsDataFileDummyImpl ( ) [virtual], [default]
```

6.132.3 Member Function Documentation

6.132.3.1 `average()`

```
double StatisticsDataFileDummyImpl::average ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 35 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.2 centil()

```
double StatisticsDataFileDummyImpl::centil (
    unsigned short num ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 75 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.3 decil()

```
double StatisticsDataFileDummyImpl::decil (
    unsigned short num ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 71 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.4 getCollector()

```
Collector_if * StatisticsDataFileDummyImpl::getCollector () [virtual]
```

Implements [Statistics_if](#).

Definition at line 94 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.5 halfWidthConfidenceInterval()

```
double StatisticsDataFileDummyImpl::halfWidthConfidenceInterval (
    double confidencelevel ) [virtual]
```

Definition at line 59 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.6 histogramClassFrequency()

```
unsigned int StatisticsDataFileDummyImpl::histogramClassFrequency (
    unsigned short classNum ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 90 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.7 histogramClassLowerLimit()

```
double StatisticsDataFileDummyImpl::histogramClassLowerLimit (
    unsigned short classNum) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 86 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.8 histogramNumClasses()

```
unsigned short StatisticsDataFileDummyImpl::histogramNumClasses () [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 82 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.9 max()

```
double StatisticsDataFileDummyImpl::max () [virtual]
```

Implements [Statistics_if](#).

Definition at line 31 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.10 mediane()

```
double StatisticsDataFileDummyImpl::mediane () [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 43 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.11 min()

```
double StatisticsDataFileDummyImpl::min () [virtual]
```

Implements [Statistics_if](#).

Definition at line 27 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.12 mode()

```
double StatisticsDataFileDummyImpl::mode ( ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 39 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.13 newSampleSize()

```
unsigned int StatisticsDataFileDummyImpl::newSampleSize ( 
    double confidencelevel,
    double halfWidth ) [virtual]
```

Definition at line 63 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.14 numElements()

```
unsigned int StatisticsDataFileDummyImpl::numElements ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 23 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.15 quartil()

```
double StatisticsDataFileDummyImpl::quartil ( 
    unsigned short num ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 67 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.16 setCollector()

```
void StatisticsDataFileDummyImpl::setCollector ( 
    Collector_if * collector ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 98 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.17 setHistogramNumClasses()

```
void StatisticsDataFileDummyImpl::setHistogramNumClasses ( unsigned short num ) [virtual]
```

Implements [StatisticsDatafile_if](#).

Definition at line 79 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.18 stddeviation()

```
double StatisticsDataFileDummyImpl::stddeviation ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 51 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.19 variance()

```
double StatisticsDataFileDummyImpl::variance ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 47 of file [StatisticsDataFileDefaultImpl.cpp](#).

6.132.3.20 variationCoef()

```
double StatisticsDataFileDummyImpl::variationCoef ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 55 of file [StatisticsDataFileDefaultImpl.cpp](#).

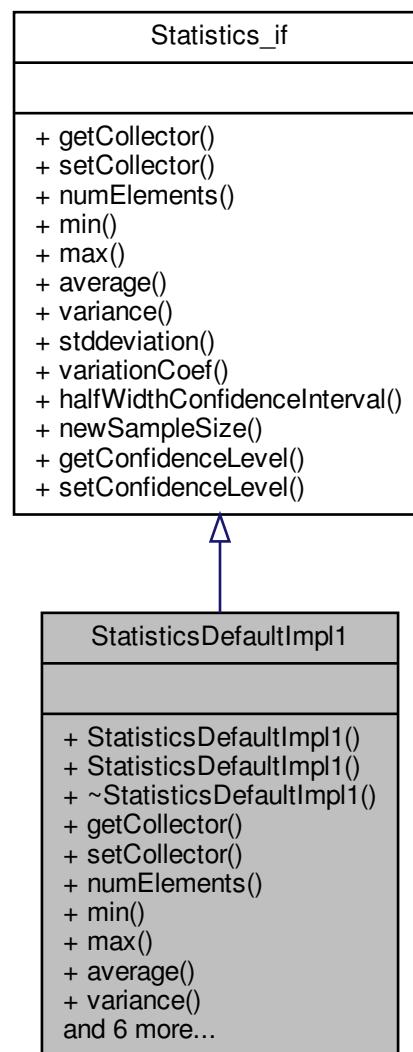
The documentation for this class was generated from the following files:

- [StatisticsDataFileDefaultImpl.h](#)
- [StatisticsDataFileDefaultImpl.cpp](#)

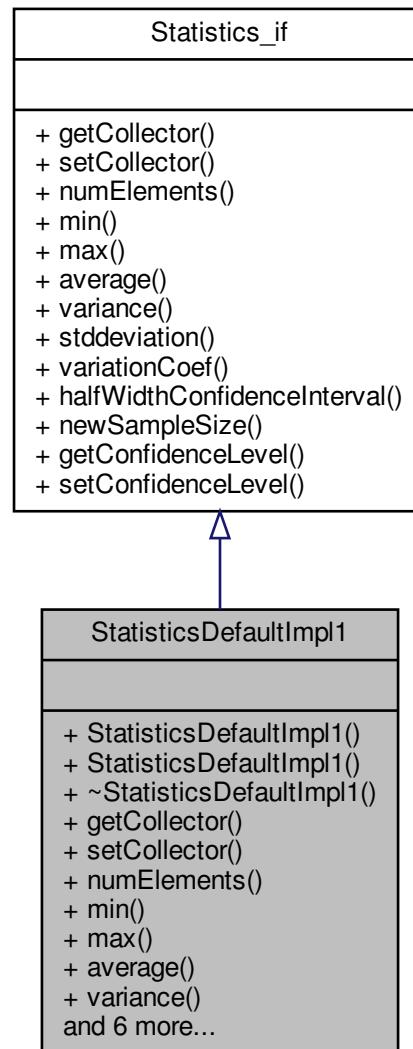
6.133 StatisticsDefaultImpl1 Class Reference

```
#include <StatisticsDefaultImpl1.h>
```

Inheritance diagram for StatisticsDefaultImpl1:



Collaboration diagram for StatisticsDefaultImpl1:



Public Member Functions

- `StatisticsDefaultImpl1 ()`
When constructor is invoked without a Collector, it is taken from `Traits<Statistics_if>::CollectorImplementation` configuration.
- `StatisticsDefaultImpl1 (Collector_if *collector)`
- `virtual ~StatisticsDefaultImpl1 ()=default`
- `virtual Collector_if * getCollector ()`
- `virtual void setCollector (Collector_if *collector)`
- `virtual unsigned int numElements ()`
- `virtual double min ()`
- `virtual double max ()`
- `virtual double average ()`

- virtual double `variance ()`
- virtual double `stddeviation ()`
- virtual double `variationCoef ()`
- virtual double `halfWidthConfidenceInterval ()`
- virtual unsigned int `newSampleSize (double halfWidth)`
- virtual double `getConfidenceLevel ()`
- virtual void `setConfidenceLevel (double confidencelevel)`

6.133.1 Detailed Description

Definition at line 20 of file [StatisticsDefaultImpl1.h](#).

6.133.2 Constructor & Destructor Documentation

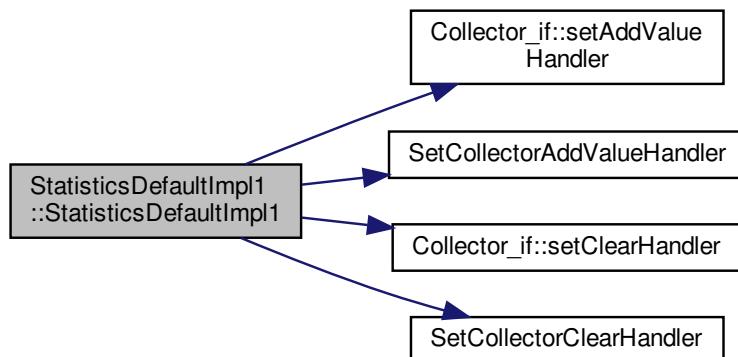
6.133.2.1 StatisticsDefaultImpl1() [1/2]

```
StatisticsDefaultImpl1::StatisticsDefaultImpl1 ( )
```

When constructor is invoked without a Collector, it is taken from `Traits<Statistics_if>::CollectorImplementation` configuration.

Definition at line 21 of file [StatisticsDefaultImpl1.cpp](#).

Here is the call graph for this function:

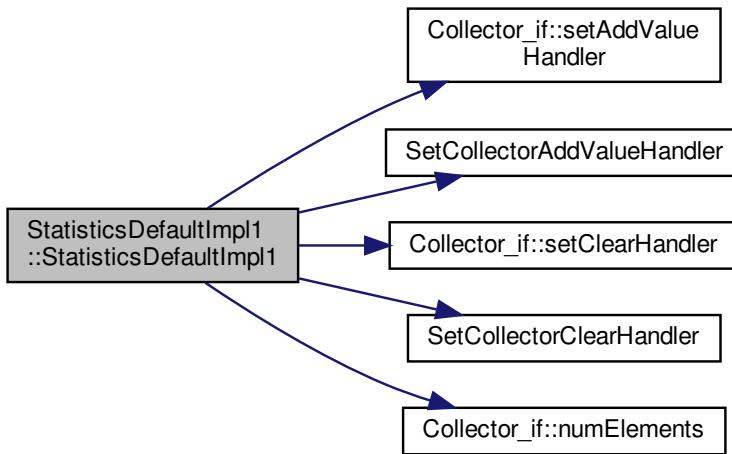


6.133.2.2 StatisticsDefaultImpl1() [2/2]

```
StatisticsDefaultImpl1::StatisticsDefaultImpl1 (
    Collector_if * collector )
```

Definition at line 29 of file [StatisticsDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.133.2.3 ~StatisticsDefaultImpl1()

```
virtual StatisticsDefaultImpl1::~StatisticsDefaultImpl1 ( ) [virtual], [default]
```

6.133.3 Member Function Documentation

6.133.3.1 average()

```
double StatisticsDefaultImpl1::average ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 92 of file [StatisticsDefaultImpl1.cpp](#).

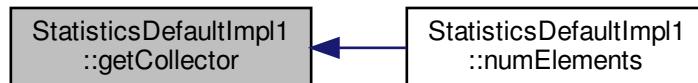
6.133.3.2 getCollector()

```
Collector_if * StatisticsDefaultImpl1::getCollector () [virtual]
```

Implements [Statistics_if](#).

Definition at line 127 of file [StatisticsDefaultImpl1.cpp](#).

Here is the caller graph for this function:



6.133.3.3 getConfidenceLevel()

```
double StatisticsDefaultImpl1::getConfidenceLevel () [virtual]
```

Implements [Statistics_if](#).

Definition at line 119 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.4 halfWidthConfidenceInterval()

```
double StatisticsDefaultImpl1::halfWidthConfidenceInterval () [virtual]
```

Implements [Statistics_if](#).

Definition at line 108 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.5 max()

```
double StatisticsDefaultImpl1::max () [virtual]
```

Implements [Statistics_if](#).

Definition at line 85 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.6 min()

```
double StatisticsDefaultImpl1::min ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 78 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.7 newSampleSize()

```
unsigned int StatisticsDefaultImpl1::newSampleSize (
    double halfWidth ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 123 of file [StatisticsDefaultImpl1.cpp](#).

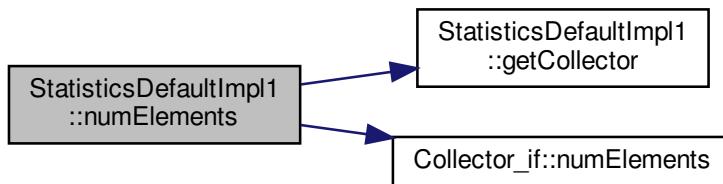
6.133.3.8 numElements()

```
unsigned int StatisticsDefaultImpl1::numElements ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 74 of file [StatisticsDefaultImpl1.cpp](#).

Here is the call graph for this function:



6.133.3.9 setCollector()

```
void StatisticsDefaultImpl1::setCollector (
    Collector_if * collector ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 131 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.10 setConfidenceLevel()

```
void StatisticsDefaultImpl1::setConfidenceLevel ( double confidencelevel ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 112 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.11 stddeviation()

```
double StatisticsDefaultImpl1::stddeviation ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 100 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.12 variance()

```
double StatisticsDefaultImpl1::variance ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 96 of file [StatisticsDefaultImpl1.cpp](#).

6.133.3.13 variationCoef()

```
double StatisticsDefaultImpl1::variationCoef ( ) [virtual]
```

Implements [Statistics_if](#).

Definition at line 104 of file [StatisticsDefaultImpl1.cpp](#).

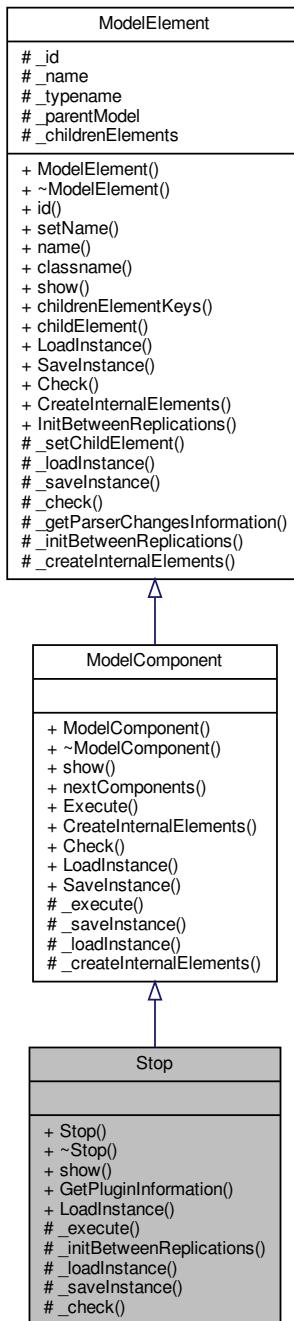
The documentation for this class was generated from the following files:

- [StatisticsDefaultImpl1.h](#)
- [StatisticsDefaultImpl1.cpp](#)

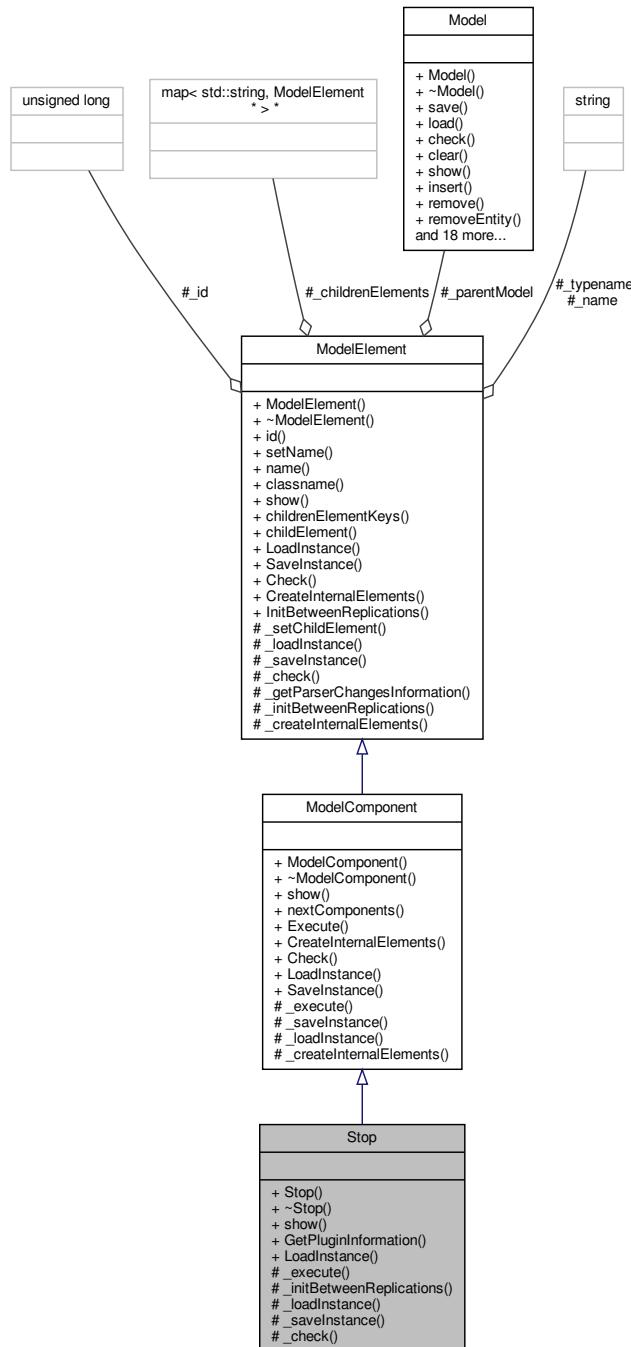
6.134 Stop Class Reference

```
#include <Stop.h>
```

Inheritance diagram for Stop:



Collaboration diagram for Stop:



Public Member Functions

- `Stop (Model *model, std::string name="")`
- virtual `~Stop ()=default`
- virtual std::string `show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.134.1 Detailed Description

Stop module DESCRIPTION The **Stop** module sets the operational status of a conveyor to inactive. The conveyor may have been activated from either the **Start** module or by initially being set to active at the start of the simulation. When the entity enters the **Stop** module, the conveyor will stop immediately, regardless of the type of conveyor or the number of entities currently on the conveyor. TYPICAL USES **Stop** a baggage conveyor after a pre-determined amount of time **Stop** a conveyor for scheduled maintenance PROMPTS Prompt Description Name Unique name of the module that will be displayed in the flowchart. Conveyor Name Name of the conveyor to stop.

Definition at line 36 of file **Stop.h**.

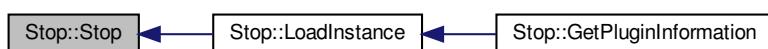
6.134.2 Constructor & Destructor Documentation

6.134.2.1 Stop()

```
Stop::Stop (
    Model * model,
    std::string name = "")
```

Definition at line 18 of file **Stop.cpp**.

Here is the caller graph for this function:



6.134.2.2 ~Stop()

```
virtual Stop::~Stop ( ) [virtual], [default]
```

6.134.3 Member Function Documentation

6.134.3.1 _check()

```
bool Stop::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Stop.cpp](#).

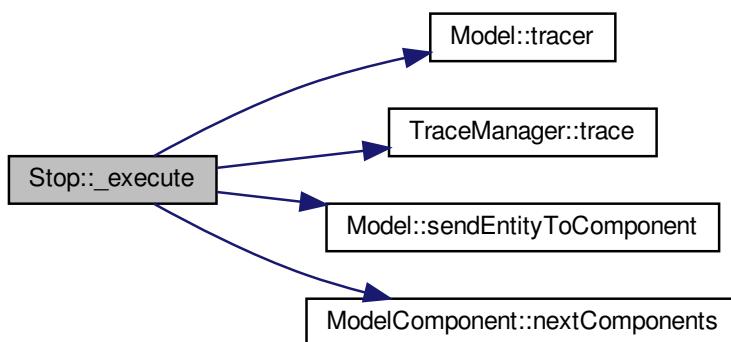
6.134.3.2 _execute()

```
void Stop::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Stop.cpp](#).

Here is the call graph for this function:



6.134.3.3 `_initBetweenReplications()`

```
void Stop::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Stop.cpp](#).

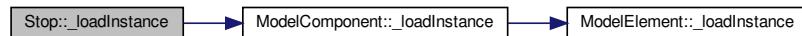
6.134.3.4 `_loadInstance()`

```
bool Stop::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

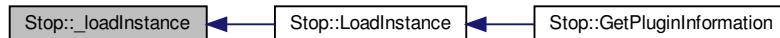
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Stop.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



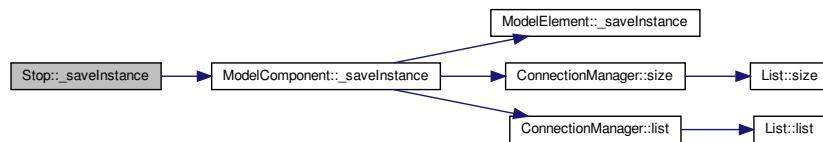
6.134.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Stop::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Stop.cpp](#).

Here is the call graph for this function:



6.134.3.6 GetPluginInformation()

```
PluginInformation * Stop::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Stop.cpp](#).

Here is the call graph for this function:



6.134.3.7 LoadInstance()

```
ModelComponent * Stop::LoadInstance ( Model * model,  
std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Stop.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



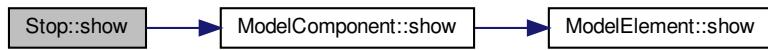
6.134.3.8 show()

```
std::string Stop::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [Stop.cpp](#).

Here is the call graph for this function:



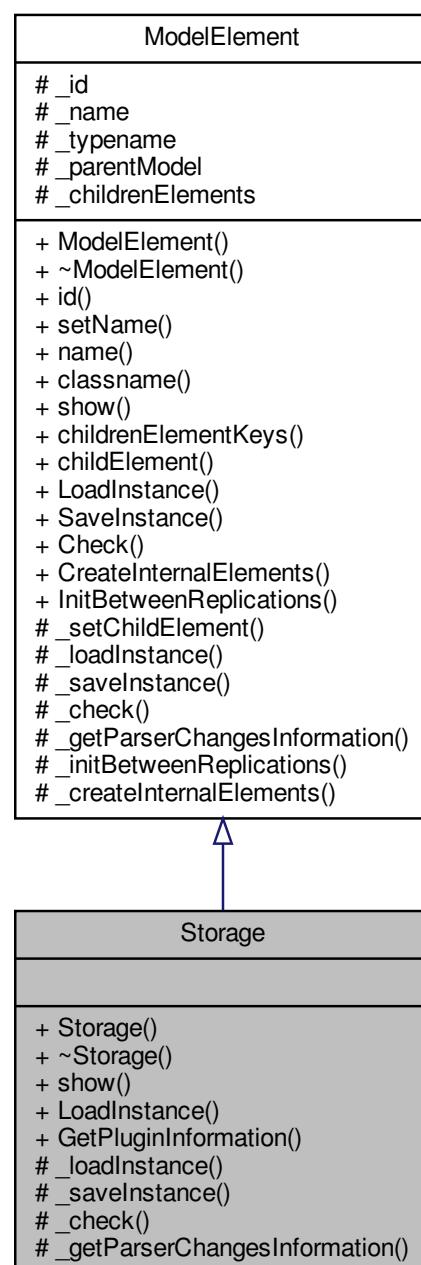
The documentation for this class was generated from the following files:

- [Stop.h](#)
- [Stop.cpp](#)

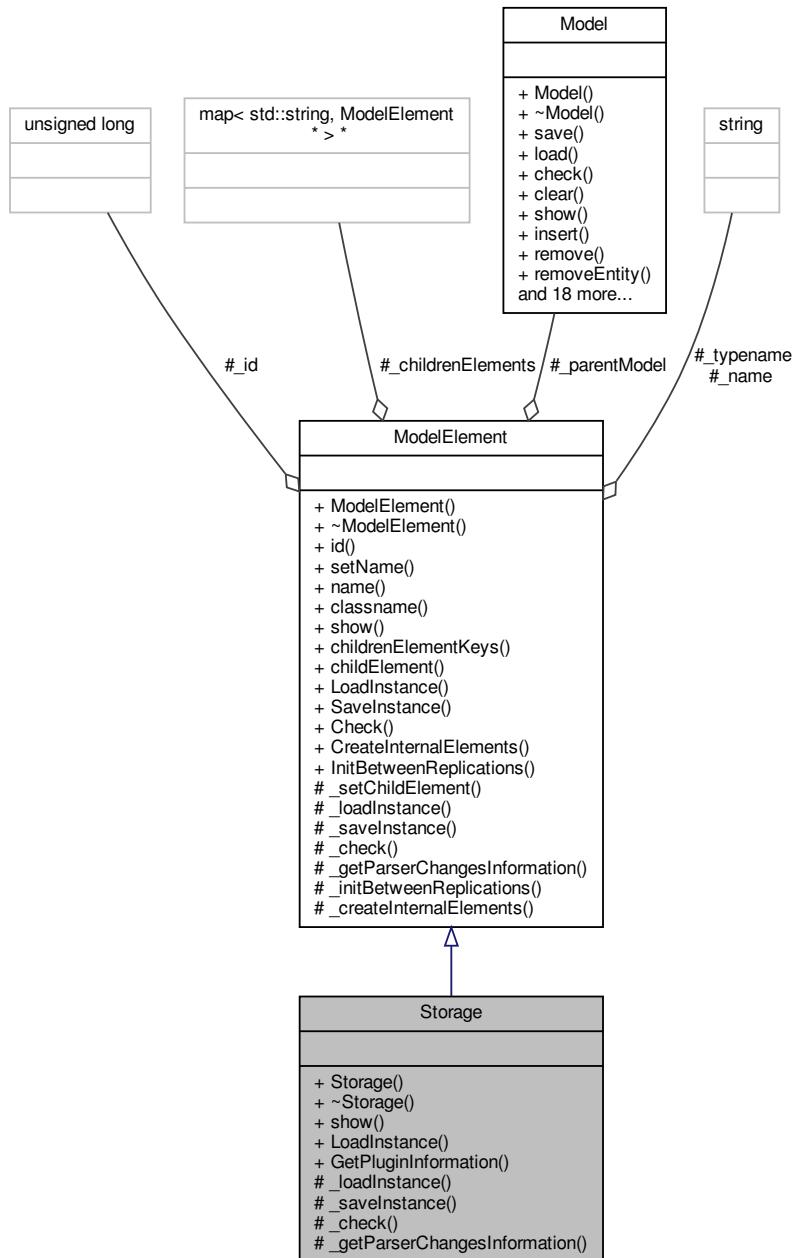
6.135 Storage Class Reference

```
#include <Storage.h>
```

Inheritance diagram for Storage:



Collaboration diagram for Storage:



Public Member Functions

- `Storage (Model *model, std::string name="")`
- virtual `~Storage ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`
- static `PluginInformation * GetPluginInformation ()`

Protected Member Functions

- virtual bool [_loadInstance](#) (std::map< std::string, std::string > *fields)
- virtual std::map< std::string, std::string > * [_saveInstance](#) ()
- virtual bool [_check](#) (std::string *errorMessage)
- virtual [ParserChangesInformation](#) * [_getParserChangesInformation](#) ()

Additional Inherited Members

6.135.1 Detailed Description

Definition at line 37 of file [Storage.h](#).

6.135.2 Constructor & Destructor Documentation

6.135.2.1 Storage()

```
Storage::Storage (
    Model * model,
    std::string name = "" )
```

Definition at line 16 of file [Storage.cpp](#).

Here is the caller graph for this function:



6.135.2.2 ~Storage()

```
virtual Storage::~Storage ( ) [virtual], [default]
```

6.135.3 Member Function Documentation

6.135.3.1 `_check()`

```
bool Storage::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 59 of file [Storage.cpp](#).

6.135.3.2 `_getParserChangesInformation()`

```
ParserChangesInformation * Storage::_getParserChangesInformation () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 65 of file [Storage.cpp](#).

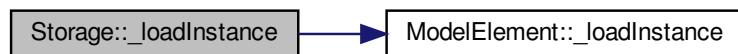
6.135.3.3 `_loadInstance()`

```
bool Storage::_loadInstance (
    std::map< std::string, std::string > * fields ) [protected], [virtual]
```

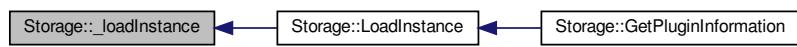
Reimplemented from [ModelElement](#).

Definition at line 40 of file [Storage.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



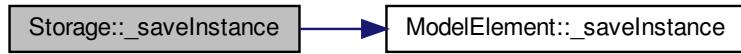
6.135.3.4 `_saveInstance()`

```
std::map< std::string, std::string > * Storage::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 52 of file [Storage.cpp](#).

Here is the call graph for this function:

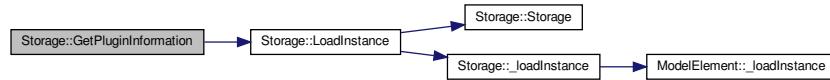


6.135.3.5 `GetPluginInformation()`

```
PluginInformation * Storage::GetPluginInformation ( ) [static]
```

Definition at line 25 of file [Storage.cpp](#).

Here is the call graph for this function:

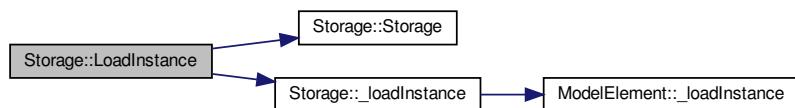


6.135.3.6 `LoadInstance()`

```
ModelElement * Storage::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 30 of file [Storage.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



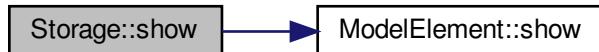
6.135.3.7 show()

```
std::string Storage::show () [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [20](#) of file [Storage.cpp](#).

Here is the call graph for this function:



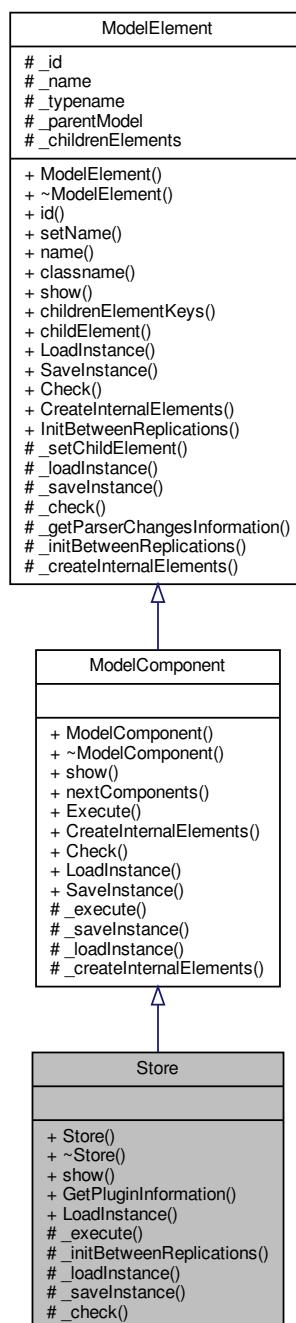
The documentation for this class was generated from the following files:

- [Storage.h](#)
- [Storage.cpp](#)

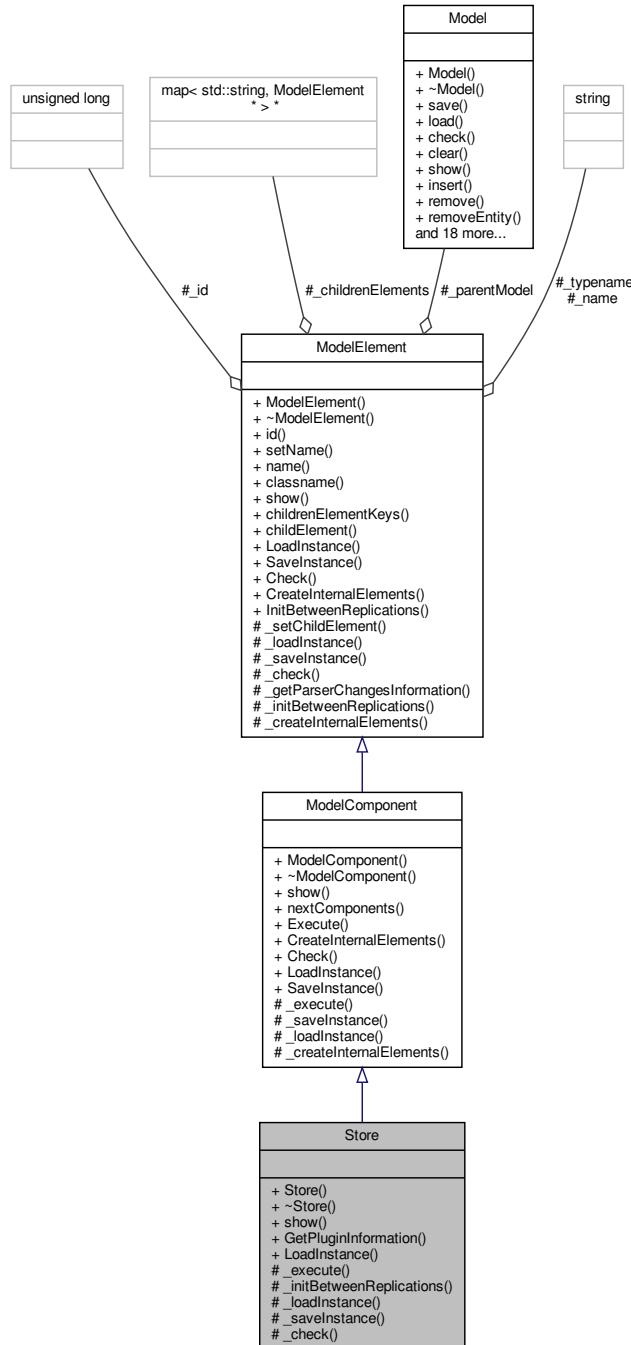
6.136 Store Class Reference

```
#include <Store.h>
```

Inheritance diagram for Store:



Collaboration diagram for Store:



Public Member Functions

- `Store (Model *model, std::string name="")`
- virtual `~Store ()=default`
- virtual `std::string show ()`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.136.1 Detailed Description

Store module DESCRIPTION The **Store** module adds an entity to storage. The **Unstore** module may then be used to remove the entity from the storage. When an entity arrives at the **Store** module, the storage specified is incremented, and the entity immediately moves to the next module in the model. Storages are useful for displaying entity animation while an entity undergoes processing in other modules. Additionally, statistics may be kept on the number of entities in storage. TYPICAL USES Animating a part through a number of delay operations (load, setup, process, unload) Tracking the number of customers within a grocery store (place in storage upon entry) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of specifying the storage name as a **Storage**, **Set**, **Attribute**, or Expression. **Storage** Name Name of the storage to which the entity will be added. Applies only when the Type is **Storage**. **Set** Name Name of the storage set from which the storage is to be selected. Applies only when the Type is **Set**. **Set** Index Index into the defined storage set that contains the desired storage name. Applies only when the Type is **Set**. **Attribute** Name of the attribute whose value contains the storage. Applies only when the Type is **Attribute**. Expression Expression that is evaluated to the storage into which the entity is placed. Applies only when the Type is **Expression**.

Definition at line 50 of file `Store.h`.

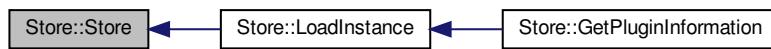
6.136.2 Constructor & Destructor Documentation

6.136.2.1 `Store()`

```
Store::Store (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file `Store.cpp`.

Here is the caller graph for this function:



6.136.2.2 ~Store()

```
virtual Store::~Store ( ) [virtual], [default]
```

6.136.3 Member Function Documentation

6.136.3.1 _check()

```
bool Store::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Store.cpp](#).

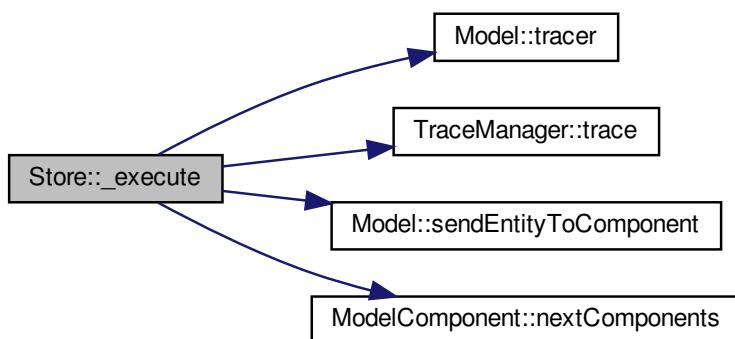
6.136.3.2 _execute()

```
void Store::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Store.cpp](#).

Here is the call graph for this function:



6.136.3.3 `_initBetweenReplications()`

```
void Store::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 48 of file [Store.cpp](#).

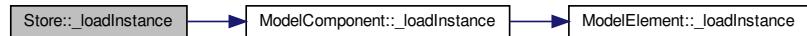
6.136.3.4 `_loadInstance()`

```
bool Store::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

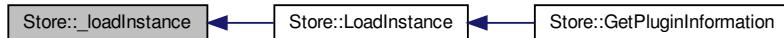
Reimplemented from [ModelComponent](#).

Definition at line 40 of file [Store.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



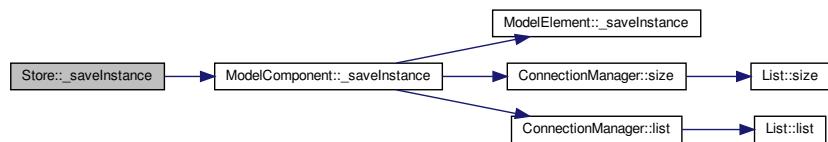
6.136.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Store::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 51 of file [Store.cpp](#).

Here is the call graph for this function:



6.136.3.6 GetPluginInformation()

```
PluginInformation * Store::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [Store.cpp](#).

Here is the call graph for this function:

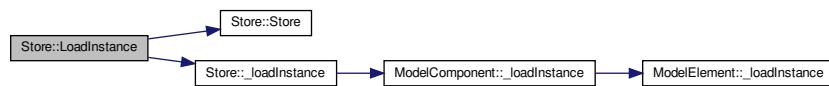


6.136.3.7 LoadInstance()

```
ModelComponent * Store::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Store.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



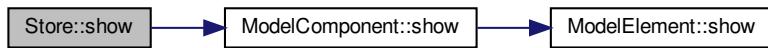
6.136.3.8 show()

```
std::string Store::show () [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 21 of file [Store.cpp](#).

Here is the call graph for this function:



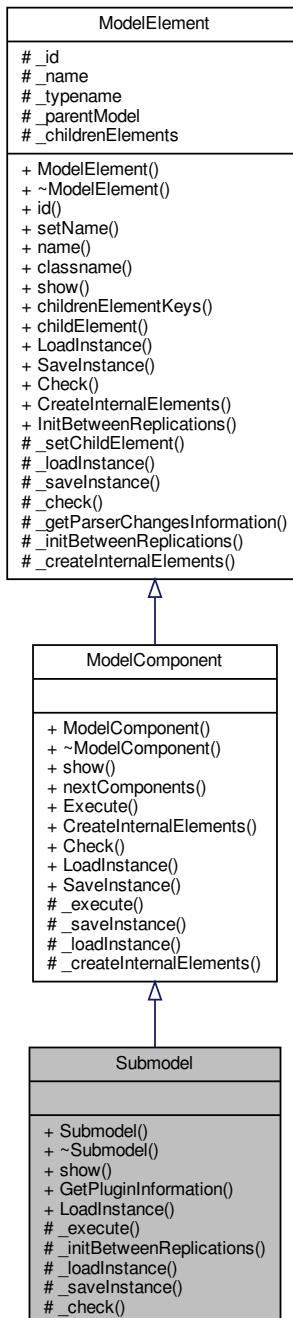
The documentation for this class was generated from the following files:

- [Store.h](#)
- [Store.cpp](#)

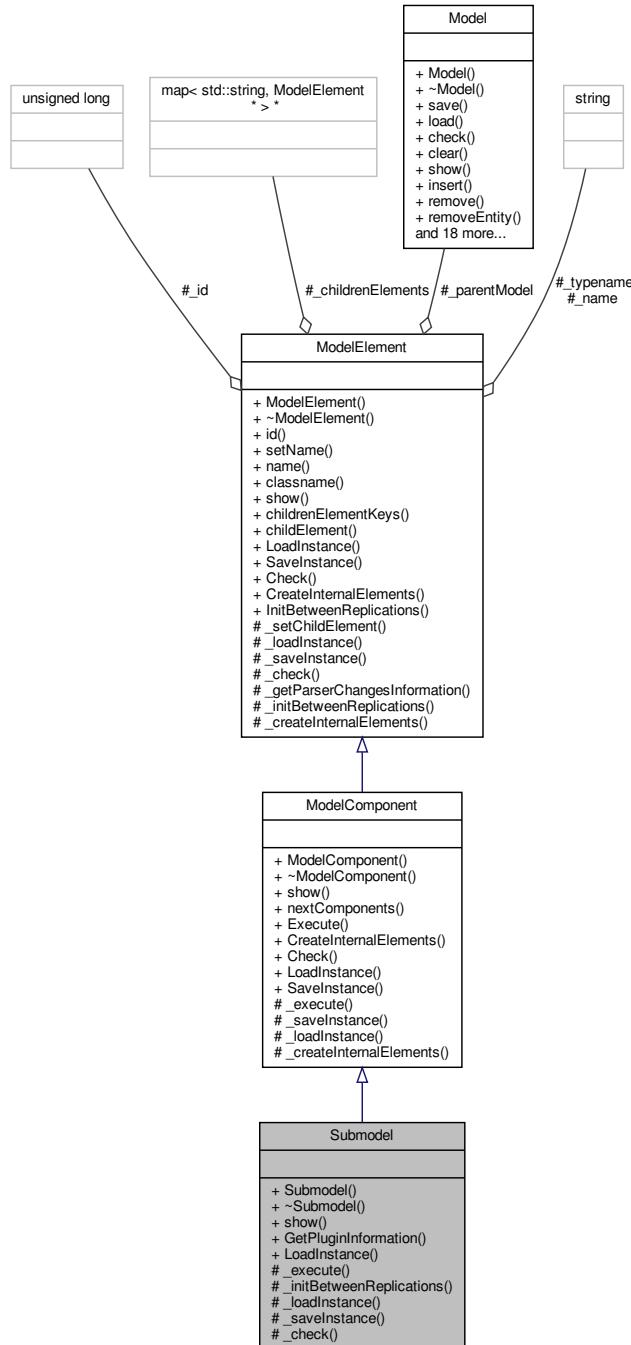
6.137 Submodel Class Reference

```
#include <Submodel.h>
```

Inheritance diagram for Submodel:



Collaboration diagram for Submodel:



Public Member Functions

- **Submodel (Model *model, std::string name="")**
- virtual **~Submodel ()=default**
- virtual std::string **show ()**

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.137.1 Detailed Description

This component ...

Definition at line 22 of file [Submodel.h](#).

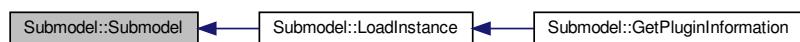
6.137.2 Constructor & Destructor Documentation

6.137.2.1 Submodel()

```
Submodel::Submodel (
    Model * model,
    std::string name = "" )
```

Definition at line 18 of file [Submodel.cpp](#).

Here is the caller graph for this function:



6.137.2.2 ~Submodel()

```
virtual Submodel::~Submodel ( ) [virtual], [default]
```

6.137.3 Member Function Documentation

6.137.3.1 `_check()`

```
bool Submodel::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 58 of file [Submodel.cpp](#).

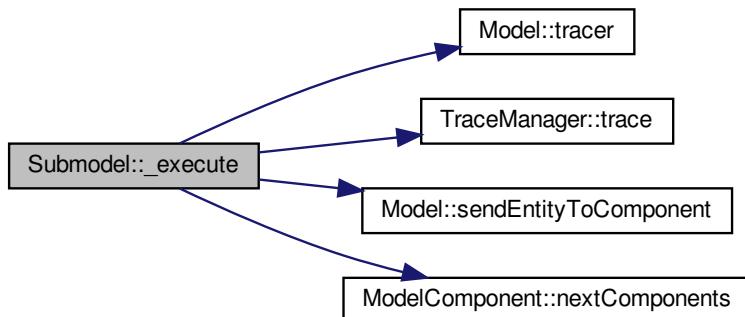
6.137.3.2 `_execute()`

```
void Submodel::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 36 of file [Submodel.cpp](#).

Here is the call graph for this function:

6.137.3.3 `_initBetweenReplications()`

```
void Submodel::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 49 of file [Submodel.cpp](#).

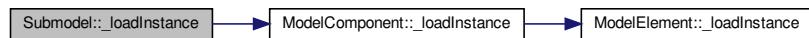
6.137.3.4 `_loadInstance()`

```
bool Submodel::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

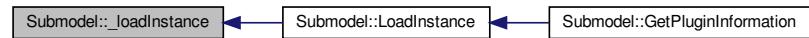
Reimplemented from [ModelComponent](#).

Definition at line 41 of file [Submodel.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



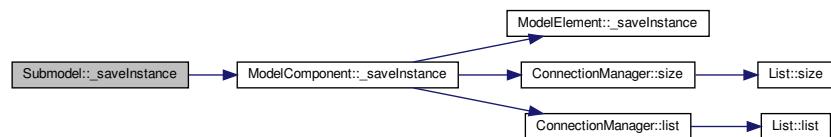
6.137.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Submodel::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 52 of file [Submodel.cpp](#).

Here is the call graph for this function:

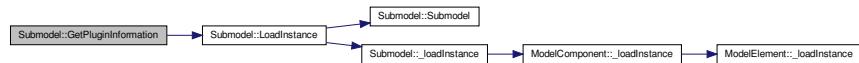


6.137.3.6 GetPluginInformation()

```
PluginInformation * Submodel::GetPluginInformation ( ) [static]
```

Definition at line 64 of file [Submodel.cpp](#).

Here is the call graph for this function:



6.137.3.7 LoadInstance()

```
ModelComponent * Submodel::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Submodel.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



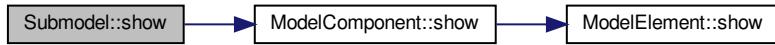
6.137.3.8 show()

```
std::string Submodel::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [22](#) of file [Submodel.cpp](#).

Here is the call graph for this function:



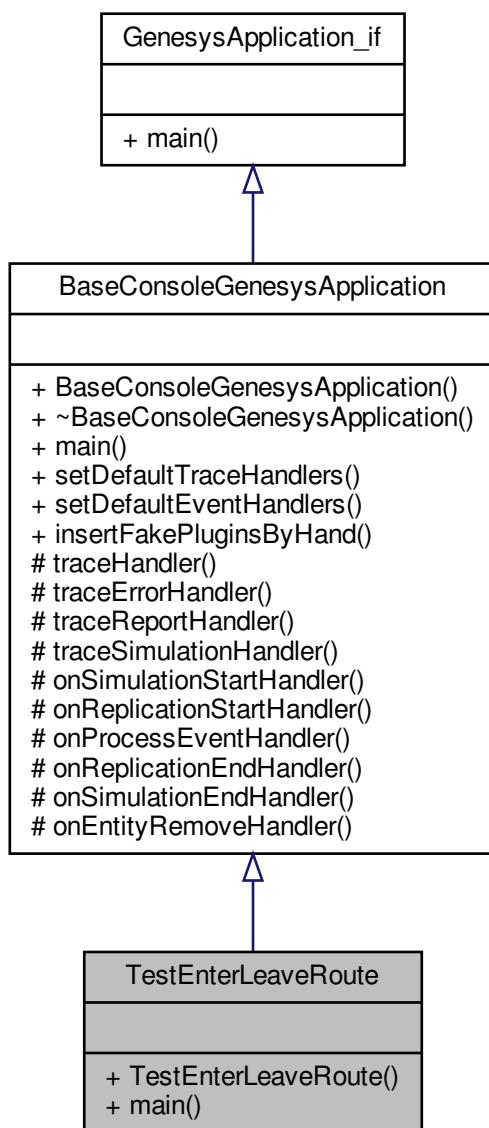
The documentation for this class was generated from the following files:

- [Submodel.h](#)
- [Submodel.cpp](#)

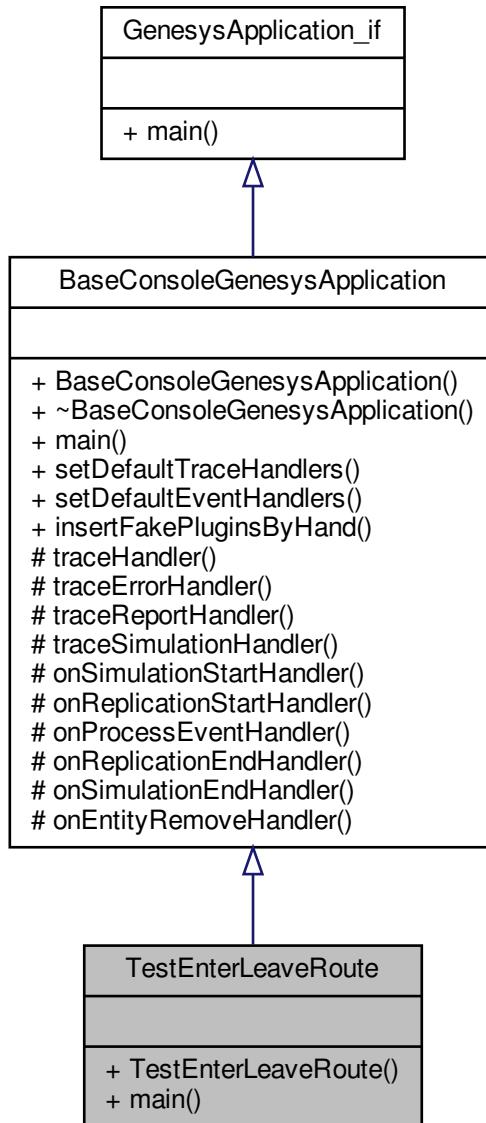
6.138 TestEnterLeaveRoute Class Reference

```
#include <TestEnterLeaveRoute.h>
```

Inheritance diagram for TestEnterLeaveRoute:



Collaboration diagram for TestEnterLeaveRoute:



Public Member Functions

- `TestEnterLeaveRoute ()`
- `int main (int argc, char **argv)`

Additional Inherited Members

6.138.1 Detailed Description

Definition at line 19 of file [TestEnterLeaveRoute.h](#).

6.138.2 Constructor & Destructor Documentation

6.138.2.1 TestEnterLeaveRoute()

```
TestEnterLeaveRoute::TestEnterLeaveRoute ( )
```

Definition at line 35 of file [TestEnterLeaveRoute.cpp](#).

6.138.3 Member Function Documentation

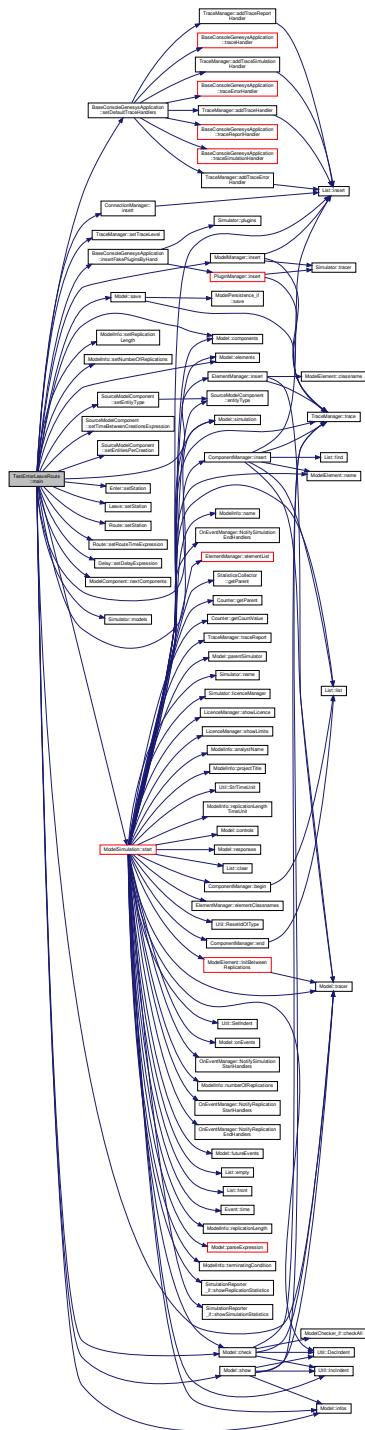
6.138.3.1 main()

```
int TestEnterLeaveRoute::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 38 of file [TestEnterLeaveRoute.cpp](#).

Here is the call graph for this function:



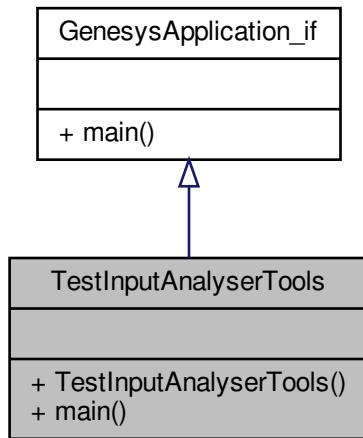
The documentation for this class was generated from the following files:

- `TestEnterLeaveRoute.h`
 - `TestEnterLeaveRoute.cpp`

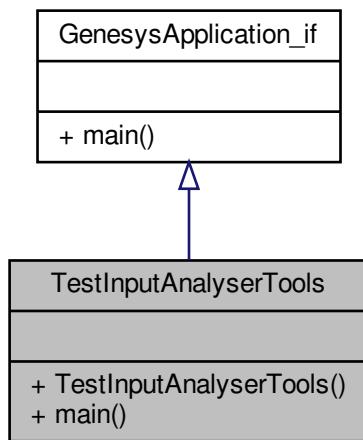
6.139 TestInputAnalyserTools Class Reference

```
#include <TestInputAnalyserTools.h>
```

Inheritance diagram for TestInputAnalyserTools:



Collaboration diagram for TestInputAnalyserTools:



Public Member Functions

- `TestInputAnalyserTools ()`
- int `main (int argc, char **argv)`

6.139.1 Detailed Description

Definition at line 21 of file [TestInputAnalyserTools.h](#).

6.139.2 Constructor & Destructor Documentation

6.139.2.1 [TestInputAnalyserTools\(\)](#)

```
TestInputAnalyserTools::TestInputAnalyserTools ( )
```

Definition at line 91 of file [TestInputAnalyserTools.cpp](#).

6.139.3 Member Function Documentation

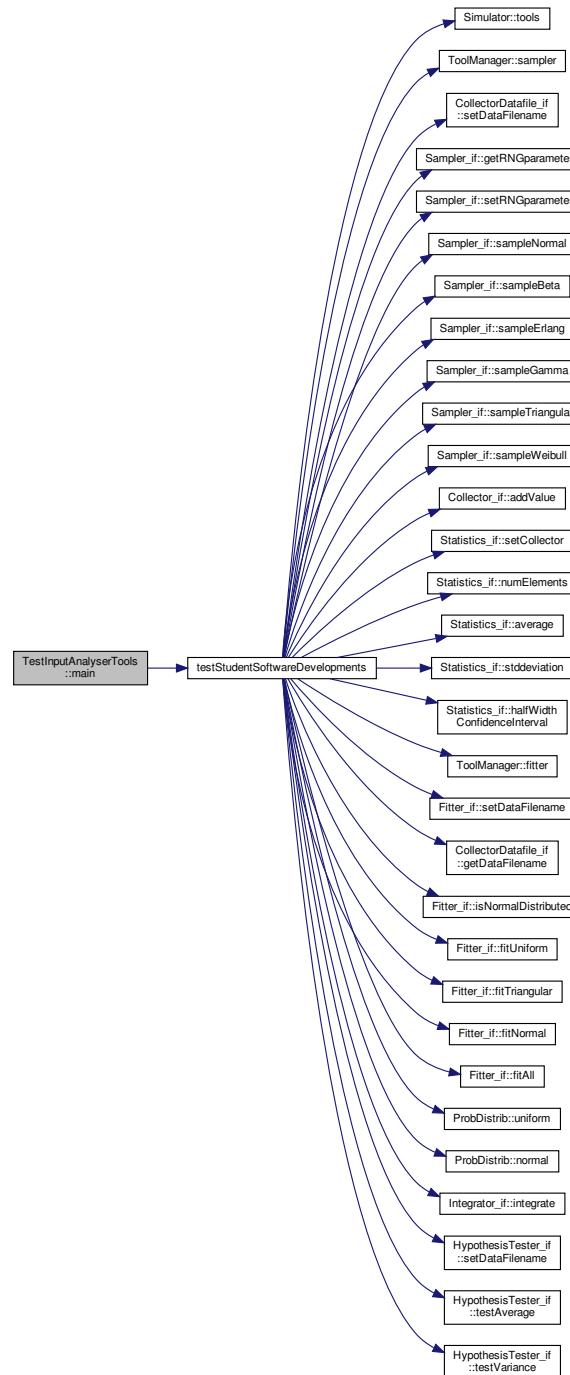
6.139.3.1 [main\(\)](#)

```
int TestInputAnalyserTools::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [GenesysApplication_if](#).

Definition at line 95 of file [TestInputAnalyserTools.cpp](#).

Here is the call graph for this function:



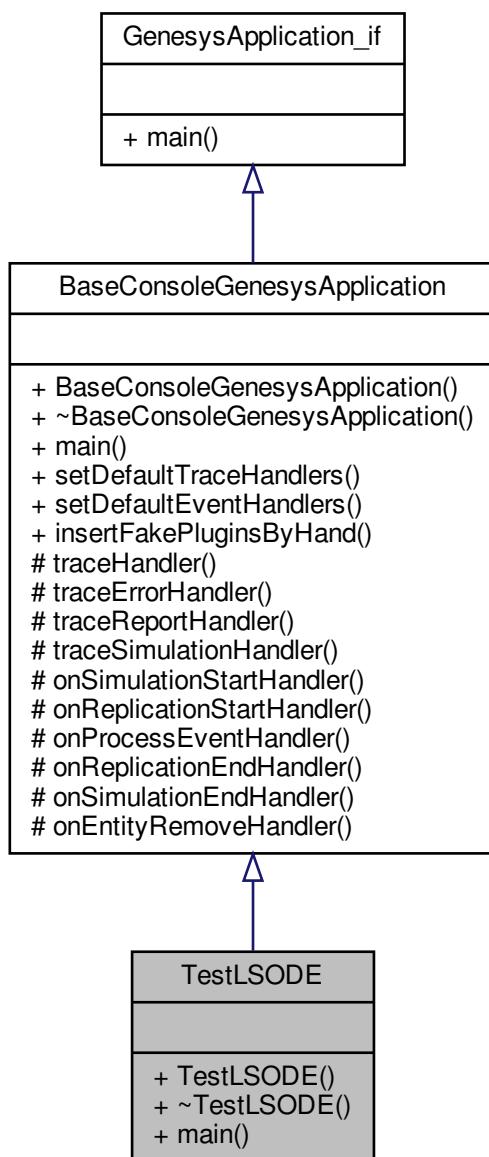
The documentation for this class was generated from the following files:

- [TestInputAnalyserTools.h](#)
- [TestInputAnalyserTools.cpp](#)

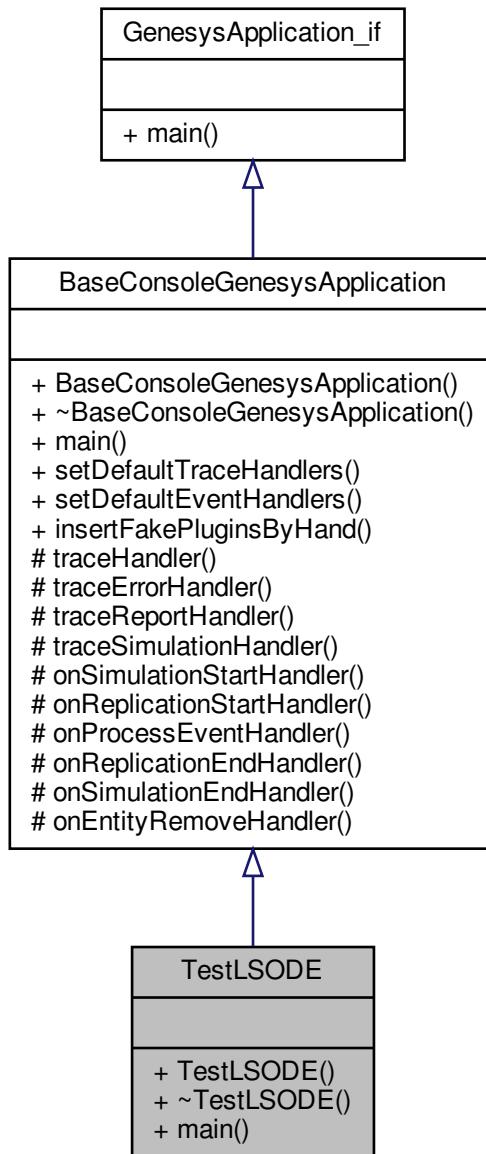
6.140 TestLSODE Class Reference

```
#include <TestLSODE.h>
```

Inheritance diagram for TestLSODE:



Collaboration diagram for TestLSODE:



Public Member Functions

- `TestLSODE ()`
- virtual `~TestLSODE ()=default`
- virtual int `main (int argc, char **argv)`

Additional Inherited Members

6.140.1 Detailed Description

Definition at line 20 of file [TestLSODE.h](#).

6.140.2 Constructor & Destructor Documentation

6.140.2.1 TestLSODE()

```
TestLSODE::TestLSODE ( )
```

Definition at line 26 of file [TestLSODE.cpp](#).

6.140.2.2 ~TestLSODE()

```
virtual TestLSODE::~TestLSODE ( ) [virtual], [default]
```

6.140.3 Member Function Documentation

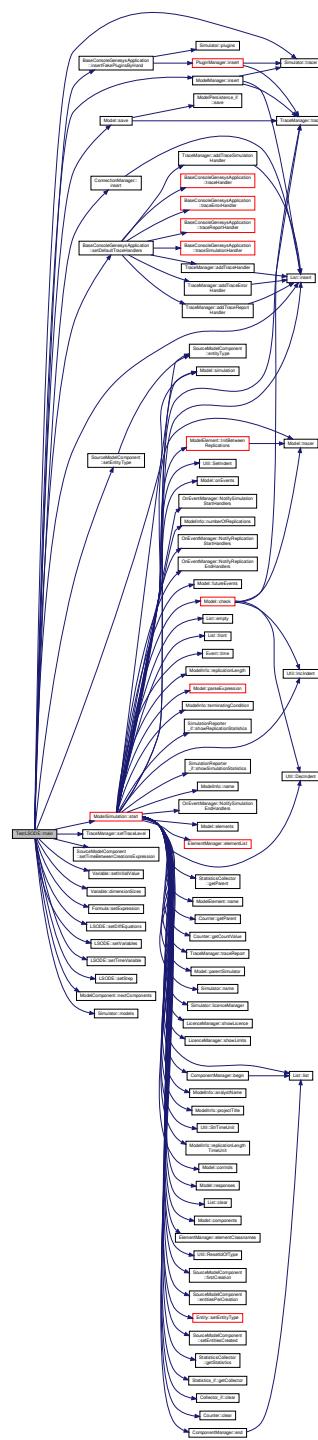
6.140.3.1 main()

```
int TestLSODE::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 29 of file [TestLSODE.cpp](#).

Here is the call graph for this function:



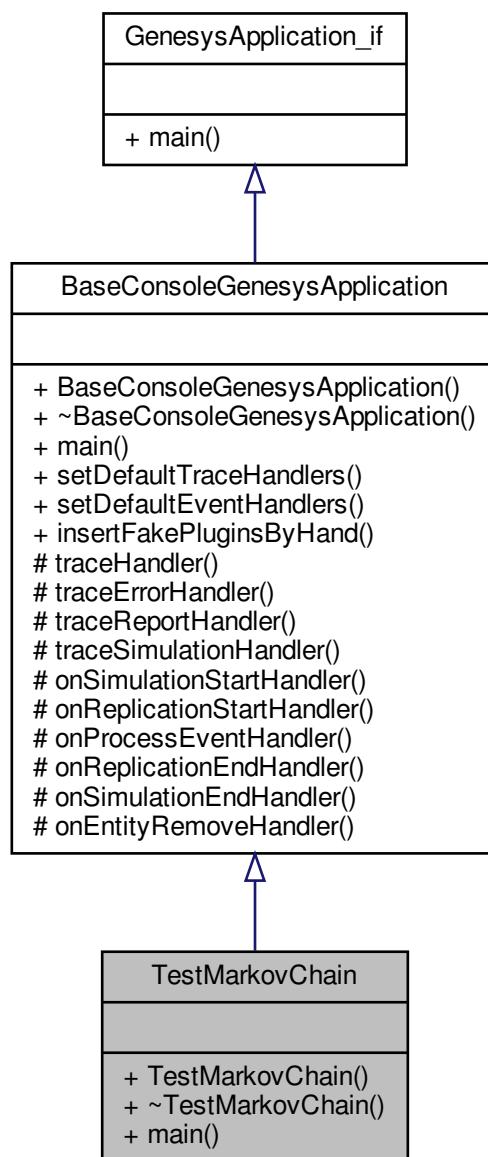
The documentation for this class was generated from the following files:

- TestLSODE.h
 - TestLSODE.cpp

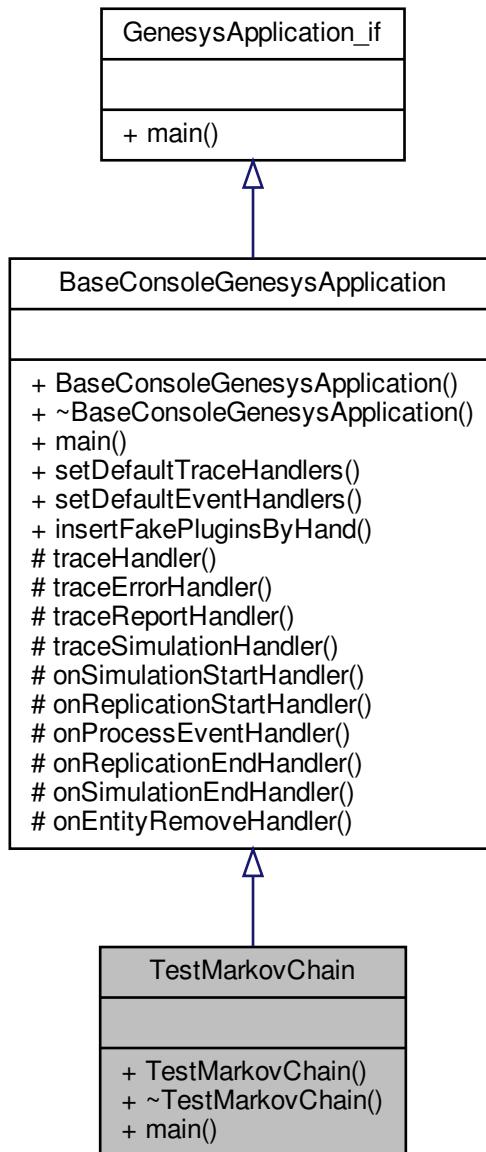
6.141 TestMarkovChain Class Reference

```
#include <TestMarkovChain.h>
```

Inheritance diagram for TestMarkovChain:



Collaboration diagram for TestMarkovChain:



Public Member Functions

- `TestMarkovChain ()`
- virtual `~TestMarkovChain ()=default`
- virtual int `main (int argc, char **argv)`

Additional Inherited Members

6.141.1 Detailed Description

Definition at line 19 of file [TestMarkovChain.h](#).

6.141.2 Constructor & Destructor Documentation

6.141.2.1 TestMarkovChain()

```
TestMarkovChain::TestMarkovChain ( )
```

Definition at line 24 of file [TestMarkovChain.cpp](#).

6.141.2.2 ~TestMarkovChain()

```
virtual TestMarkovChain::~TestMarkovChain ( ) [virtual], [default]
```

6.141.3 Member Function Documentation

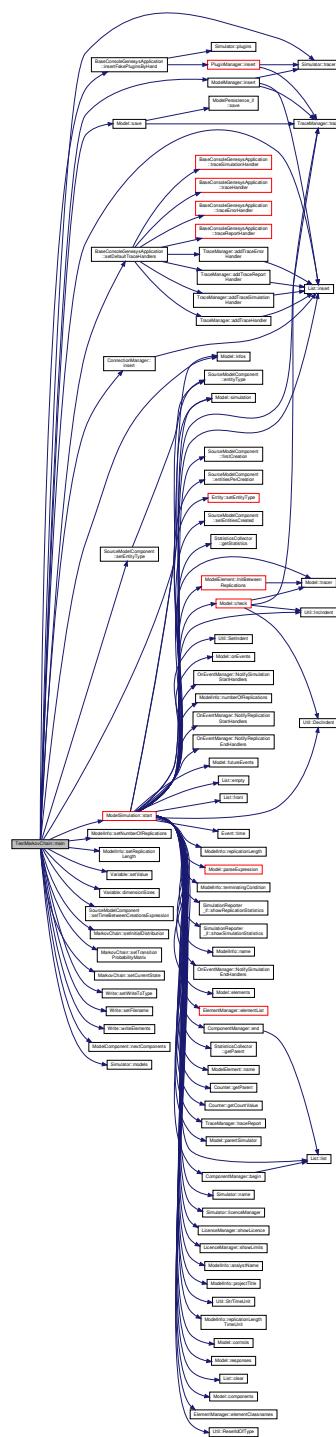
6.141.3.1 main()

```
int TestMarkovChain::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 27 of file [TestMarkovChain.cpp](#).

Here is the call graph for this function:



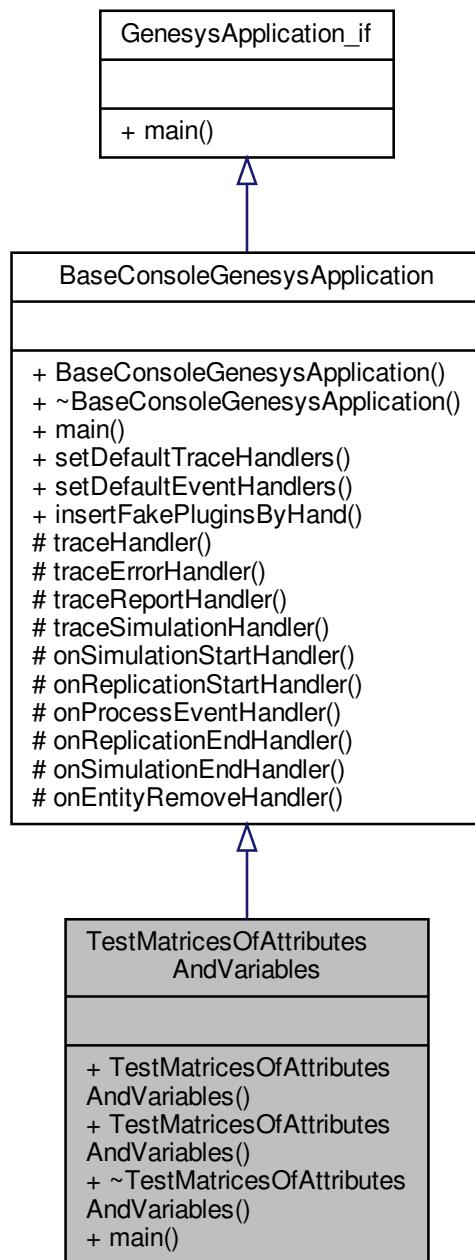
The documentation for this class was generated from the following files:

- `TestMarkovChain.h`
 - `TestMarkovChain.cpp`

6.142 TestMatricesOfAttributesAndVariables Class Reference

```
#include <TestMatricesOfAttributesAndVariables.h>
```

Inheritance diagram for TestMatricesOfAttributesAndVariables:



Collaboration diagram for TestMatricesOfAttributesAndVariables:



Public Member Functions

- `TestMatricesOfAttributesAndVariables ()`
- `TestMatricesOfAttributesAndVariables (const TestMatricesOfAttributesAndVariables &orig)`
- `virtual ~TestMatricesOfAttributesAndVariables ()`
- `virtual int main (int argc, char **argv)`

Additional Inherited Members

6.142.1 Detailed Description

Definition at line 20 of file [TestMatricesOfAttributesAndVariables.h](#).

6.142.2 Constructor & Destructor Documentation

6.142.2.1 `TestMatricesOfAttributesAndVariables()` [1/2]

```
TestMatricesOfAttributesAndVariables::TestMatricesOfAttributesAndVariables ( )
```

Definition at line 27 of file [TestMatricesOfAttributesAndVariables.cpp](#).

6.142.2.2 `TestMatricesOfAttributesAndVariables()` [2/2]

```
TestMatricesOfAttributesAndVariables::TestMatricesOfAttributesAndVariables ( const TestMatricesOfAttributesAndVariables & orig )
```

Definition at line 30 of file [TestMatricesOfAttributesAndVariables.cpp](#).

6.142.2.3 `~TestMatricesOfAttributesAndVariables()`

```
TestMatricesOfAttributesAndVariables::~TestMatricesOfAttributesAndVariables ( ) [virtual]
```

Definition at line 33 of file [TestMatricesOfAttributesAndVariables.cpp](#).

6.142.3 Member Function Documentation

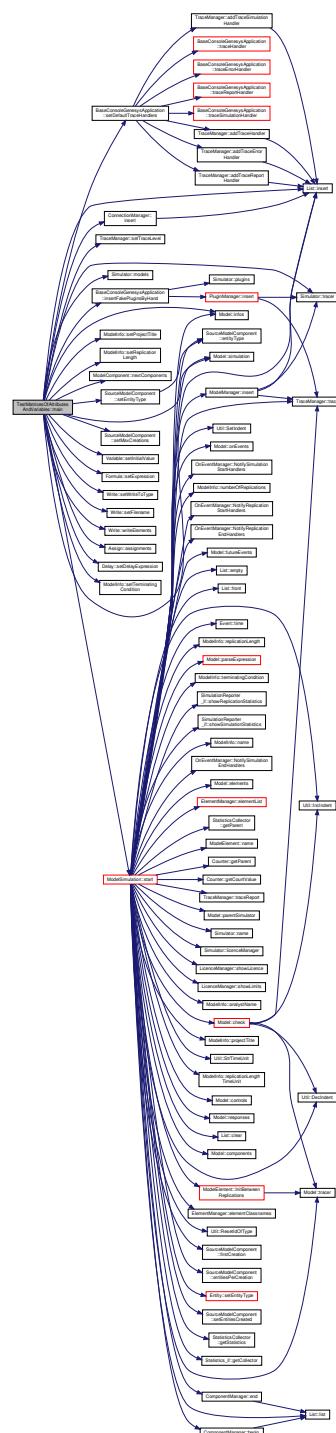
6.142.3.1 main()

```
int TestMatricesOfAttributesAndVariables::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 36 of file [TestMatricesOfAttributesAndVariables.cpp](#).

Here is the call graph for this function:



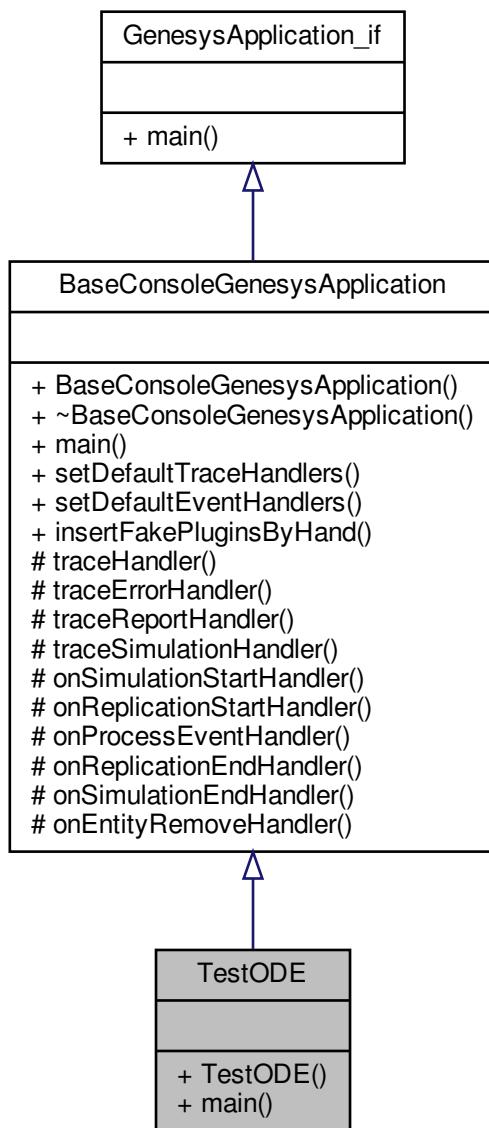
The documentation for this class was generated from the following files:

- [TestMatricesOfAttributesAndVariables.h](#)
- [TestMatricesOfAttributesAndVariables.cpp](#)

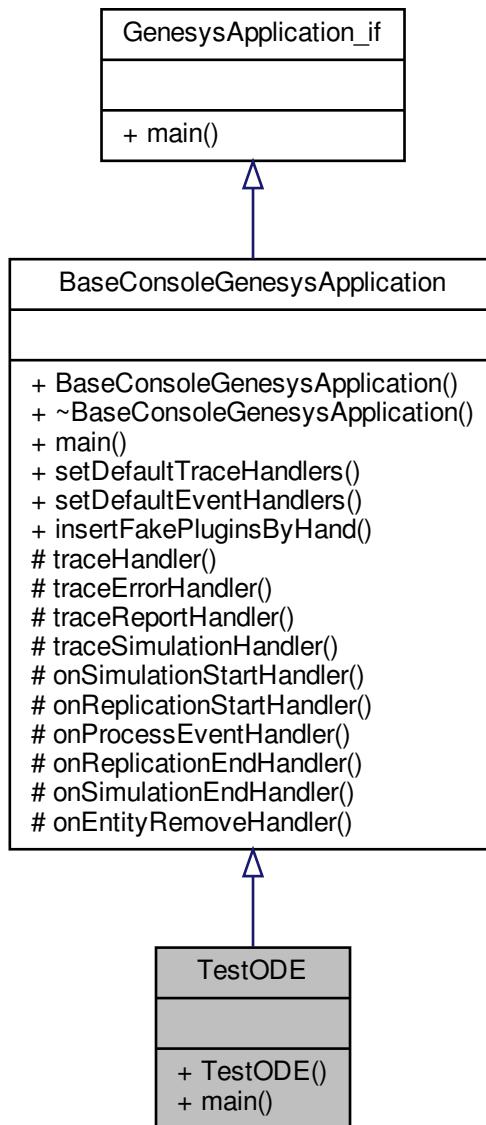
6.143 TestODE Class Reference

```
#include <TestFunctions.h>
```

Inheritance diagram for TestODE:



Collaboration diagram for TestODE:



Public Member Functions

- `TestODE ()`
- virtual int `main` (int argc, char **argv)

Additional Inherited Members

6.143.1 Detailed Description

Definition at line 19 of file [TestFunctions.h](#).

6.143.2 Constructor & Destructor Documentation

6.143.2.1 TestODE()

```
TestODE::TestODE ( )
```

Definition at line 31 of file [TestFunctions.cpp](#).

6.143.3 Member Function Documentation

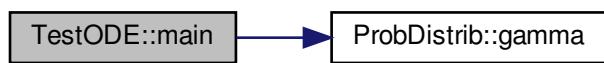
6.143.3.1 main()

```
int TestODE::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 34 of file [TestFunctions.cpp](#).

Here is the call graph for this function:



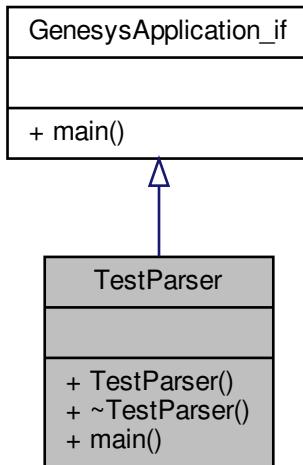
The documentation for this class was generated from the following files:

- [TestFunctions.h](#)
- [TestFunctions.cpp](#)

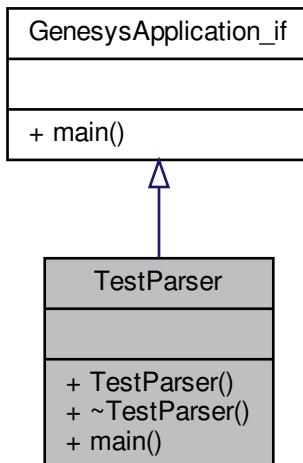
6.144 TestParser Class Reference

```
#include <TestParser.h>
```

Inheritance diagram for TestParser:



Collaboration diagram for TestParser:



Public Member Functions

- `TestParser ()`
- `virtual ~TestParser ()=default`
- `virtual int main (int argc, char **argv)`

6.144.1 Detailed Description

Definition at line 19 of file [TestParser.h](#).

6.144.2 Constructor & Destructor Documentation

6.144.2.1 TestParser()

```
TestParser::TestParser ( )
```

Definition at line 19 of file [TestParser.cpp](#).

6.144.2.2 ~TestParser()

```
virtual TestParser::~TestParser ( ) [virtual], [default]
```

6.144.3 Member Function Documentation

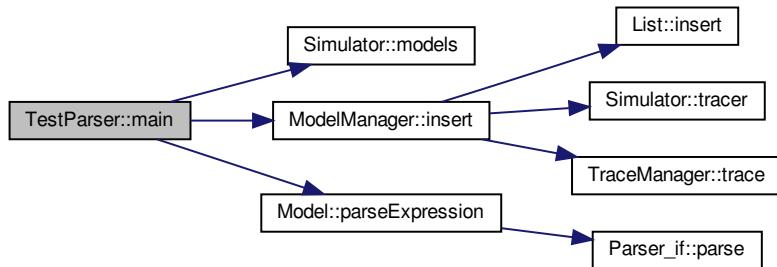
6.144.3.1 main()

```
int TestParser::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [GenesysApplication_if](#).

Definition at line 23 of file [TestParser.cpp](#).

Here is the call graph for this function:



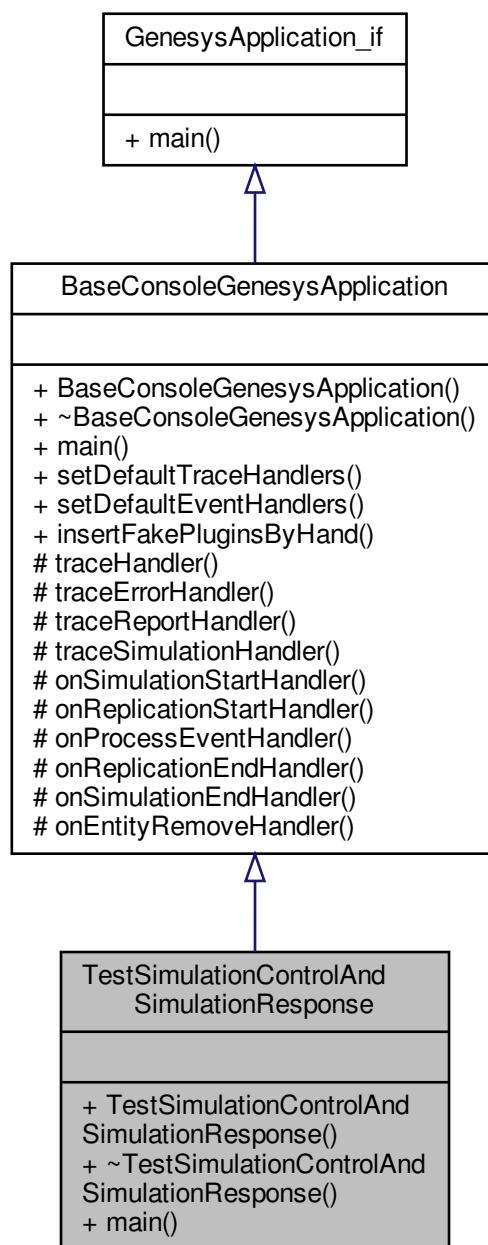
The documentation for this class was generated from the following files:

- [TestParser.h](#)
- [TestParser.cpp](#)

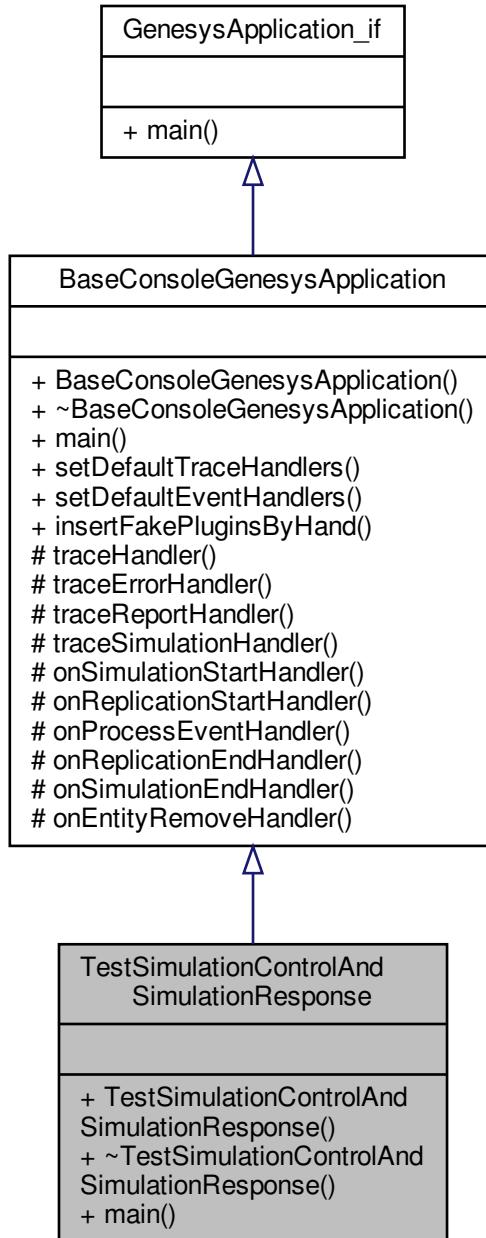
6.145 TestSimulationControlAndSimulationResponse Class Reference

```
#include <TestSimulationControlAndSimulationResponse.h>
```

Inheritance diagram for TestSimulationControlAndSimulationResponse:



Collaboration diagram for TestSimulationControlAndSimulationResponse:



Public Member Functions

- `TestSimulationControlAndSimulationResponse ()`
- `virtual ~TestSimulationControlAndSimulationResponse ()=default`
- `virtual int main (int argc, char **argv)`

Additional Inherited Members

6.145.1 Detailed Description

Definition at line 19 of file [TestSimulationControlAndSimulationResponse.h](#).

6.145.2 Constructor & Destructor Documentation

6.145.2.1 `TestSimulationControlAndSimulationResponse()`

```
TestSimulationControlAndSimulationResponse::TestSimulationControlAndSimulationResponse ()
```

Definition at line 21 of file [TestSimulationControlAndSimulationResponse.cpp](#).

6.145.2.2 `~TestSimulationControlAndSimulationResponse()`

```
virtual TestSimulationControlAndSimulationResponse::~TestSimulationControlAndSimulationResponse () [virtual], [default]
```

6.145.3 Member Function Documentation

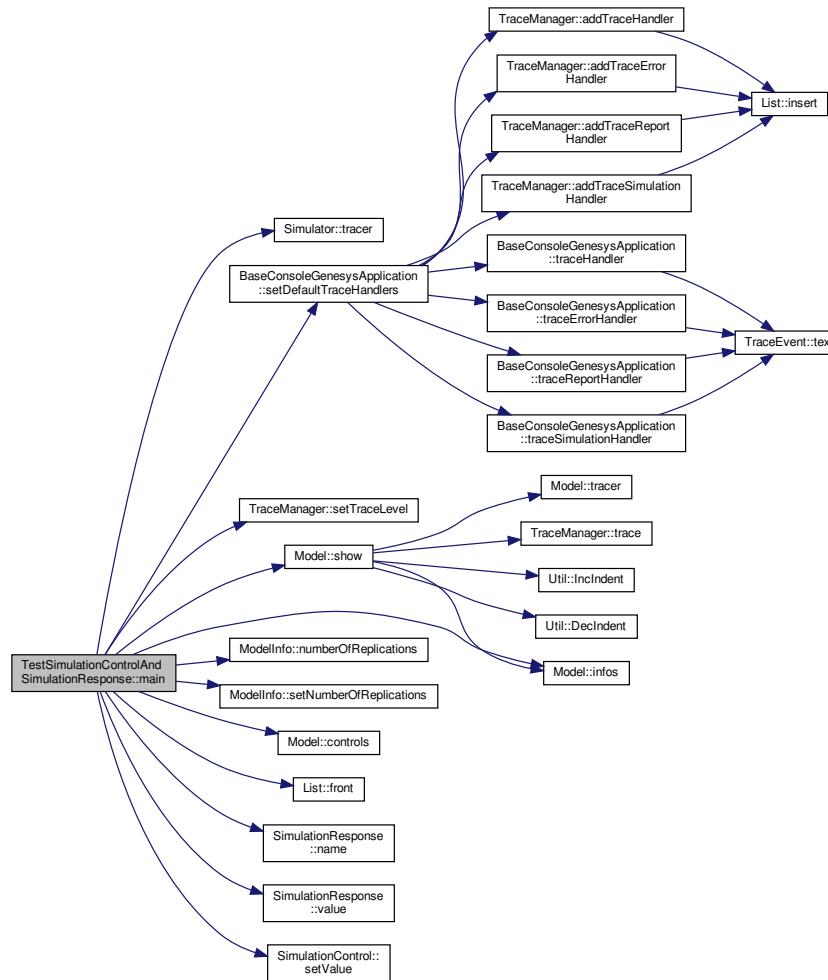
6.145.3.1 `main()`

```
int TestSimulationControlAndSimulationResponse::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [BaseConsoleGenesysApplication](#).

Definition at line 25 of file [TestSimulationControlAndSimulationResponse.cpp](#).

Here is the call graph for this function:



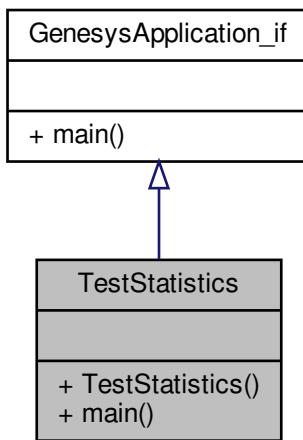
The documentation for this class was generated from the following files:

- [TestSimulationControlAndSimulationResponse.h](#)
- [TestSimulationControlAndSimulationResponse.cpp](#)

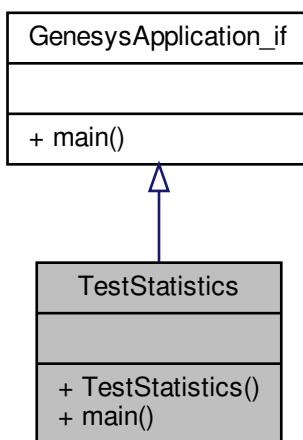
6.146 TestStatistics Class Reference

```
#include <TestStatistics.h>
```

Inheritance diagram for TestStatistics:



Collaboration diagram for TestStatistics:



Public Member Functions

- [TestStatistics \(\)](#)
- int [main \(int argc, char **argv\)](#)

6.146.1 Detailed Description

Definition at line [19](#) of file [TestStatistics.h](#).

6.146.2 Constructor & Destructor Documentation

6.146.2.1 TestStatistics()

```
TestStatistics::TestStatistics ( )
```

Definition at line 20 of file [TestStatistics.cpp](#).

6.146.3 Member Function Documentation

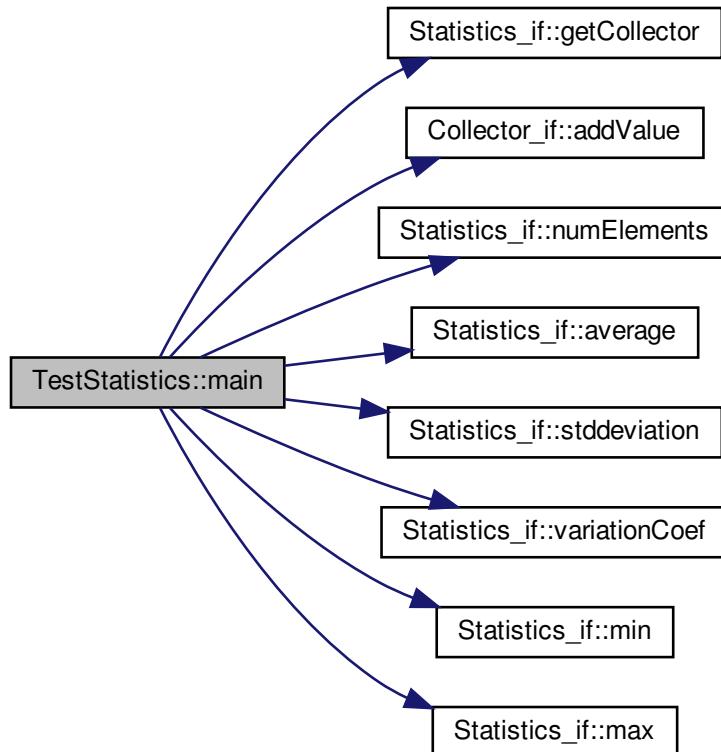
6.146.3.1 main()

```
int TestStatistics::main (
    int argc,
    char ** argv ) [virtual]
```

Implements [GenesysApplication_if](#).

Definition at line 23 of file [TestStatistics.cpp](#).

Here is the call graph for this function:



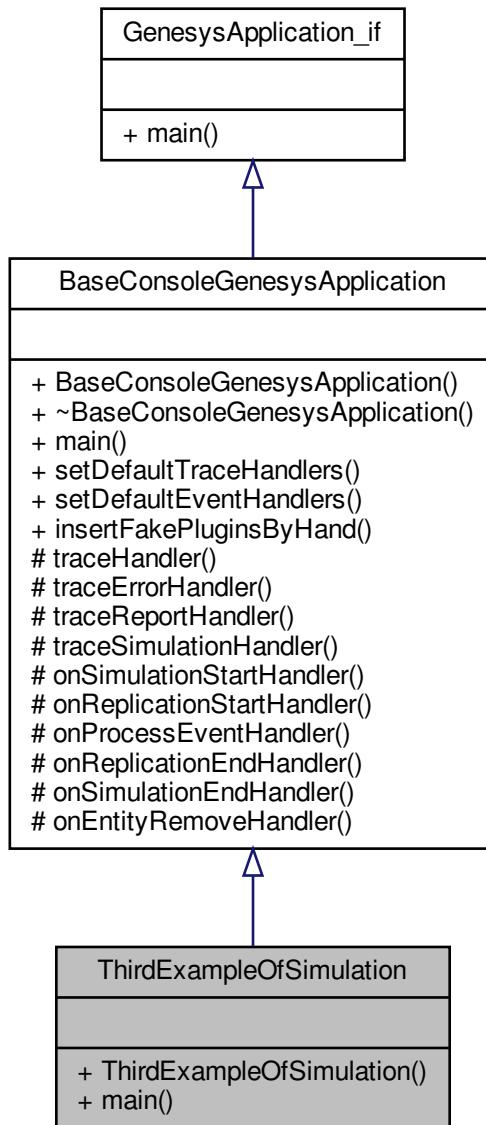
The documentation for this class was generated from the following files:

- [TestStatistics.h](#)
- [TestStatistics.cpp](#)

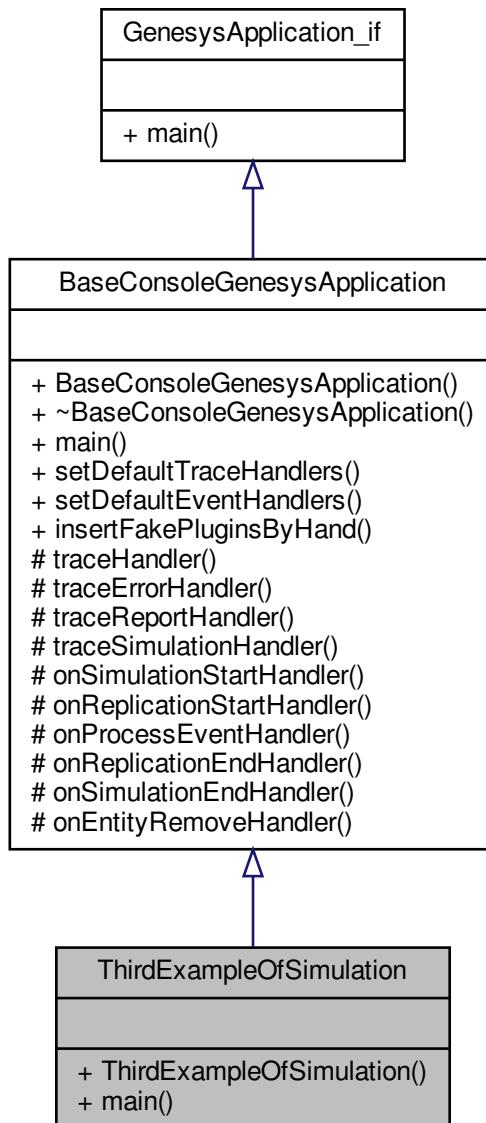
6.147 ThirdExampleOfSimulation Class Reference

```
#include <ThirdExampleOfSimultion.h>
```

Inheritance diagram for ThirdExampleOfSimulation:



Collaboration diagram for ThirdExampleOfSimulation:



Public Member Functions

- `ThirdExampleOfSimulation ()`
- `virtual int main (int argc, char **argv)`

Additional Inherited Members

6.147.1 Detailed Description

Definition at line 19 of file [ThirdExampleOfSimultion.h](#).

6.147.2 Constructor & Destructor Documentation

6.147.2.1 ThirdExampleOfSimulation()

```
ThirdExampleOfSimulation::ThirdExampleOfSimulation ( )
```

Definition at line 31 of file [ThirdExampleOfSimulation.cpp](#).

6.147.3 Member Function Documentation

6.147.3.1 main()

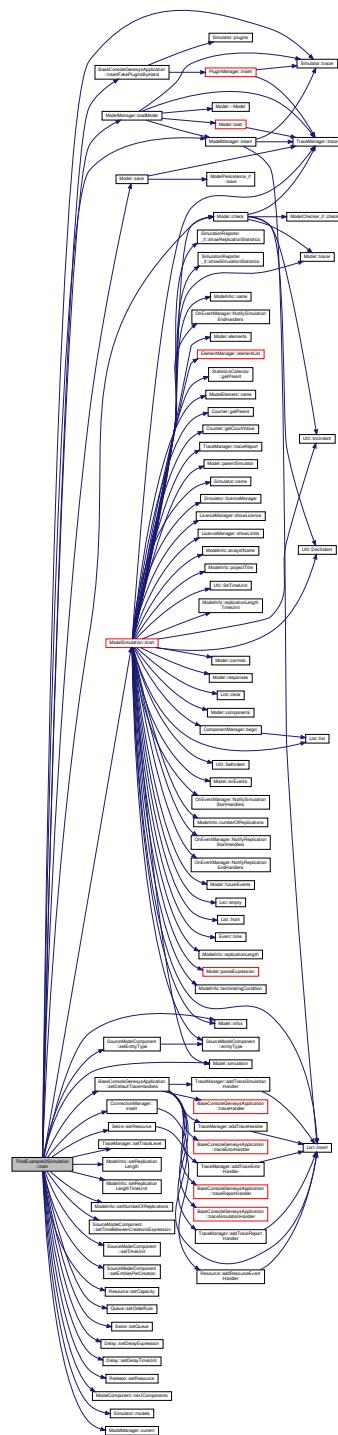
```
int ThirdExampleOfSimulation::main (
    int argc,
    char ** argv ) [virtual]
```

This is the main function of the application. It instantiates the simulator, builds a simulation model and then simulate that model.

Implements [BaseConsoleGenesysApplication](#).

Definition at line 38 of file [ThirdExampleOfSimulation.cpp](#).

Here is the call graph for this function:



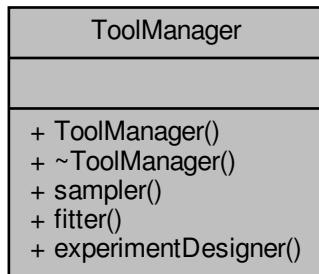
The documentation for this class was generated from the following files:

- ThirdExampleOfSimulation.h
 - ThirdExampleOfSimulation.cpp

6.148 ToolManager Class Reference

```
#include <ToolManager.h>
```

Collaboration diagram for ToolManager:



Public Member Functions

- [ToolManager \(Simulator *_simulator\)](#)
- virtual [~ToolManager \(\)=default](#)
- [Sampler_if * sampler \(\) const](#)
Returns the Sampler, used to generate samples accordingly to a probability distribution.
- [Fitter_if * fitter \(\) const](#)
- [ProcessAnalyser_if * experimentDesigner \(\) const](#)
Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

6.148.1 Detailed Description

Definition at line [23](#) of file [ToolManager.h](#).

6.148.2 Constructor & Destructor Documentation

6.148.2.1 ToolManager()

```
ToolManager::ToolManager (   
    Simulator * _simulator )
```

Definition at line [18](#) of file [ToolManager.cpp](#).

6.148.2.2 ~ToolManager()

```
virtual ToolManager::~ToolManager ( ) [virtual], [default]
```

6.148.3 Member Function Documentation

6.148.3.1 experimentDesigner()

```
ProcessAnalyser_if * ToolManager::experimentDesigner ( ) const
```

Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

Definition at line 33 of file [ToolManager.cpp](#).

6.148.3.2 fitter()

```
Fitter_if * ToolManager::fitter ( ) const
```

Definition at line 29 of file [ToolManager.cpp](#).

Here is the caller graph for this function:



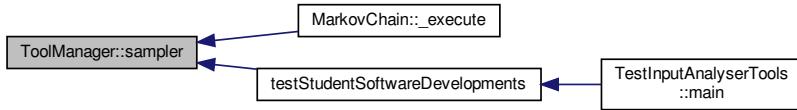
6.148.3.3 sampler()

```
Sampler_if * ToolManager::sampler ( ) const
```

Returns the Sampler, used to generate samples accordingly to a probability distribution.

Definition at line 25 of file [ToolManager.cpp](#).

Here is the caller graph for this function:



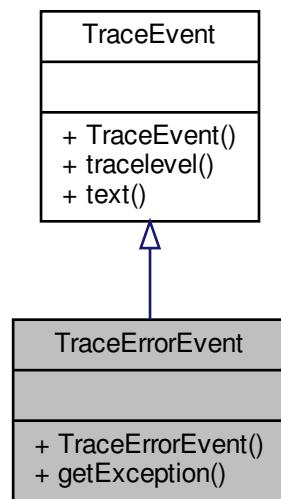
The documentation for this class was generated from the following files:

- [ToolManager.h](#)
- [ToolManager.cpp](#)

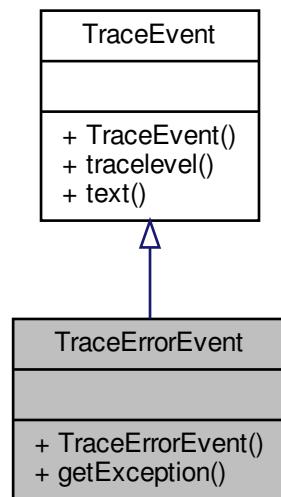
6.149 TraceErrorEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceErrorEvent:



Collaboration diagram for TraceErrorEvent:



Public Member Functions

- [TraceErrorEvent \(std::string `text`, std::exception `e`\)](#)
- [std::exception `getException \(\) const`](#)

6.149.1 Detailed Description

Definition at line [44](#) of file [TraceManager.h](#).

6.149.2 Constructor & Destructor Documentation

6.149.2.1 TraceErrorEvent()

```
TraceErrorEvent::TraceErrorEvent (   
    std::string text,  
    std::exception e ) [inline]
```

Definition at line [47](#) of file [TraceManager.h](#).

6.149.3 Member Function Documentation

6.149.3.1 getException()

```
std::exception TraceErrorEvent::getException ( ) const [inline]
```

Definition at line [51](#) of file [TraceManager.h](#).

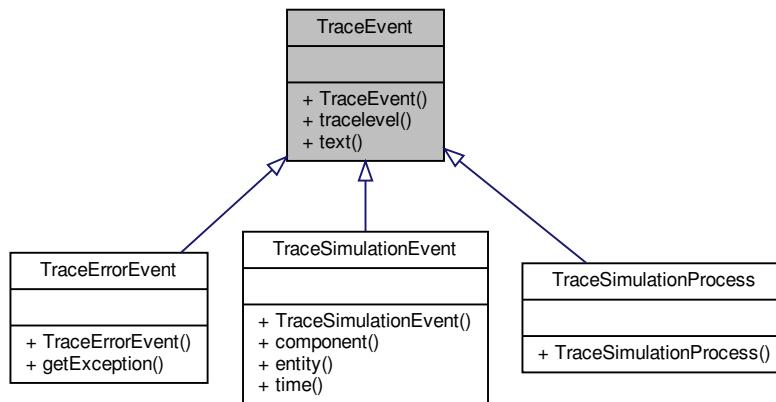
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

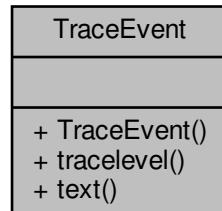
6.150 TraceEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceEvent:



Collaboration diagram for TraceEvent:



Public Member Functions

- `TraceEvent (Util::TraceLevel level, std::string text)`
- `Util::TraceLevel tracelevel () const`
- `std::string text () const`

6.150.1 Detailed Description

Definition at line 24 of file [TraceManager.h](#).

6.150.2 Constructor & Destructor Documentation

6.150.2.1 TraceEvent()

```
TraceEvent::TraceEvent (
    Util::TraceLevel level,
    std::string text ) [inline]
```

Definition at line 27 of file [TraceManager.h](#).

Here is the call graph for this function:



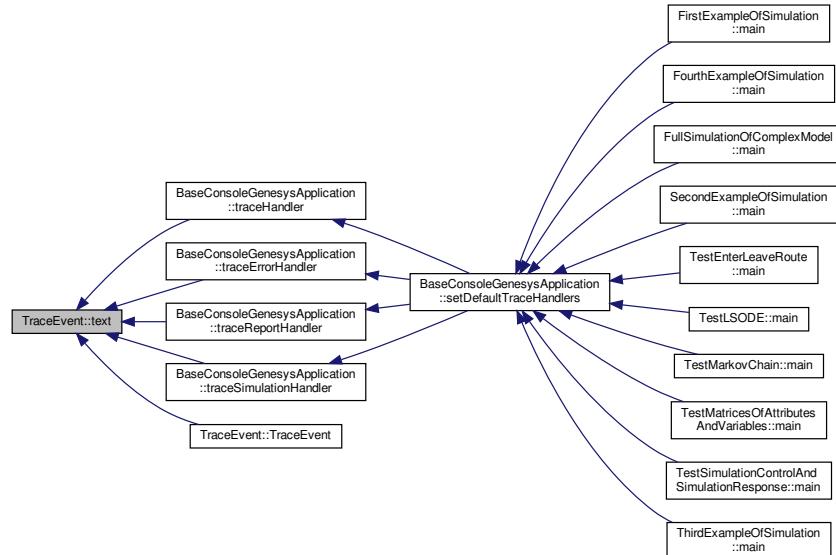
6.150.3 Member Function Documentation

6.150.3.1 text()

```
std::string TraceEvent::text ( ) const [inline]
```

Definition at line 36 of file [TraceManager.h](#).

Here is the caller graph for this function:



6.150.3.2 tracelevel()

```
Util::TraceLevel TraceEvent::tracelevel () const [inline]
```

Definition at line 32 of file [TraceManager.h](#).

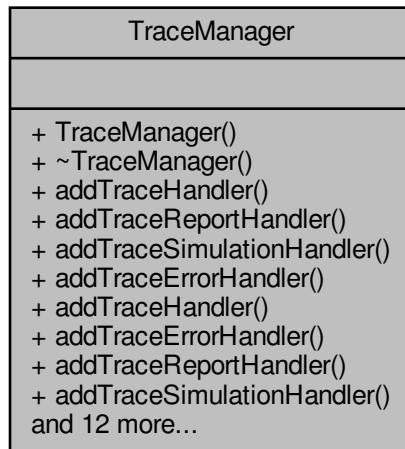
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

6.151 TraceManager Class Reference

```
#include <TraceManager.h>
```

Collaboration diagram for TraceManager:



Public Member Functions

- `TraceManager (Simulator *simulator)`
- virtual `~TraceManager ()=default`
- void `addTraceHandler (traceListener traceListener)`
- void `addTraceReportHandler (traceListener traceReportListener)`
- void `addTraceSimulationHandler (traceSimulationListener traceSimulationListener)`
- void `addTraceErrorHandler (traceErrorListener traceErrorListener)`
- template<typename Class >
void `addTraceHandler (Class *object, void(Class::*function)(TraceEvent))`
- template<typename Class >
void `addTraceErrorHandler (Class *object, void(Class::*function)(TraceErrorEvent))`
- template<typename Class >
void `addTraceReportHandler (Class *object, void(Class::*function)(TraceEvent))`

- template<typename Class >
void **addTraceSimulationHandler** (Class *object, void(Class::*function)(TraceSimulationEvent))
- void **trace** (Util::TraceLevel level, std::string text)
- void **traceError** (std::exception e, std::string text)
- void **traceReport** (Util::TraceLevel level, std::string text)
- void **traceSimulation** (Util::TraceLevel level, double time, Entity *entity, ModelComponent *component, std::string text)
- void **trace** (std::string text, Util::TraceLevel level=Util::TraceLevel::componentInternal)
- void **traceError** (std::string text, std::exception e)
- void **traceReport** (std::string text, Util::TraceLevel level=Util::TraceLevel::report)
- void **traceSimulation** (double time, Entity *entity, ModelComponent *component, std::string text, Util::TraceLevel level=Util::TraceLevel::componentInternal)
- List< std::string > * **errorMessages** () const
- void **setTraceLevel** (Util::TraceLevel _traceLevel)
- Util::TraceLevel **traceLevel** () const
- Simulator * **parentSimulator** () const

6.151.1 Detailed Description

The **TraceManager** is used to trace back model simulation information and track/debug the simulation. It works as the model simulation output (cout) and allows external methods to hook up such output as listeners.

Definition at line 111 of file [TraceManager.h](#).

6.151.2 Constructor & Destructor Documentation

6.151.2.1 TraceManager()

```
TraceManager::TraceManager (
    Simulator * simulator )
```

Definition at line 17 of file [TraceManager.cpp](#).

6.151.2.2 ~TraceManager()

```
virtual TraceManager::~TraceManager ( ) [virtual], [default]
```

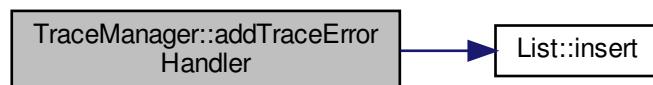
6.151.3 Member Function Documentation

6.151.3.1 addTraceErrorHandler() [1/2]

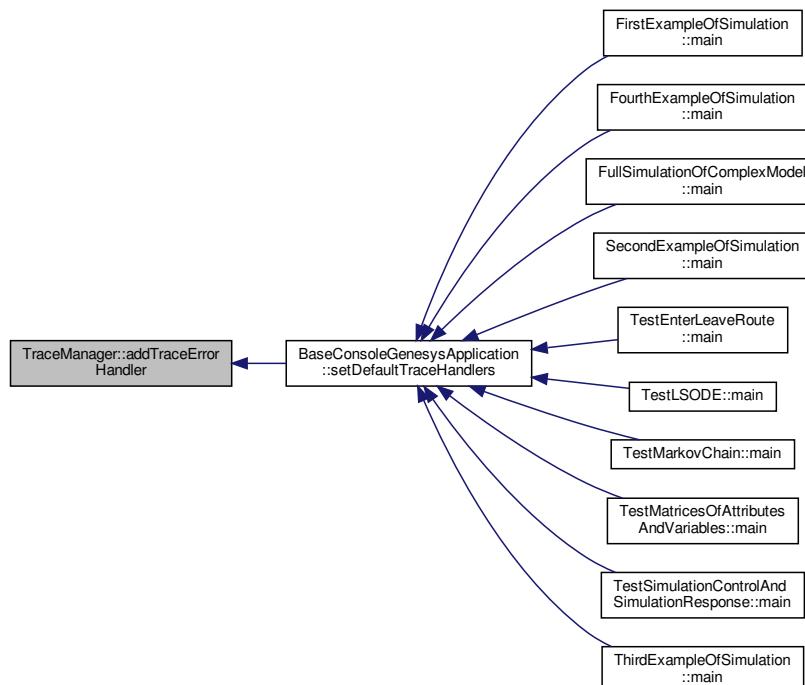
```
void TraceManager::addTraceErrorHandler (
    traceErrorListener traceErrorListener )
```

Definition at line 60 of file [TraceManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.151.3.2 addTraceErrorHandler() [2/2]

```
template<typename Class >
void TraceManager::addTraceErrorHandler (
    Class * object,
    void(Class::*)(TraceErrorEvent) function )
```

Definition at line 174 of file [TraceManager.h](#).

6.151.3.3 addTraceHandler() [1/2]

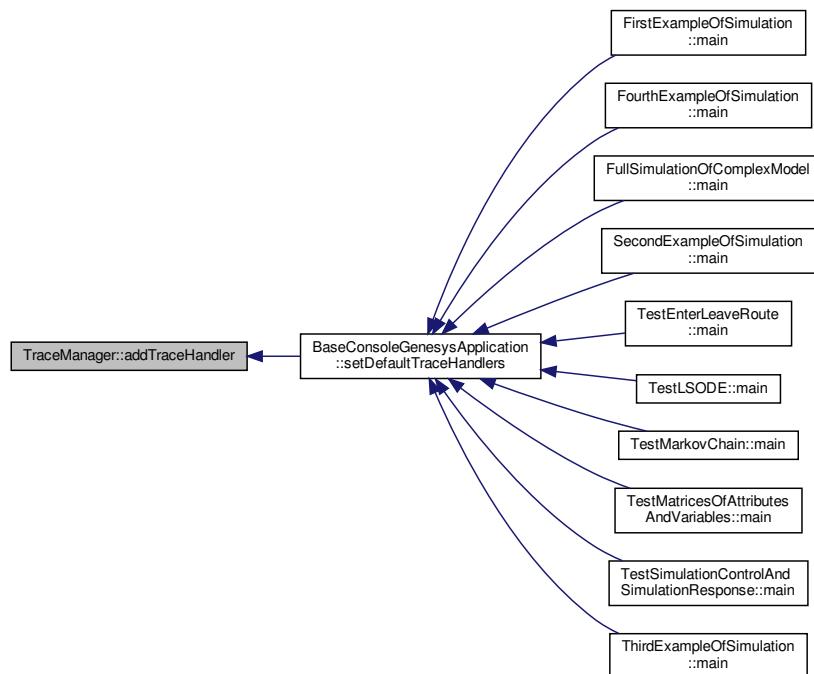
```
void TraceManager::addTraceHandler (
    traceListener traceListener )
```

Definition at line 48 of file [TraceManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.151.3.4 addTraceHandler() [2/2]

```
template<typename Class >
void TraceManager::addTraceHandler (
    Class * object,
    void(Class::*)(TraceEvent) function )
```

Definition at line 170 of file [TraceManager.h](#).

6.151.3.5 addTraceReportHandler() [1/2]

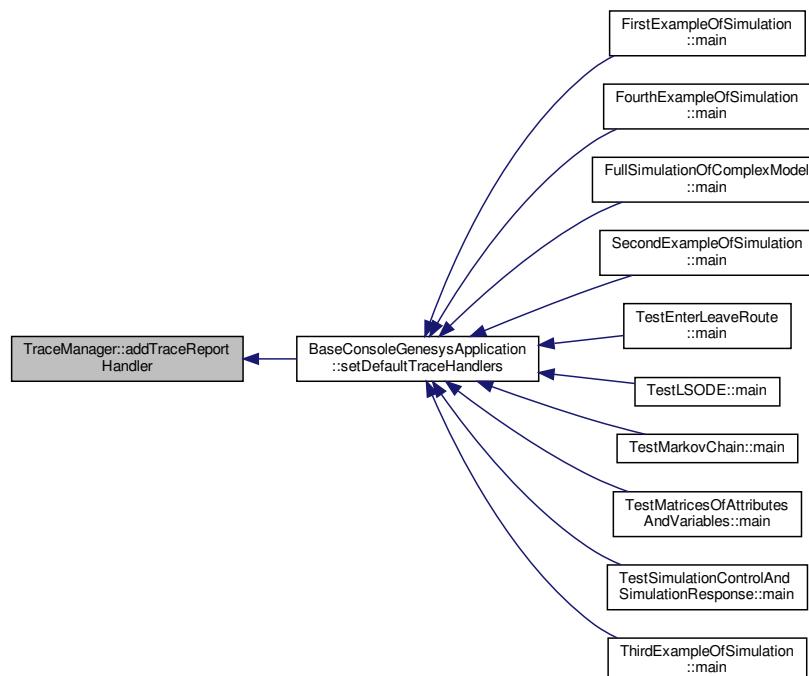
```
void TraceManager::addTraceReportHandler (
    traceListener traceReportListener )
```

Definition at line 64 of file [TraceManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.151.3.6 addTraceReportHandler() [2/2]

```
template<typename Class >
void TraceManager::addTraceReportHandler (
    Class * object,
    void(Class::*)(TraceEvent) function )
```

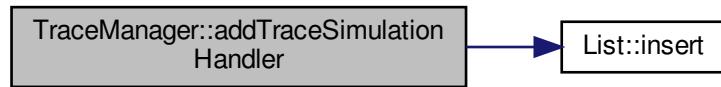
Definition at line 178 of file [TraceManager.h](#).

6.151.3.7 addTraceSimulationHandler() [1/2]

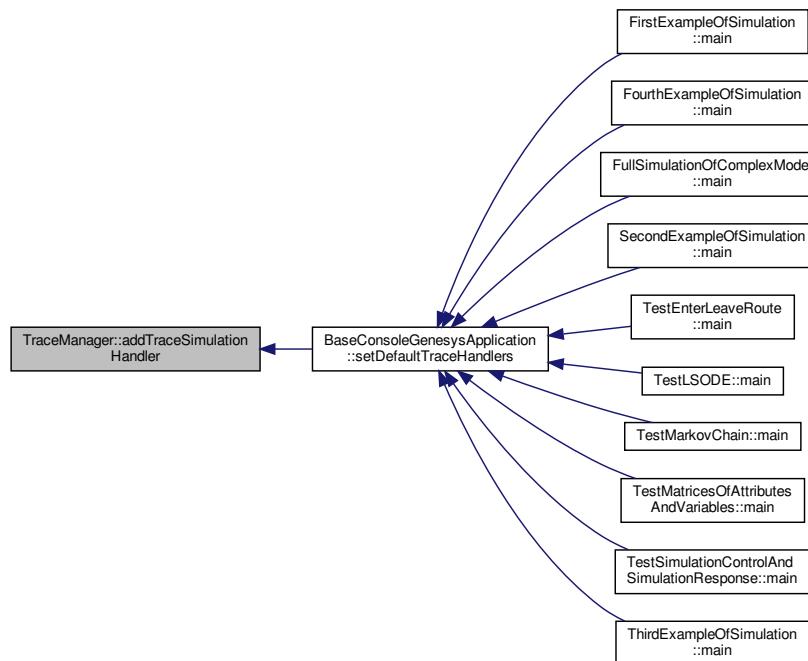
```
void TraceManager::addTraceSimulationHandler (
    traceSimulationListener traceSimulationListener )
```

Definition at line 56 of file [TraceManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.151.3.8 addTraceSimulationHandler() [2/2]

```
template<typename Class >
void TraceManager::addTraceSimulationHandler (
    Class * object,
    void(Class::*)(TraceSimulationEvent) function )
```

Definition at line 182 of file [TraceManager.h](#).

6.151.3.9 errorMessages()

```
List< std::string > * TraceManager::errorMessages ( ) const
```

Definition at line 136 of file [TraceManager.cpp](#).

6.151.3.10 parentSimulator()

```
Simulator * TraceManager::parentSimulator ( ) const
```

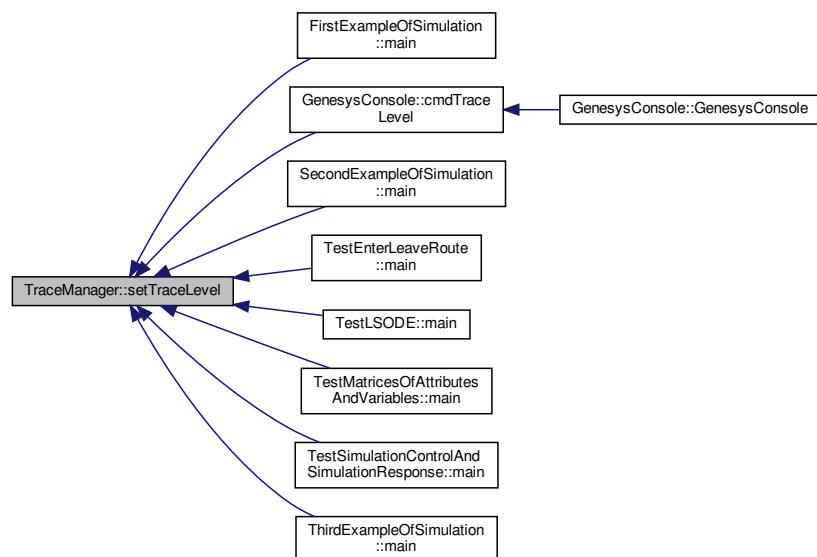
Definition at line 31 of file [TraceManager.cpp](#).

6.151.3.11 setTraceLevel()

```
void TraceManager::setTraceLevel ( Util::TraceLevel _traceLevel )
```

Definition at line 23 of file [TraceManager.cpp](#).

Here is the caller graph for this function:



6.151.3.12 trace() [1/2]

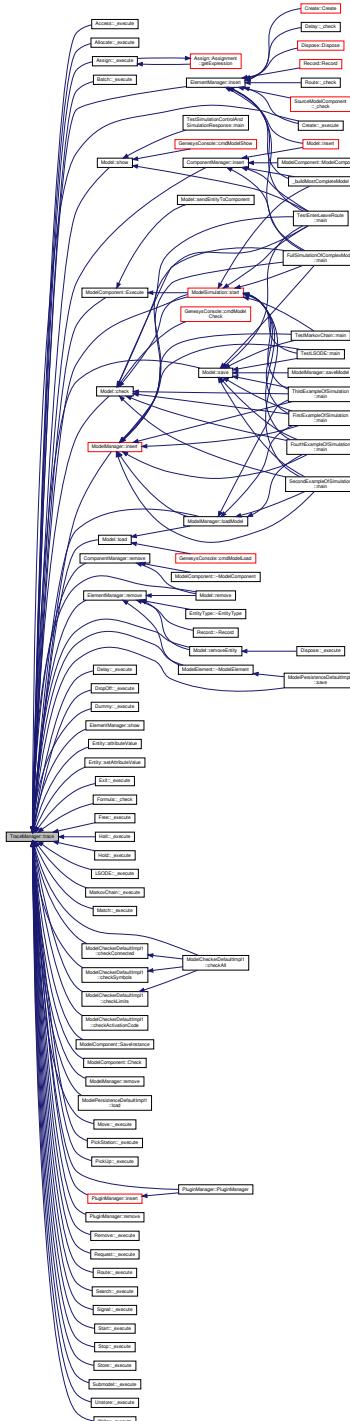
```
void TraceManager::trace (
```

`Util::TraceLevel level,`

```
        std::string text )
```

Definition at line 68 of file [TraceManager.cpp](#).

Here is the caller graph for this function:



6.151.3.13 trace() [2/2]

```
void TraceManager::trace (
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::componentInternal )
```

Definition at line 72 of file [TraceManager.cpp](#).

Here is the call graph for this function:

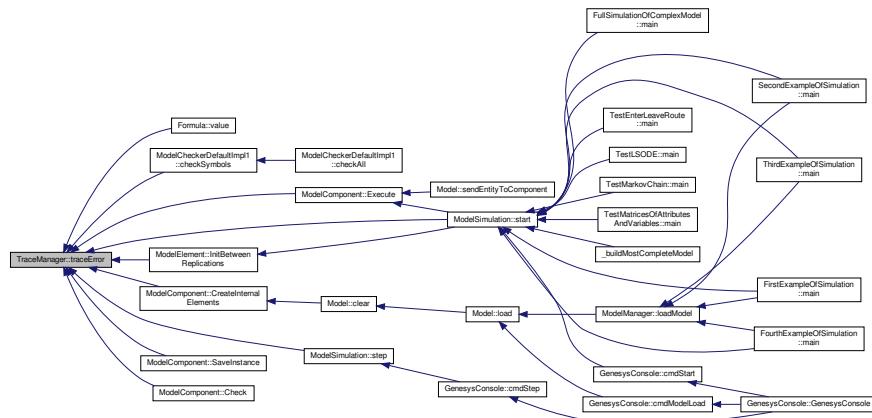


6.151.3.14 traceError() [1/2]

```
void TraceManager::traceError (
    std::exception e,
    std::string text )
```

Definition at line 87 of file [TraceManager.cpp](#).

Here is the caller graph for this function:



6.151.3.15 traceError() [2/2]

```
void TraceManager::traceError (
    std::string text,
    std::exception e )
```

Definition at line 91 of file [TraceManager.cpp](#).

Here is the call graph for this function:



6.151.3.16 traceLevel()

```
Util::TraceLevel TraceManager::traceLevel () const
```

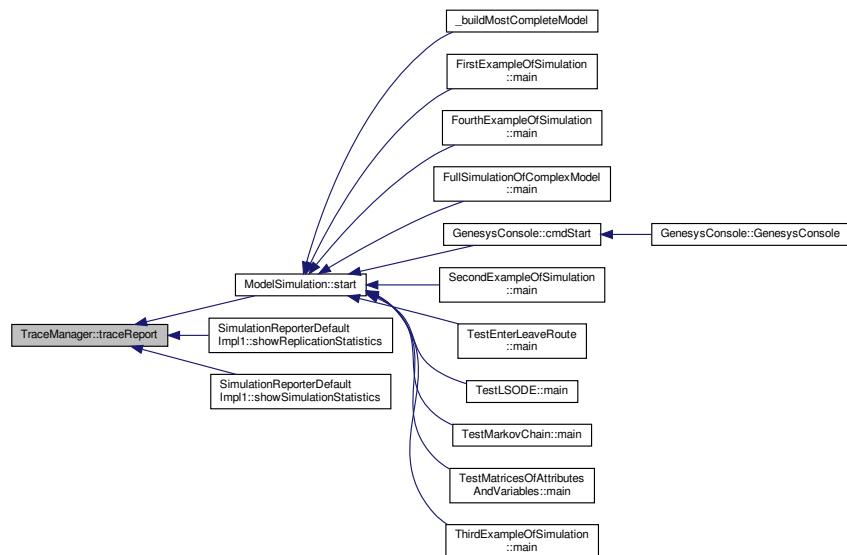
Definition at line 27 of file [TraceManager.cpp](#).

6.151.3.17 traceReport() [1/2]

```
void TraceManager::traceReport (
    Util::TraceLevel level,
    std::string text )
```

Definition at line 119 of file [TraceManager.cpp](#).

Here is the caller graph for this function:

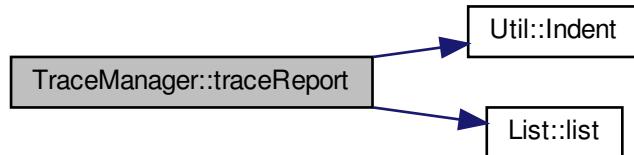


6.151.3.18 traceReport() [2/2]

```
void TraceManager::traceReport (
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::report )
```

Definition at line 123 of file [TraceManager.cpp](#).

Here is the call graph for this function:

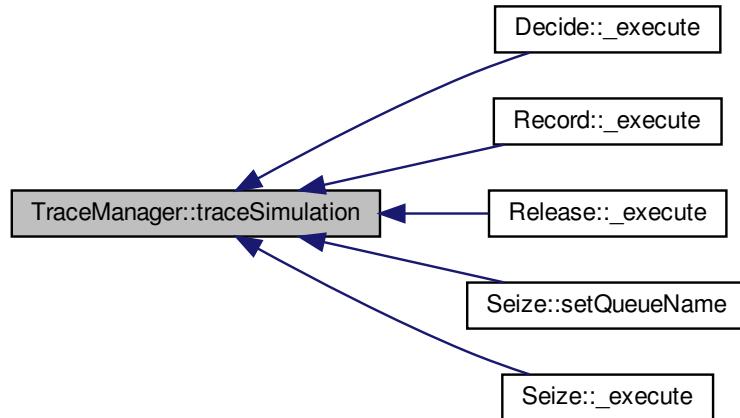


6.151.3.19 traceSimulation() [1/2]

```
void TraceManager::traceSimulation (
    Util::TraceLevel level,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text )
```

Definition at line 102 of file [TraceManager.cpp](#).

Here is the caller graph for this function:

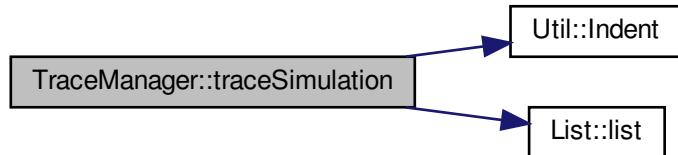


6.151.3.20 traceSimulation() [2/2]

```
void TraceManager::traceSimulation (
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text,
    Util::TraceLevel level = Util::TraceLevel::componentInternal )
```

Definition at line 106 of file [TraceManager.cpp](#).

Here is the call graph for this function:



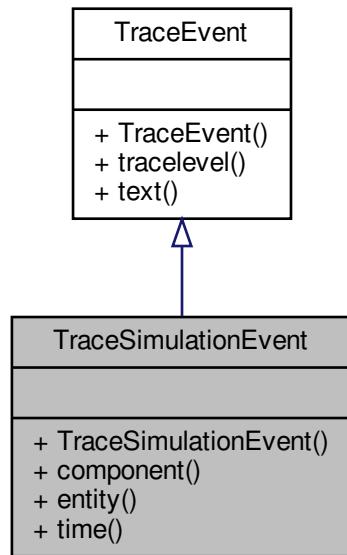
The documentation for this class was generated from the following files:

- [TraceManager.h](#)
- [TraceManager.cpp](#)

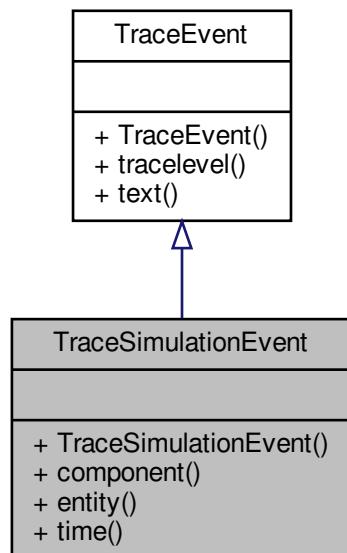
6.152 TraceSimulationEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationEvent:



Collaboration diagram for TraceSimulationEvent:



Public Member Functions

- `TraceSimulationEvent (Util::TraceLevel level, double time, Entity *entity, ModelComponent *component, std::string text)`
- `ModelComponent * component () const`
- `Entity * entity () const`
- `double time () const`

6.152.1 Detailed Description

Definition at line 58 of file [TraceManager.h](#).

6.152.2 Constructor & Destructor Documentation

6.152.2.1 TraceSimulationEvent()

```
TraceSimulationEvent::TraceSimulationEvent (
    Util::TraceLevel level,
    double time,
    Entity * entity,
    ModelComponent * component,
    std::string text ) [inline]
```

Definition at line 61 of file [TraceManager.h](#).

6.152.3 Member Function Documentation

6.152.3.1 component()

```
ModelComponent* TraceSimulationEvent::component () const [inline]
```

Definition at line 67 of file [TraceManager.h](#).

6.152.3.2 entity()

```
Entity* TraceSimulationEvent::entity () const [inline]
```

Definition at line 71 of file [TraceManager.h](#).

6.152.3.3 time()

```
double TraceSimulationEvent::time ( ) const [inline]
```

Definition at line 75 of file [TraceManager.h](#).

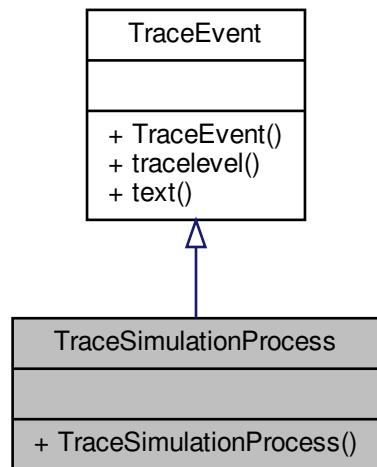
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

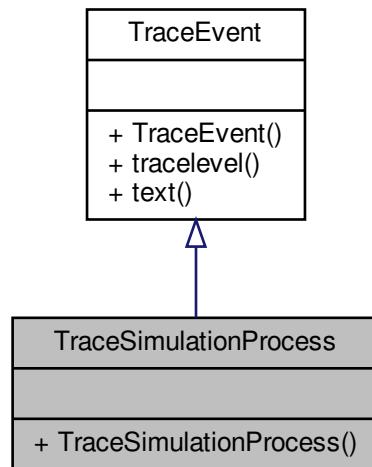
6.153 TraceSimulationProcess Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationProcess:



Collaboration diagram for TraceSimulationProcess:



Public Member Functions

- [TraceSimulationProcess \(Util::TraceLevel level, std::string text\)](#)

6.153.1 Detailed Description

Events related to simulation "process" (usually process analyser), associated to entire replication or simulation events (begin/end/pause of replication/simulation)

Todo : CLASS NOT FULLY IMPLEMENTED (to be implemented for process analyser)

Definition at line [89](#) of file [TraceManager.h](#).

6.153.2 Constructor & Destructor Documentation

6.153.2.1 TraceSimulationProcess()

```

TraceSimulationProcess::TraceSimulationProcess (
    Util::TraceLevel level,
    std::string text ) [inline]
  
```

Definition at line [92](#) of file [TraceManager.h](#).

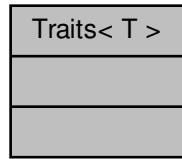
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

6.154 Traits< T > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< T >:



6.154.1 Detailed Description

```
template<typename T>
struct Traits< T >
```

Definition at line 71 of file [Traits.h](#).

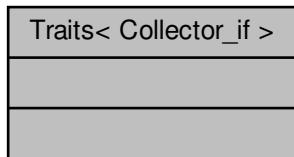
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.155 Traits< Collector_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Collector_if >:



Public Types

- [typedef CollectorDatafileDefaultImpl1 Implementation](#)

6.155.1 Detailed Description

```
template<>
struct Traits< Collector_if >
```

Definition at line 137 of file [Traits.h](#).

6.155.2 Member Typedef Documentation

6.155.2.1 Implementation

```
typedef CollectorDatafileDefaultImpl1 Traits< Collector_if >::Implementation
```

Definition at line 138 of file [Traits.h](#).

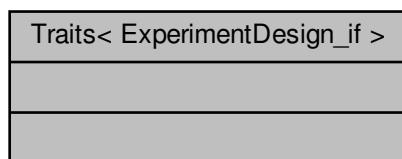
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.156 Traits< ExperimentDesign_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ExperimentDesign_if >:



Public Types

- [typedef ExperimentDesignDefaultImpl1 Implementation](#)

6.156.1 Detailed Description

```
template<>
struct Traits< ExperimentDesign_if >
```

Definition at line 171 of file [Traits.h](#).

6.156.2 Member Typedef Documentation

6.156.2.1 Implementation

```
typedef ExperimentDesignDefaultImpl1 Traits< ExperimentDesign_if >::Implementation
```

Definition at line 172 of file [Traits.h](#).

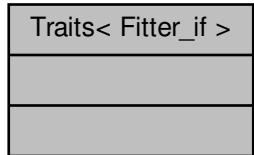
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.157 Traits< Fitter_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Fitter_if >:



Public Types

- [typedef FitterDefaultImpl1 Implementation](#)

6.157.1 Detailed Description

```
template<>
struct Traits< Fitter_if >
```

Definition at line 158 of file [Traits.h](#).

6.157.2 Member Typedef Documentation

6.157.2.1 Implementation

```
typedef FitterDefaultImpl1 Traits< Fitter_if >::Implementation
```

Definition at line 159 of file [Traits.h](#).

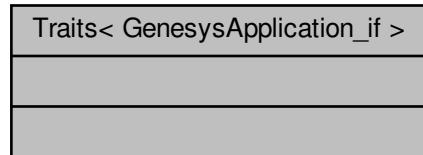
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.158 Traits< GenesysApplication_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< GenesysApplication_if >:



Public Types

- [typedef FirstExampleOfSimulation Application](#)

6.158.1 Detailed Description

```
template<>
struct Traits< GenesysApplication_if >
```

Definition at line 78 of file [Traits.h](#).

6.158.2 Member Typedef Documentation

6.158.2.1 Application

```
typedef FirstExampleOfSimulation Traits< GenesysApplication_if >::Application
```

Definition at line 82 of file [Traits.h](#).

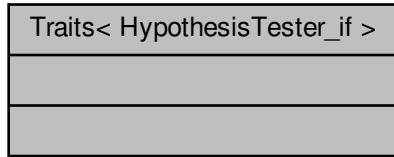
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.159 Traits< HypothesisTester_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< HypothesisTester_if >:



Public Types

- [typedef IntegratorDefaultImpl1 IntegratorImplementation](#)
- [typedef HypothesisTesterDefaultImpl1 Implementation](#)

6.159.1 Detailed Description

```
template<>
struct Traits< HypothesisTester_if >
```

Definition at line 162 of file [Traits.h](#).

6.159.2 Member Typedef Documentation

6.159.2.1 Implementation

```
typedef HypothesisTesterDefaultImpl1 Traits< HypothesisTester_if >::Implementation
```

Definition at line 164 of file [Traits.h](#).

6.159.2.2 IntegratorImplementation

```
typedef IntegratorDefaultImpl1 Traits< HypothesisTester_if >::IntegratorImplementation
```

Definition at line 163 of file [Traits.h](#).

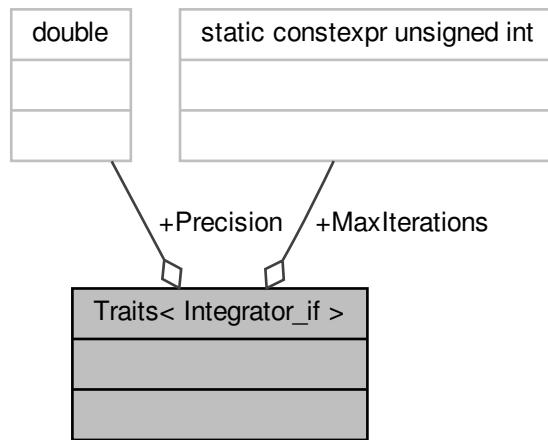
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.160 Traits< Integrator_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Integrator_if >:



Public Types

- `typedef IntegratorDefaultImpl1 Implementation`

Static Public Attributes

- `static constexpr unsigned int MaxIterations = 1e3`
- `static constexpr double Precision = 1e-9`

6.160.1 Detailed Description

```
template<>
struct Traits< Integrator_if >
```

Definition at line 147 of file [Traits.h](#).

6.160.2 Member Typedef Documentation

6.160.2.1 Implementation

```
typedef IntegratorDefaultImpl1 Traits< Integrator_if >::Implementation
```

Definition at line 148 of file [Traits.h](#).

6.160.3 Member Data Documentation

6.160.3.1 MaxIterations

```
constexpr unsigned int Traits< Integrator_if >::MaxIterations = 1e3 [static]
```

Definition at line 149 of file [Traits.h](#).

6.160.3.2 Precision

```
constexpr double Traits< Integrator_if >::Precision = 1e-9 [static]
```

Definition at line 150 of file [Traits.h](#).

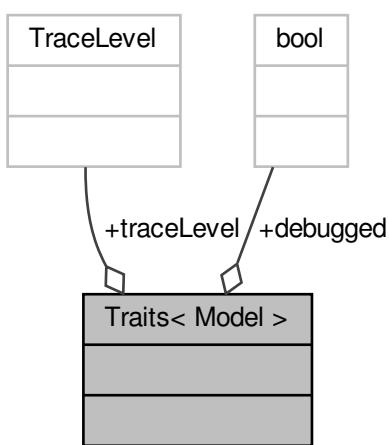
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.161 Traits< Model > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Model >:



Static Public Attributes

- static const bool `debugged` = true
- static const `Util::TraceLevel traceLevel` = `Util::TraceLevel::modelSimulationEvent`

6.161.1 Detailed Description

```
template<>
struct Traits< Model >
```

Definition at line 111 of file [Traits.h](#).

6.161.2 Member Data Documentation

6.161.2.1 `debugged`

```
const bool Traits< Model >::debugged = true [static]
```

Definition at line 112 of file [Traits.h](#).

6.161.2.2 `traceLevel`

```
const Util::TraceLevel Traits< Model >::traceLevel = Util::TraceLevel::modelSimulationEvent
[static]
```

Definition at line 113 of file [Traits.h](#).

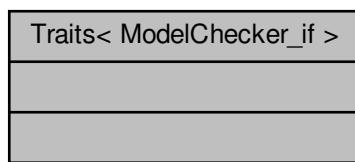
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.162 Traits< ModelChecker_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelChecker_if >:



Public Types

- [typedef ModelCheckerDefaultImpl1 Implementation](#)

6.162.1 Detailed Description

```
template<>
struct Traits< ModelChecker_if >
```

Definition at line 124 of file [Traits.h](#).

6.162.2 Member Typedef Documentation

6.162.2.1 Implementation

```
typedef ModelCheckerDefaultImpl1 Traits< ModelChecker_if >::Implementation
```

Definition at line 125 of file [Traits.h](#).

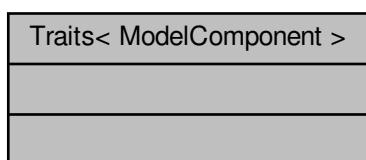
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.163 Traits< ModelComponent > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelComponent >:



Public Types

- [typedef StatisticsDefaultImpl1 StatisticsCollector_StatisticsImplementation](#)
- [typedef CollectorDefaultImpl1 StatisticsCollector_CollectorImplementation](#)

6.163.1 Detailed Description

```
template<>
struct Traits< ModelComponent >
```

Definition at line 128 of file [Traits.h](#).

6.163.2 Member Typedef Documentation

6.163.2.1 StatisticsCollector_CollectorImplementation

```
typedef CollectorDefaultImpl1 Traits< ModelComponent >::StatisticsCollector_CollectorImplementation
```

Definition at line 130 of file [Traits.h](#).

6.163.2.2 StatisticsCollector_StatisticsImplementation

```
typedef StatisticsDefaultImpl1 Traits< ModelComponent >::StatisticsCollector_StatisticsImplementation
```

Definition at line 129 of file [Traits.h](#).

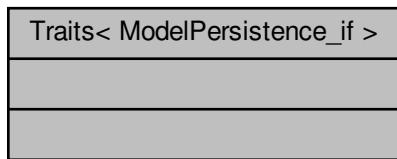
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.164 Traits< ModelPersistence_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ModelPersistence_if >:



Public Types

- [typedef ModelPersistenceDefaultImpl1 Implementation](#)

6.164.1 Detailed Description

```
template<>
struct Traits< ModelPersistence_if >
```

Definition at line 116 of file [Traits.h](#).

6.164.2 Member Typedef Documentation

6.164.2.1 Implementation

```
typedef ModelPersistenceDefaultImpl1 Traits< ModelPersistence_if >::Implementation
```

Definition at line 117 of file [Traits.h](#).

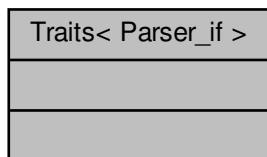
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.165 Traits< Parser_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Parser_if >:



Public Types

- [typedef ParserDefaultImpl2 Implementation](#)

6.165.1 Detailed Description

```
template<>
struct Traits< Parser_if >
```

Definition at line 102 of file [Traits.h](#).

6.165.2 Member Typedef Documentation

6.165.2.1 Implementation

```
typedef ParserDefaultImpl2 Traits< Parser_if >::Implementation
```

Definition at line 103 of file [Traits.h](#).

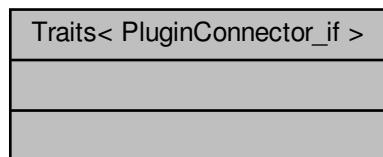
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.166 Traits< PluginConnector_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< PluginConnector_if >:



Public Types

- [typedef PluginConnectorDummyImpl1 Implementation](#)

6.166.1 Detailed Description

```
template<>
struct Traits< PluginConnector_if >
```

Definition at line 98 of file [Traits.h](#).

6.166.2 Member Typedef Documentation

6.166.2.1 Implementation

```
typedef PluginConnectorDummyImpl1 Traits< PluginConnector_if >::Implementation
```

Definition at line 99 of file [Traits.h](#).

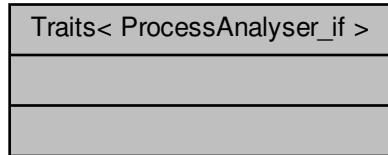
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.167 Traits< ProcessAnalyser_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< ProcessAnalyser_if >:



Public Types

- [typedef ProcessAnalyserDefaultImpl1 Implementation](#)

6.167.1 Detailed Description

```
template<>
struct Traits< ProcessAnalyser_if >
```

Definition at line 175 of file [Traits.h](#).

6.167.2 Member Typedef Documentation

6.167.2.1 Implementation

```
typedef ProcessAnalyserDefaultImpl1 Traits< ProcessAnalyser_if >::Implementation
```

Definition at line 176 of file [Traits.h](#).

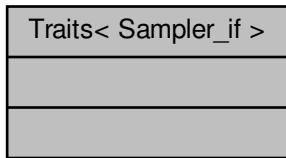
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.168 Traits< Sampler_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Sampler_if >:



Public Types

- [typedef SamplerDefaultImpl1 Implementation](#)
- [typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters](#)

6.168.1 Detailed Description

```
template<>
struct Traits< Sampler_if >
```

Definition at line 153 of file [Traits.h](#).

6.168.2 Member Typedef Documentation

6.168.2.1 Implementation

```
typedef SamplerDefaultImpl1 Traits< Sampler_if >::Implementation
```

Definition at line 154 of file [Traits.h](#).

6.168.2.2 Parameters

```
typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Traits< Sampler_if >::Parameters
```

Definition at line 155 of file [Traits.h](#).

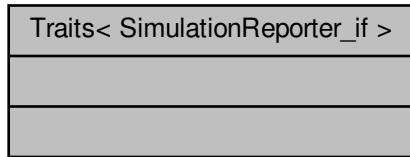
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.169 Traits< SimulationReporter_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< SimulationReporter_if >:



Public Types

- [typedef SimulationReporterDefaultImpl1 Implementation](#)

6.169.1 Detailed Description

```
template<>
struct Traits< SimulationReporter_if >
```

Definition at line 120 of file [Traits.h](#).

6.169.2 Member Typedef Documentation

6.169.2.1 Implementation

```
typedef SimulationReporterDefaultImpl1 Traits< SimulationReporter_if >::Implementation
```

Definition at line 121 of file [Traits.h](#).

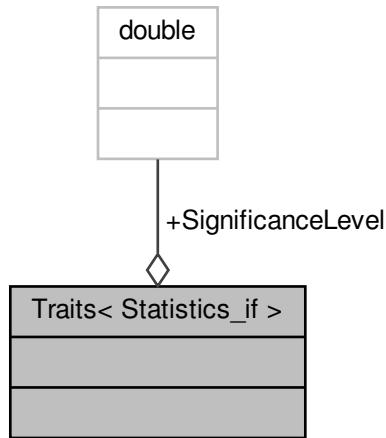
The documentation for this struct was generated from the following file:

- [Traits.h](#)

6.170 Traits< Statistics_if > Struct Template Reference

```
#include <Traits.h>
```

Collaboration diagram for Traits< Statistics_if >:



Public Types

- `typedef StatisticsDefaultImpl1 Implementation`
- `typedef CollectorDefaultImpl1 CollectorImplementation`

Static Public Attributes

- `static constexpr double SignificanceLevel = 0.05`

6.170.1 Detailed Description

```
template<>
struct Traits< Statistics_if >
```

Definition at line 141 of file [Traits.h](#).

6.170.2 Member Typedef Documentation

6.170.2.1 CollectorImplementation

```
typedef CollectorDefaultImpl1 Traits< Statistics_if >::CollectorImplementation
```

Definition at line 143 of file [Traits.h](#).

6.170.2.2 Implementation

```
typedef StatisticsDefaultImpl1 Traits< Statistics_if >::Implementation
```

Definition at line 142 of file [Traits.h](#).

6.170.3 Member Data Documentation

6.170.3.1 SignificanceLevel

```
constexpr double Traits< Statistics_if >::SignificanceLevel = 0.05 [static]
```

Definition at line 144 of file [Traits.h](#).

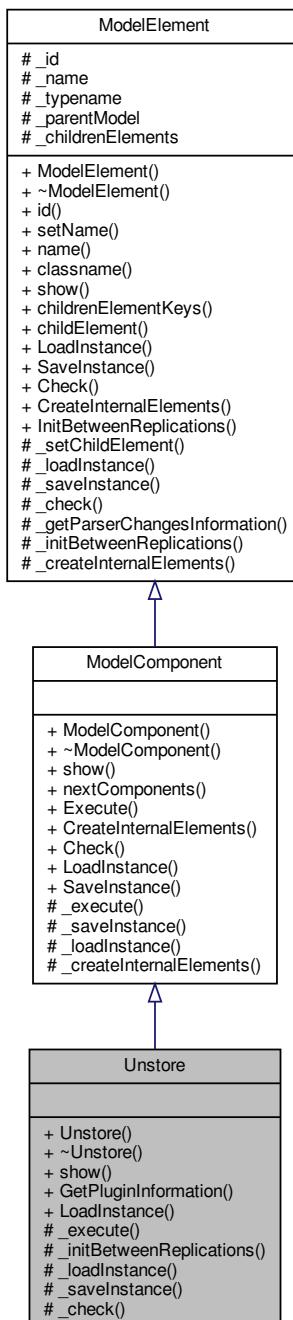
The documentation for this struct was generated from the following file:

- [Traits.h](#)

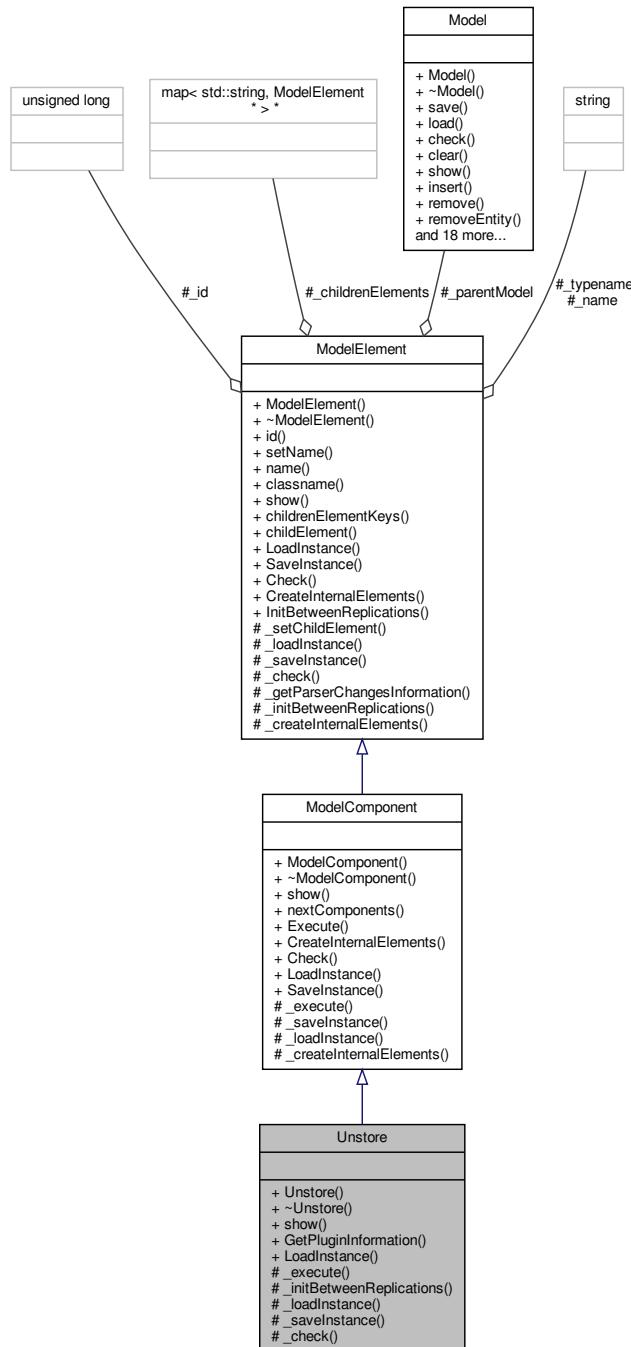
6.171 Unstore Class Reference

```
#include <Unstore.h>
```

Inheritance diagram for Unstore:



Collaboration diagram for Unstore:



Public Member Functions

- [Unstore \(Model *model, std::string name=""\)](#)
- virtual [~Unstore \(\)=default](#)
- virtual std::string [show \(\)](#)

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.171.1 Detailed Description

Unstore module DESCRIPTION The **Unstore** module removes an entity from storage. When an entity arrives at the **Unstore** module, the storage specified is decreased and the entity immediately moves to the next module in the model. TYPICAL USES Removing the entity from an animation location when processing is complete Tracking the number of customers within a grocery store (unstore upon exit) PROMPTS Prompt Description Name Unique module identifier displayed on the module shape. Type Method of specifying the storage name as a **Storage**, **Set**, **Attribute**, or Expression. Default will remove an entity from the last storage that it entered. **Storage** Name Name of the storage to which the entity will be added. Applies only when the Type is **Storage**. **Set** Name Name of the storage set from which the storage is to be selected. Applies only when the Type is **Set**. **Set** Index Index into the defined storage set that contains the desired storage name. Applies only when the Type is **Set**. **Attribute** Name of the attribute whose value contains the storage. Applies only when the Type is **Attribute**. Expression Expression that is evaluated to the storage into which the entity is placed. Applies only when the Type is Expression.

Definition at line 45 of file [Unstore.h](#).

6.171.2 Constructor & Destructor Documentation

6.171.2.1 Unstore()

```
Unstore::Unstore (
    Model * model,
    std::string name = "" )
```

Definition at line 17 of file [Unstore.cpp](#).

Here is the caller graph for this function:



6.171.2.2 ~Unstore()

```
virtual Unstore::~Unstore ( ) [virtual], [default]
```

6.171.3 Member Function Documentation

6.171.3.1 _check()

```
bool Unstore::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 57 of file [Unstore.cpp](#).

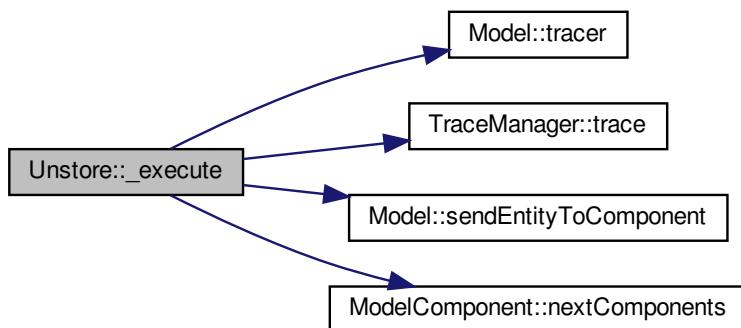
6.171.3.2 _execute()

```
void Unstore::_execute (
    Entity * entity ) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 35 of file [Unstore.cpp](#).

Here is the call graph for this function:



6.171.3.3 `_initBetweenReplications()`

```
void Unstore::_initBetweenReplications ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line [48](#) of file [Unstore.cpp](#).

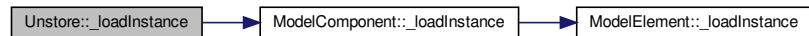
6.171.3.4 `_loadInstance()`

```
bool Unstore::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

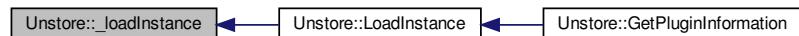
Reimplemented from [ModelComponent](#).

Definition at line [40](#) of file [Unstore.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



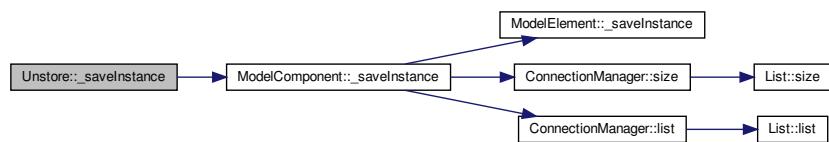
6.171.3.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Unstore::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [51](#) of file [Unstore.cpp](#).

Here is the call graph for this function:



6.171.3.6 GetPluginInformation()

```
PluginInformation * Unstore::GetPluginInformation ( ) [static]
```

Definition at line 63 of file [Unstore.cpp](#).

Here is the call graph for this function:

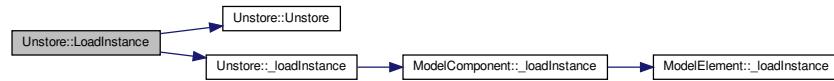


6.171.3.7 LoadInstance()

```
ModelComponent * Unstore::LoadInstance (
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 25 of file [Unstore.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



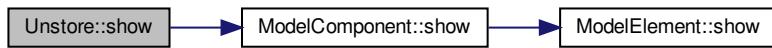
6.171.3.8 show()

```
std::string Unstore::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line [21](#) of file [Unstore.cpp](#).

Here is the call graph for this function:



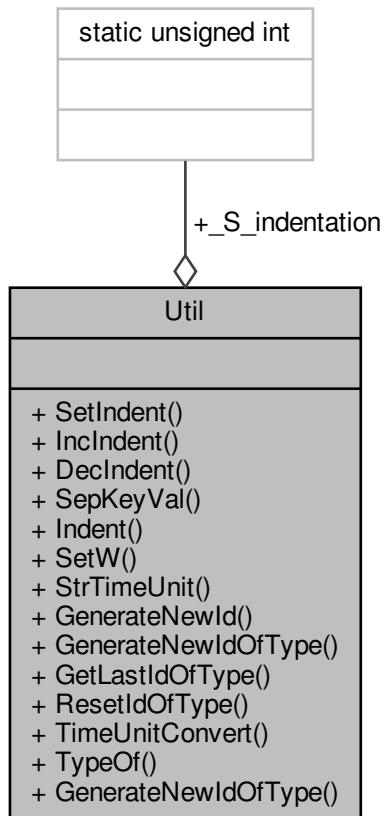
The documentation for this class was generated from the following files:

- [Unstore.h](#)
- [Unstore.cpp](#)

6.172 Util Class Reference

```
#include <Util.h>
```

Collaboration diagram for Util:



Public Types

- enum `TimeUnit` : int {
 `TimeUnit::picosecond` = 1, `TimeUnit::nanosecond` = 2, `TimeUnit::microsecond` = 3, `TimeUnit::milisecond` = 4, `TimeUnit::second` = 5, `TimeUnit::minute` = 6, `TimeUnit::hour` = 7, `TimeUnit::day` = 8, `TimeUnit::week` = 9 }
- enum `TraceLevel` : int {
 `TraceLevel::noTraces` = 0, `TraceLevel::errorFatal` = 1, `TraceLevel::errorRecover` = 2, `TraceLevel::warning` = 3, `TraceLevel::report` = 4, `TraceLevel::simulatorResult` = 5, `TraceLevel::toolResult` = 6, `TraceLevel::modelResult` = 7, `TraceLevel::componentResult` = 8, `TraceLevel::elementResult` = 9, `TraceLevel::modelSimulationEvent` = 10, `TraceLevel::componentArrival` = 11, `TraceLevel::simulatorInternal` = 12, `TraceLevel::toolInternal` = 13, `TraceLevel::modelSimulationInternal` = 14, `TraceLevel::modelInternal` = 14, `TraceLevel::componentInternal` = 15, `TraceLevel::elementInternal` = 16, `TraceLevel::simulatorDetailed` = 17, `TraceLevel::toolDetailed` = 17, `TraceLevel::modelSimulationDetailed` = 18, `TraceLevel::modelDetailed` = 19, `TraceLevel::componentDetailed` = 20, `TraceLevel::elementDetailed` = 21, `TraceLevel::everythingMostDetailed` = 30 }
- `typedef unsigned long identification`
- `typedef unsigned int rank`

Static Public Member Functions

- static void [SetIndent](#) (const unsigned short indent)
- static void [InIndent](#) ()
- static void [DeclIndent](#) ()
- static void [SepKeyVal](#) (std::string str, std::string *key, std::string *value)
- static std::string [Indent](#) ()
- static std::string [SetW](#) (std::string text, unsigned short width)
- static std::string [StrTimeUnit](#) ([Util::TimeUnit](#) timeUnit)
- static [Util::identification](#) [GenerateNewId](#) ()
- static [Util::identification](#) [GenerateNewIdOfType](#) (std::string objtype)
- static [Util::identification](#) [GetLastIdOfType](#) (std::string objtype)
- static void [ResetIdOfType](#) (std::string objtype)
- static double [TimeUnitConvert](#) ([Util::TimeUnit](#) timeUnit1, [Util::TimeUnit](#) timeUnit2)
- template<class T >
 static std::string [TypeOf](#) ()
- template<class T >
 static [Util::identification](#) [GenerateNewIdOfType](#) ()

Static Public Attributes

- static unsigned int [_S_indentation](#)

6.172.1 Detailed Description

Definition at line [63](#) of file [Util.h](#).

6.172.2 Member Typedef Documentation

6.172.2.1 [identification](#)

```
typedef unsigned long Util::identification
```

Definition at line [65](#) of file [Util.h](#).

6.172.2.2 [rank](#)

```
typedef unsigned int Util::rank
```

Definition at line [66](#) of file [Util.h](#).

6.172.3 Member Enumeration Documentation

6.172.3.1 [TimeUnit](#)

```
enum Util::TimeUnit : int [strong]
```

Enumerator

picosecond	
nanosecond	
microsecond	
milisecond	
second	
minute	
hour	
day	
week	

Definition at line 68 of file [Util.h](#).

6.172.3.2 TraceLevel

```
enum Util::TraceLevel : int [strong]
```

Enumerator

noTraces	
errorFatal	
errorRecover	
warning	
report	
simulatorResult	
toolResult	
modelResult	
componentResult	
elementResult	
modelSimulationEvent	
componentArrival	
simulatorInternal	
toolInternal	
modelSimulationInternal	
modelInternal	
componentInternal	
elementInternal	
simulatorDetailed	
toolDetailed	
modelSimulationDetailed	
modelDetailed	
componentDetailed	
elementDetailed	
everythingMostDetailed	

Definition at line 81 of file [Util.h](#).

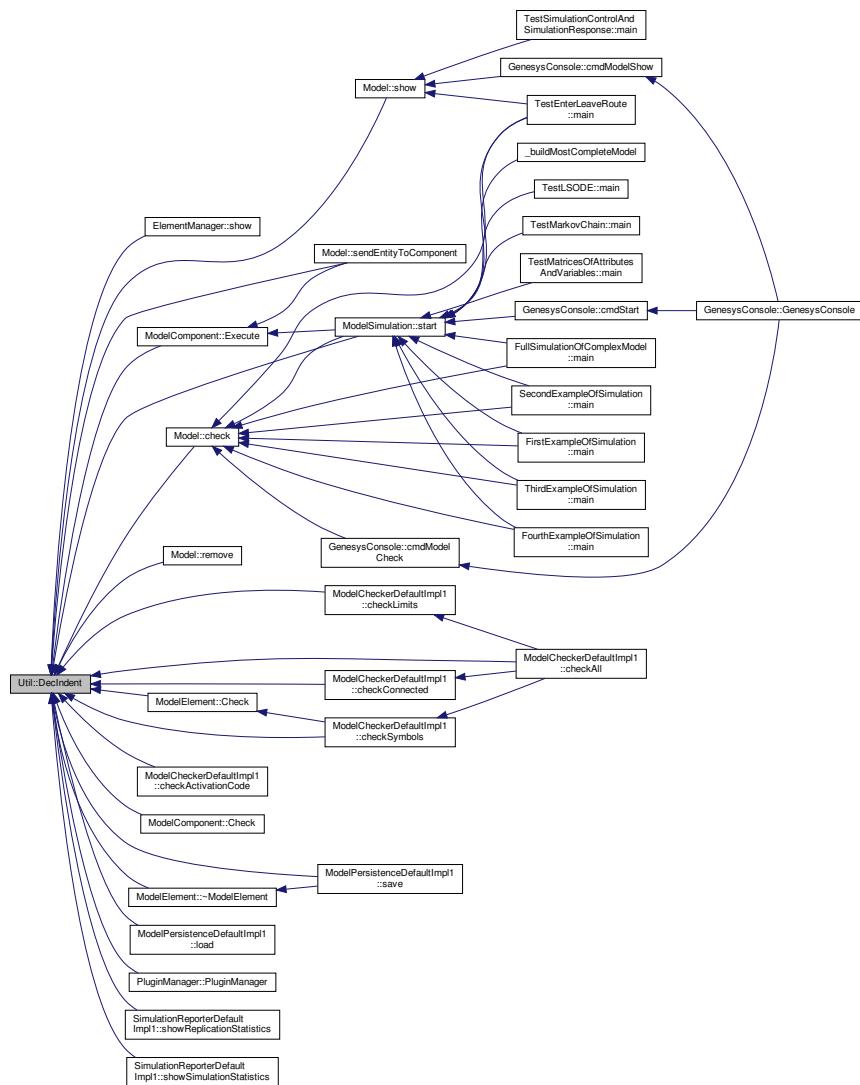
6.172.4 Member Function Documentation

6.172.4.1 Declndent()

```
void Util::DecIndent ( ) [static]
```

Definition at line 48 of file [Util.cpp](#).

Here is the caller graph for this function:



6.172.4.2 GenerateNewId()

```
Util::identification Util::GenerateNewId ( ) [static]
```

Definition at line 102 of file [Util.cpp](#).

Here is the caller graph for this function:



6.172.4.3 GenerateNewIdOfType() [1/2]

```
Util::identification Util::GenerateNewIdOfType (
    std::string objtype) [static]
```

Definition at line 107 of file [Util.cpp](#).

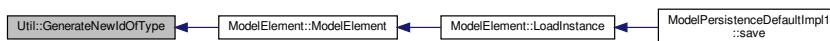
6.172.4.4 GenerateNewIdOfType() [2/2]

```
template<class T>
static Util::identification Util::GenerateNewIdOfType ( ) [inline], [static]
```

Every component or element has a unique ID for its class, but not unique for other classes. IDs are generated sequentially for each class.

Definition at line 154 of file [Util.h](#).

Here is the caller graph for this function:



6.172.4.5 GetLastIdOfType()

```
Util::identification Util::GetLastIdOfType (
    std::string objtype ) [static]
```

Definition at line 120 of file [Util.cpp](#).

Here is the caller graph for this function:

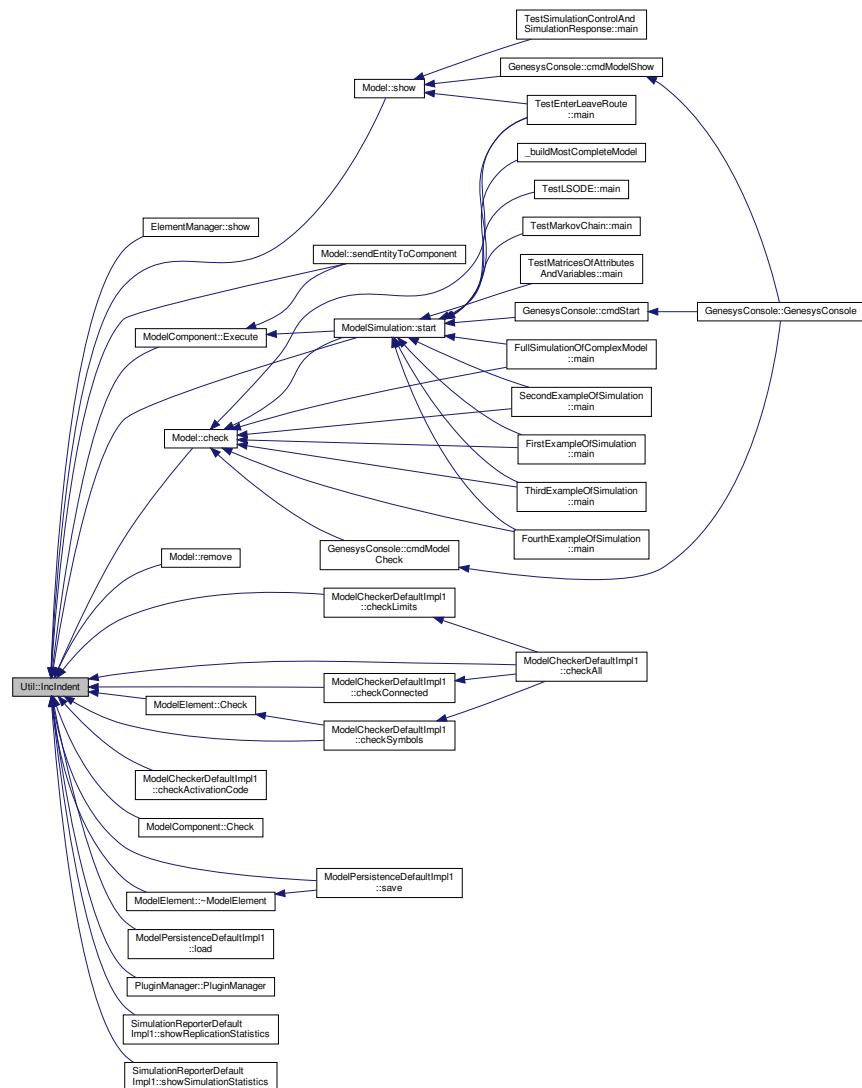


6.172.4.6 IncIndent()

```
void Util::IncIndent ( ) [static]
```

Definition at line 40 of file [Util.cpp](#).

Here is the caller graph for this function:

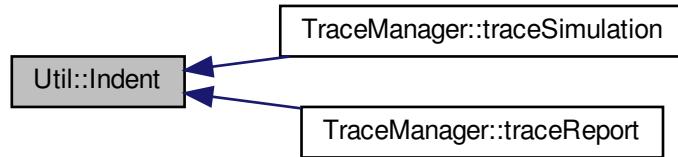


6.172.4.7 Indent()

```
std::string Util::Indent( ) [static]
```

Definition at line 73 of file [Util.cpp](#).

Here is the caller graph for this function:

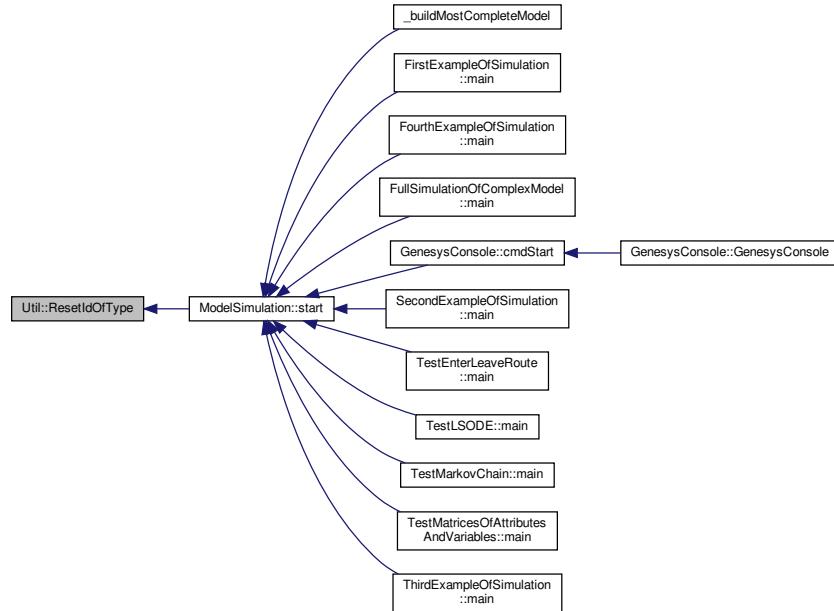


6.172.4.8 ResetIdOfType()

```
void Util::ResetIdOfType ( std::string objtype ) [static]
```

Definition at line 134 of file [Util.cpp](#).

Here is the caller graph for this function:



6.172.4.9 SepKeyVal()

```
void Util::SepKeyVal (
    std::string str,
    std::string * key,
    std::string * value ) [static]
```

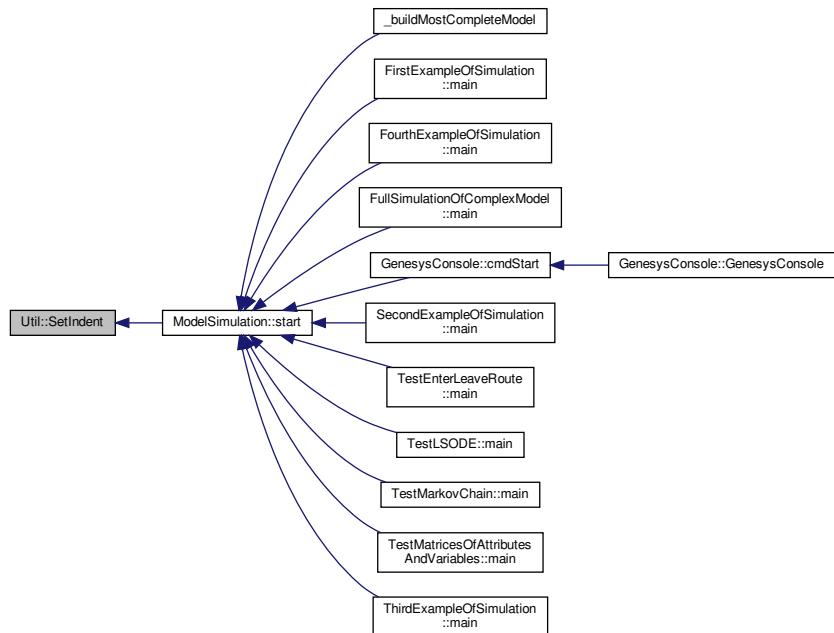
Definition at line 52 of file [Util.cpp](#).

6.172.4.10 SetIndent()

```
void Util::SetIndent (
    const unsigned short indent ) [static]
```

Definition at line 44 of file [Util.cpp](#).

Here is the caller graph for this function:

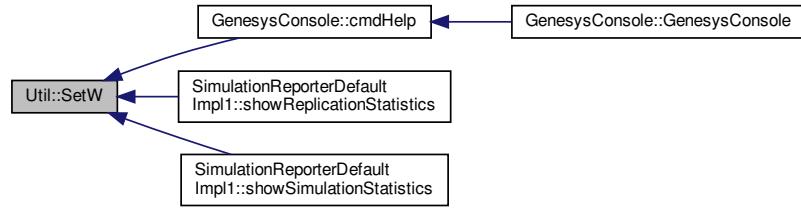


6.172.4.11 SetW()

```
std::string Util::SetW (
    std::string text,
    unsigned short width ) [static]
```

Definition at line 81 of file [Util.cpp](#).

Here is the caller graph for this function:

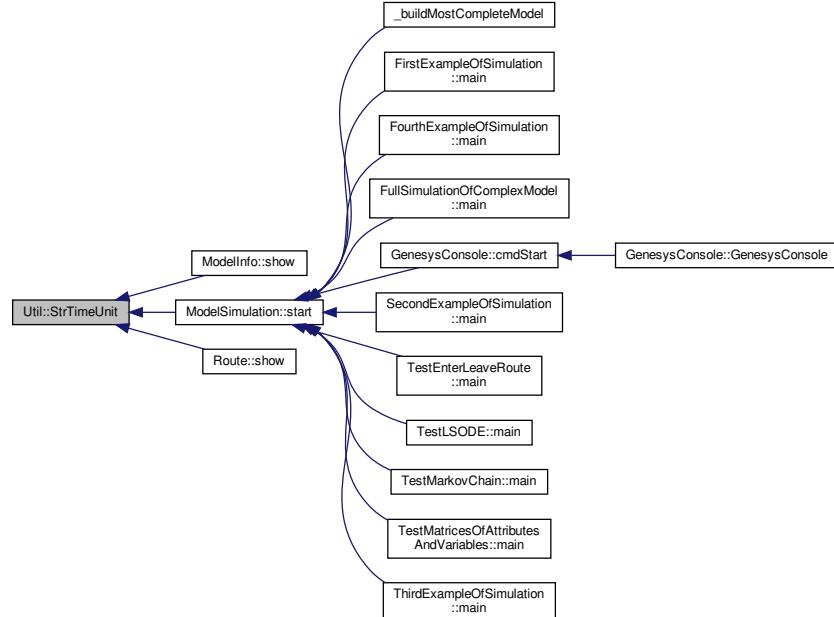


6.172.4.12 StrTimeUnit()

```
std::string Util::StrTimeUnit (
```

Definition at line 87 of file [Util.cpp](#).

Here is the caller graph for this function:

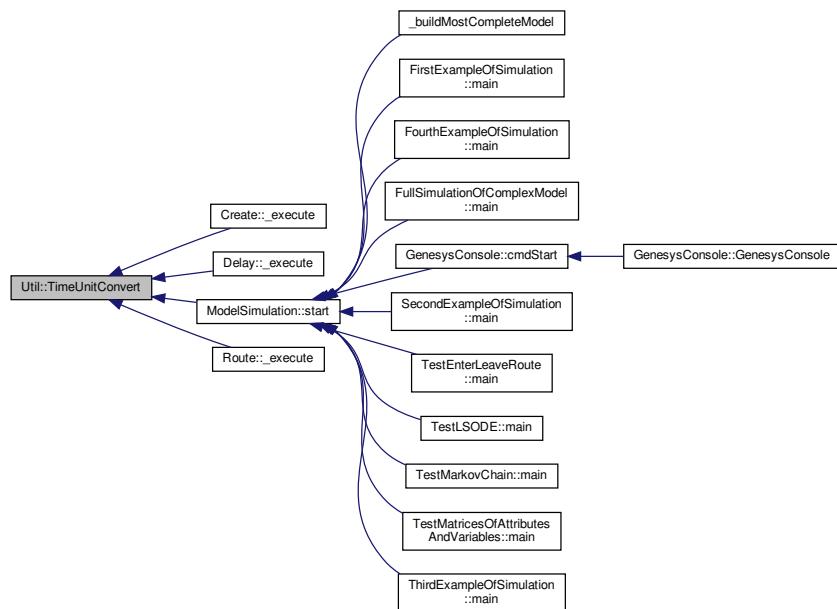


6.172.4.13 TimeUnitConvert()

```
double Util::TimeUnitConvert (
    Util::TimeUnit timeUnit1,
    Util::TimeUnit timeUnit2) [static]
```

Definition at line 144 of file [Util.cpp](#).

Here is the caller graph for this function:



6.172.4.14 TypeOf()

```
template<class T>
static std::string Util::TypeOf() [inline], [static]
```

Return the name of the class used as T.

Definition at line 136 of file [Util.h](#).

6.172.5 Member Data Documentation

6.172.5.1 _S_indentation

```
unsigned int Util::_S_indentation [static]
```

Definition at line 114 of file [Util.h](#).

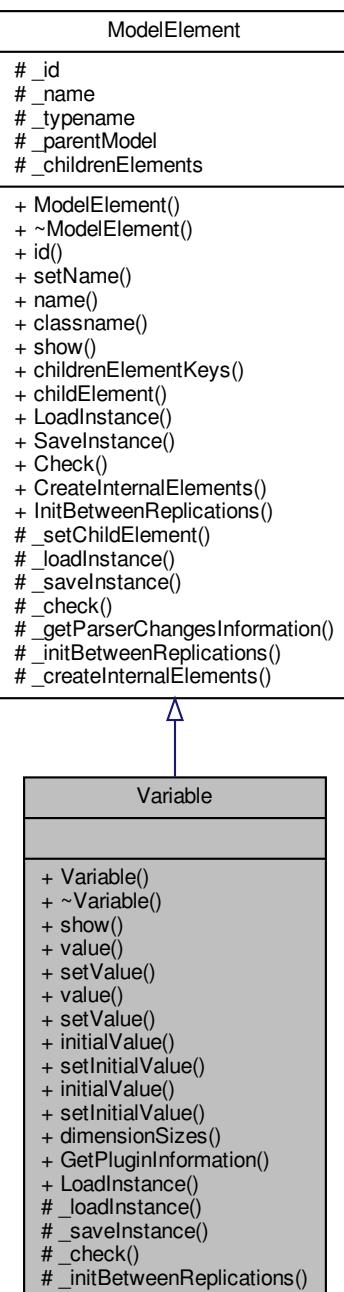
The documentation for this class was generated from the following files:

- [Util.h](#)
- [Util.cpp](#)

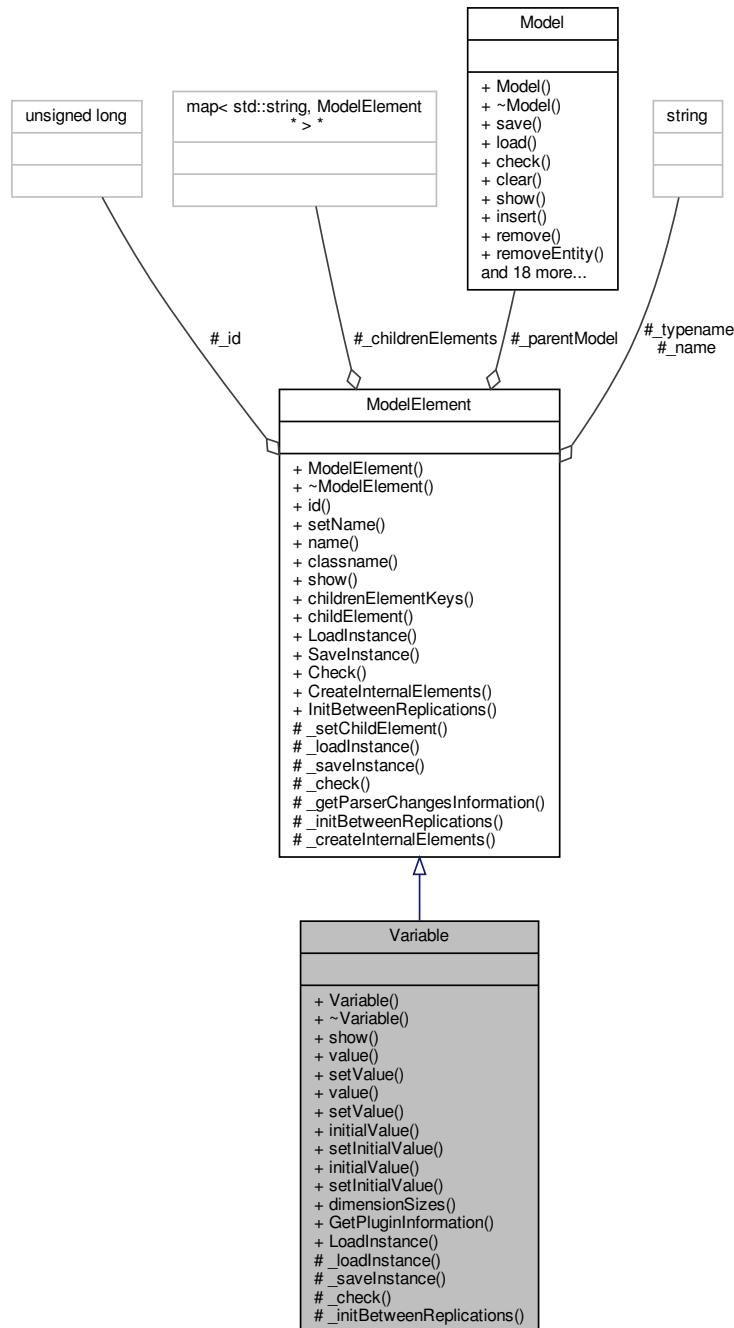
6.173 Variable Class Reference

```
#include <Variable.h>
```

Inheritance diagram for Variable:



Collaboration diagram for Variable:



Public Member Functions

- `Variable (Model *model, std::string name = "")`
- `virtual ~Variable ()=default`
- `virtual std::string show ()`
- `double value ()`
- `void setValue (double value)`

- double `value` (std::string index)
- void `setValue` (std::string index, double `value`)
- double `initialValue` ()
- void `setInitialValue` (double `value`)
- double `initialValue` (std::string index)
- void `setInitialValue` (std::string index, double `value`)
- `List< unsigned int > * dimensionSizes () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelElement * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`
- virtual void `_initBetweenReplications ()`

Additional Inherited Members

6.173.1 Detailed Description

Variable module DESCRIPTION This data module is used to define a variable's dimension and values. You can reference variables in other modules (for example, the **Decide** module), reassign new values to variables with the **Assign** module, and use variables in any expression. You can use an external data file to specify variable values, and you can specify the variable's initial values in the **Variable** module. If you use both methods, the values are read at different times, depending on the options you specify, including the **File** Read Time, the Clear Option, and the replication parameters you specify in the Run Setup dialog box. For more information, see the online Help. There are three methods for manually editing the Initial Values of a **Variable** module: Using the standard spreadsheet interface. In the module spreadsheet, right-click on the Initial Values cell and select the Edit via spreadsheet menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the module dialog box. In the module spreadsheet, right-click on any cell and select the Edit via dialog menu item. The values for two-dimensional arrays should be entered one column at a time. Array elements not explicitly assigned are assumed to have the last entered value. Using the two-dimensional (2-D) spreadsheet interface. In the module spreadsheet, click on the Initial Values cell.

TYPICAL USES Number of documents processed per hour. Serial number to assign to parts for unique identification. Space available in a facility.

PROMPTS Prompt Description Name The unique name of the variable being defined.

Rows Number of rows in a one- or two-dimensional variable.

Columns Number of columns in a two-dimensional variable.

Report Statistics Check box for determining whether or not statistics will be collected. This field is visible when the rows and columns are not specified (that is, for single variables).

Data Type The data type of the values stored in the variable. Valid types are Real and String. The default type is Real.

Clear Option Defines the time (if at all) when the value(s) of the variable is reset to the initial value(s) specified. Specifying Statistics resets this variable to its initial value(s) whenever statistics are cleared. Specifying System resets this variable to its initial value(s) whenever the system is cleared. Specifying None indicates that this variable is never reset to its initial value(s), except prior to the first replication.

File Name Name of the file from which to read the variable's value or values. You can use any file access type supported by Arena except sequential text files and Lotus spreadsheet (.wks) files. If the file name you specify has not been created yet, Arena will create it, but you must edit the file to specify the file access type, path, and recordset (if required).

Recordset Name of the recordset in the specified file from which to read values. This field is available only if you specify a **File Name** for a file that has been set up with a file access type, path, and recordset. Arena uses the Rows and Columns properties to determine the amount of data to read from the recordset. A recordset is required for all file types except .xml. The recordset size must be equal to or

greater than the number of rows and columns specified for the variable. [File Read Time](#) Specifies when to read the values from the file into the variable. If you select PreCheck, the values for the variable are read while the model is still in Edit mode (prior to the model being checked and compiled). If you select BeginSimulation, values are read when the model is compiled, prior to the first replication. If you select BeginReplication, values are read prior to each replication. Initial Values Lists the initial value or values of the variable. You can assign new values to the variable at different stages of the model by using the [Assign](#) module. Initial Value [Variable](#) value at the start of the simulation.

Definition at line [90](#) of file [Variable.h](#).

6.173.2 Constructor & Destructor Documentation

6.173.2.1 Variable()

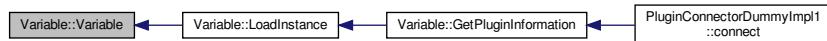
```
Variable::Variable (
    Model * model,
    std::string name = "" )
```

Definition at line [17](#) of file [Variable.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.173.2.2 ~Variable()

```
virtual Variable::~Variable ( ) [virtual], [default]
```

6.173.3 Member Function Documentation

6.173.3.1 `_check()`

```
bool Variable::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 129 of file [Variable.cpp](#).

6.173.3.2 `_initBetweenReplications()`

```
void Variable::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 133 of file [Variable.cpp](#).

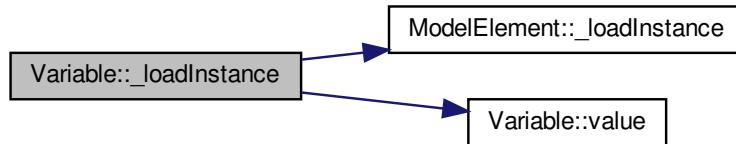
6.173.3.3 `_loadInstance()`

```
bool Variable::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

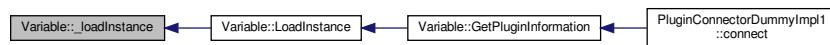
Reimplemented from [ModelElement](#).

Definition at line 98 of file [Variable.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



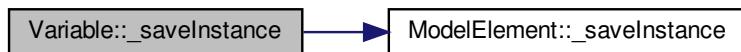
6.173.3.4 `_saveInstance()`

```
std::map< std::string, std::string > * Variable::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 115 of file [Variable.cpp](#).

Here is the call graph for this function:

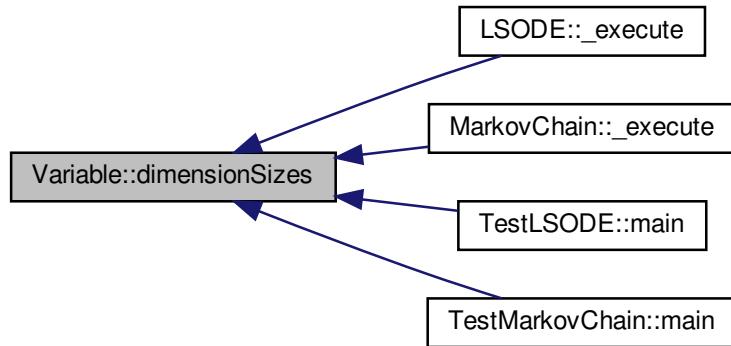


6.173.3.5 `dimensionSizes()`

```
List< unsigned int > * Variable::dimensionSizes ( ) const
```

Definition at line 84 of file [Variable.cpp](#).

Here is the caller graph for this function:

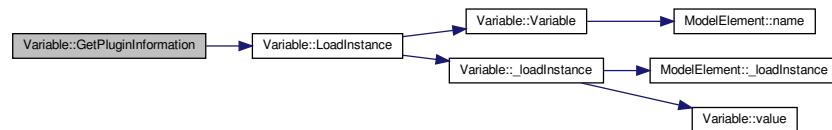


6.173.3.6 GetPluginInformation()

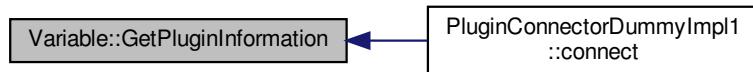
```
PluginInformation * Variable::GetPluginInformation ( ) [static]
```

Definition at line 25 of file [Variable.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.173.3.7 initialValue() [1/2]

```
double Variable::initialValue ( )
```

Definition at line 57 of file [Variable.cpp](#).

6.173.3.8 initialValue() [2/2]

```
double Variable::initialValue ( std::string index )
```

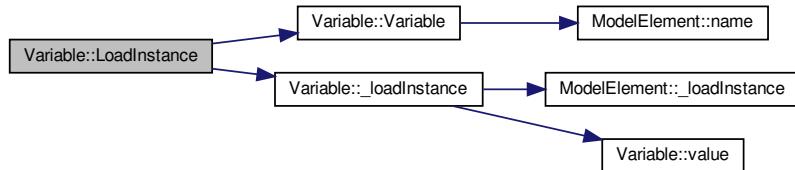
Definition at line 65 of file [Variable.cpp](#).

6.173.3.9 LoadInstance()

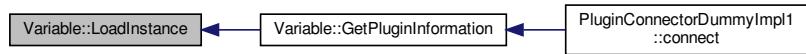
```
ModelElement * Variable::LoadInstance (
    Model * model,
    std::map< std::string, std::string * > * fields ) [static]
```

Definition at line 88 of file [Variable.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

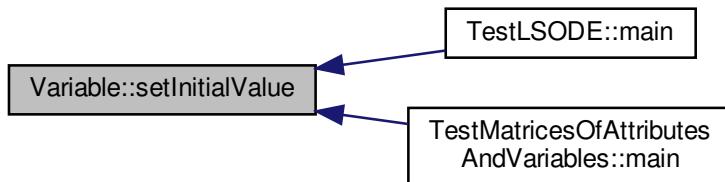


6.173.3.10 setInitialValue() [1/2]

```
void Variable::setInitialValue (
    double value )
```

Definition at line 61 of file [Variable.cpp](#).

Here is the caller graph for this function:



6.173.3.11 setInitialValue() [2/2]

```
void Variable::setInitialValue (
    std::string index,
    double value )
```

Definition at line 74 of file [Variable.cpp](#).

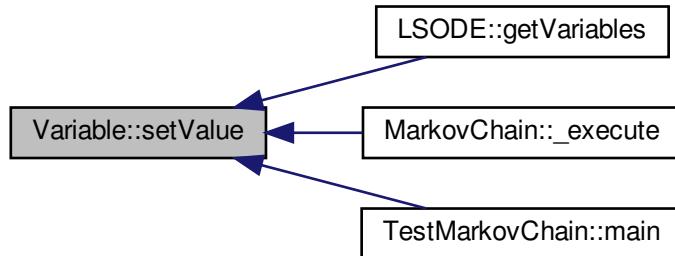
Here is the call graph for this function:

**6.173.3.12 setValue() [1/2]**

```
void Variable::setValue (
    double value )
```

Definition at line 43 of file [Variable.cpp](#).

Here is the caller graph for this function:

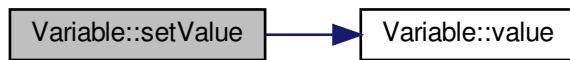


6.173.3.13 setValue() [2/2]

```
void Variable::setValue (
    std::string index,
    double value )
```

Definition at line 47 of file [Variable.cpp](#).

Here is the call graph for this function:



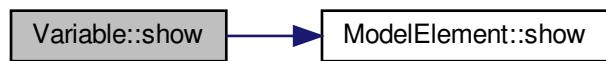
6.173.3.14 show()

```
std::string Variable::show ( ) [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 21 of file [Variable.cpp](#).

Here is the call graph for this function:

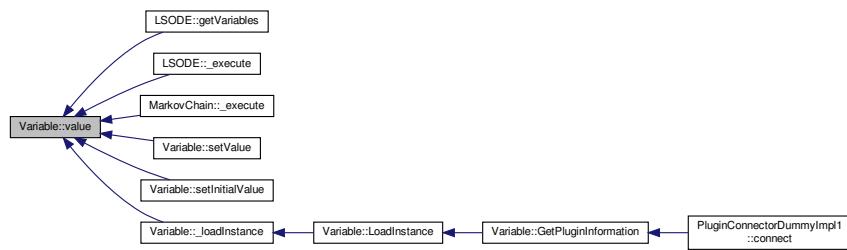


6.173.3.15 value() [1/2]

```
double Variable::value ( )
```

Definition at line 30 of file [Variable.cpp](#).

Here is the caller graph for this function:



6.173.3.16 value() [2/2]

```
double Variable::value (
    std::string index )
```

Definition at line 34 of file [Variable.cpp](#).

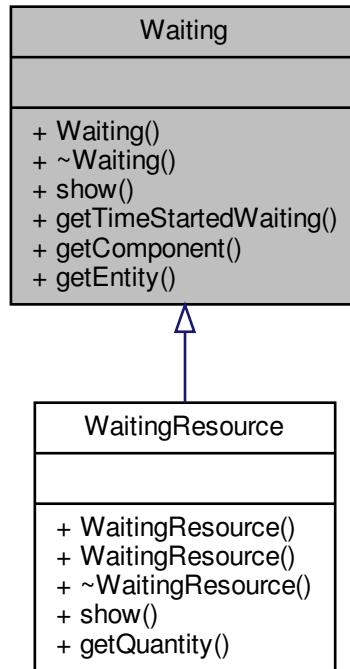
The documentation for this class was generated from the following files:

- [Variable.h](#)
- [Variable.cpp](#)

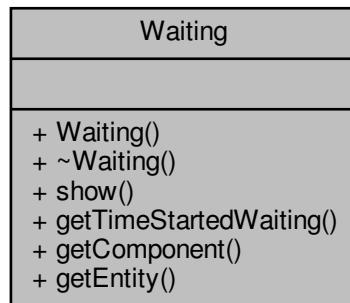
6.174 Waiting Class Reference

```
#include <Queue.h>
```

Inheritance diagram for Waiting:



Collaboration diagram for Waiting:



Public Member Functions

- `Waiting (Entity *entity, ModelComponent *component, double timeStartedWaiting)`
- virtual `~Waiting ()=default`

- virtual std::string [show\(\)](#)
- double [getTimeStartedWaiting\(\)](#) const
- [ModelComponent * getComponent\(\)](#) const
- [Entity * getEntity\(\)](#) const

6.174.1 Detailed Description

Definition at line [25](#) of file [Queue.h](#).

6.174.2 Constructor & Destructor Documentation

6.174.2.1 Waiting()

```
Waiting::Waiting (
    Entity * entity,
    ModelComponent * component,
    double timeStartedWaiting ) [inline]
```

Definition at line [28](#) of file [Queue.h](#).

Here is the call graph for this function:



6.174.2.2 ~Waiting()

```
virtual Waiting::~Waiting ( ) [virtual], [default]
```

Here is the caller graph for this function:



6.174.3 Member Function Documentation

6.174.3.1 getComponent()

```
ModelComponent* Waiting::getComponent ( ) const [inline]
```

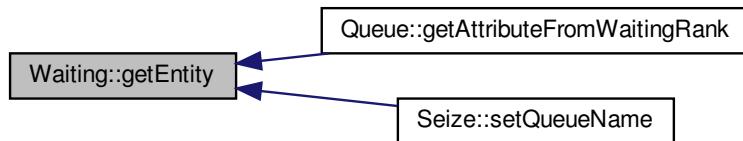
Definition at line 49 of file [Queue.h](#).

6.174.3.2 getEntity()

```
Entity* Waiting::getEntity ( ) const [inline]
```

Definition at line 53 of file [Queue.h](#).

Here is the caller graph for this function:

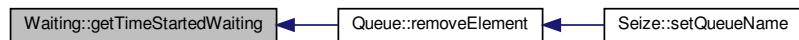


6.174.3.3 getTimeStartedWaiting()

```
double Waiting::getTimeStartedWaiting ( ) const [inline]
```

Definition at line 45 of file [Queue.h](#).

Here is the caller graph for this function:



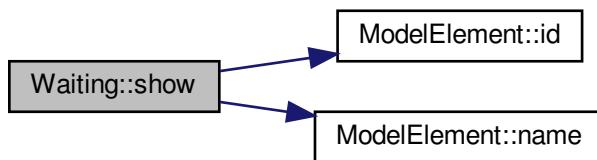
6.174.3.4 show()

```
virtual std::string Waiting::show ( ) [inline], [virtual]
```

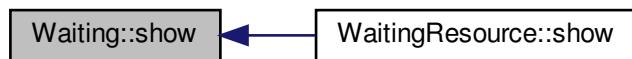
Reimplemented in [WaitingResource](#).

Definition at line [37](#) of file [Queue.h](#).

Here is the call graph for this function:



Here is the caller graph for this function:



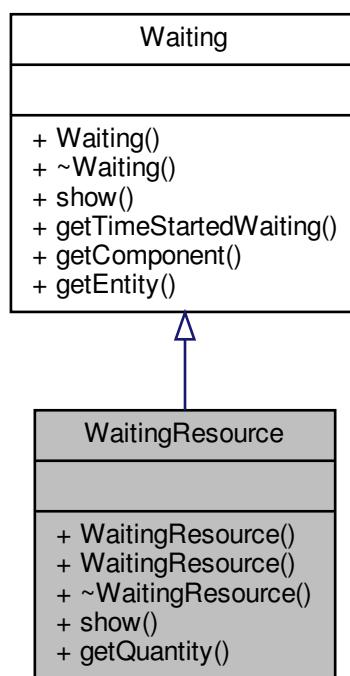
The documentation for this class was generated from the following file:

- [Queue.h](#)

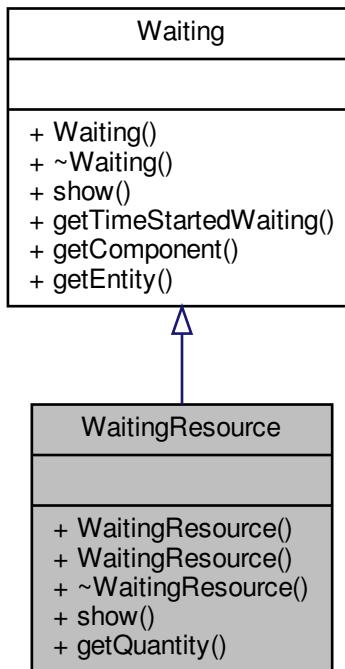
6.175 WaitingResource Class Reference

```
#include <Seize.h>
```

Inheritance diagram for WaitingResource:



Collaboration diagram for WaitingResource:



Public Member Functions

- `WaitingResource (Entity *entity, ModelComponent *component, double timeStartedWaiting, unsigned int quantity)`
- `WaitingResource (const WaitingResource &orig)`
- `virtual ~WaitingResource ()`
- `virtual std::string show ()`
- `unsigned int getQuantity () const`

6.175.1 Detailed Description

Definition at line 24 of file [Seize.h](#).

6.175.2 Constructor & Destructor Documentation

6.175.2.1 WaitingResource() [1/2]

```
WaitingResource::WaitingResource (
    Entity * entity,
    ModelComponent * component,
    double timeStartedWaiting,
    unsigned int quantity ) [inline]
```

Definition at line 27 of file [Seize.h](#).

6.175.2.2 WaitingResource() [2/2]

```
WaitingResource::WaitingResource (
    const WaitingResource & orig ) [inline]
```

Definition at line 31 of file [Seize.h](#).

6.175.2.3 ~WaitingResource()

```
virtual WaitingResource::~WaitingResource ( ) [inline], [virtual]
```

Definition at line 34 of file [Seize.h](#).

6.175.3 Member Function Documentation

6.175.3.1 getQuantity()

```
unsigned int WaitingResource::getQuantity ( ) const [inline]
```

Definition at line 44 of file [Seize.h](#).

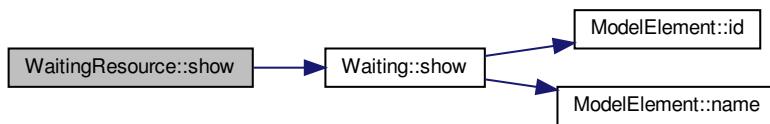
6.175.3.2 show()

```
virtual std::string WaitingResource::show ( ) [inline], [virtual]
```

Reimplemented from [Waiting](#).

Definition at line 38 of file [Seize.h](#).

Here is the call graph for this function:



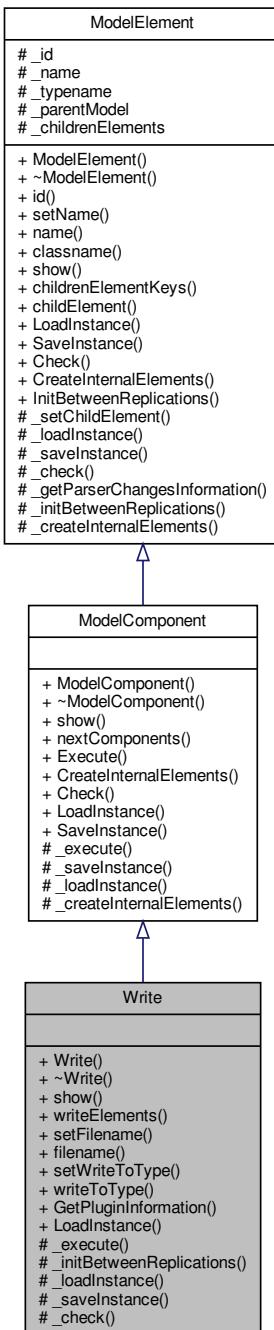
The documentation for this class was generated from the following file:

- [Seize.h](#)

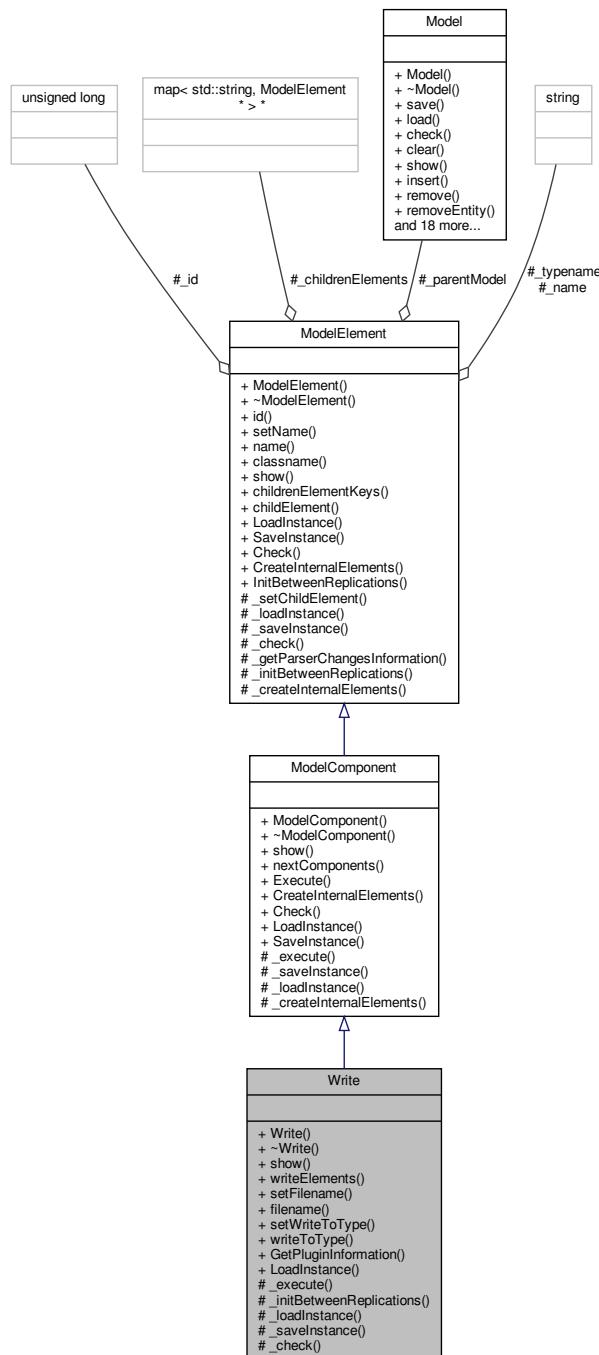
6.176 Write Class Reference

```
#include <Write.h>
```

Inheritance diagram for Write:



Collaboration diagram for Write:



Public Types

- enum `WriteToType` : int { `WriteToType::SCREEN` = 1, `WriteToType::FILE` = 2 }

Public Member Functions

- `Write (Model *model, std::string name="")`

- virtual `~Write ()=default`
- virtual std::string `show ()`
- `List< WriteElement * > * writeElements () const`
- void `setFilename (std::string _filename)`
- std::string `filename () const`
- void `setWriteToType (WriteToType _writeToType)`
- `Write::WriteToType writeToType () const`

Static Public Member Functions

- static `PluginInformation * GetPluginInformation ()`
- static `ModelComponent * LoadInstance (Model *model, std::map< std::string, std::string > *fields)`

Protected Member Functions

- virtual void `_execute (Entity *entity)`
- virtual void `_initBetweenReplications ()`
- virtual bool `_loadInstance (std::map< std::string, std::string > *fields)`
- virtual std::map< std::string, std::string > * `_saveInstance ()`
- virtual bool `_check (std::string *errorMessage)`

Additional Inherited Members

6.176.1 Detailed Description

This component ...

Definition at line 35 of file [Write.h](#).

6.176.2 Member Enumeration Documentation

6.176.2.1 WriteToType

```
enum Write::WriteToType : int [strong]
```

Enumerator

SCREEN	
FILE	

Definition at line 38 of file [Write.h](#).

6.176.3 Constructor & Destructor Documentation

6.176.3.1 Write()

```
Write::Write (
    Model * model,
    std::string name = "" )
```

Definition at line 19 of file [Write.cpp](#).

Here is the caller graph for this function:



6.176.3.2 ~Write()

```
virtual Write::~Write ( ) [virtual], [default]
```

6.176.4 Member Function Documentation

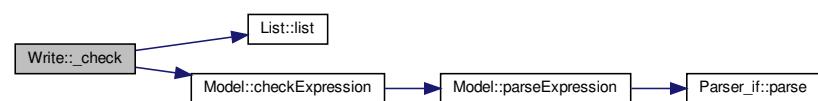
6.176.4.1 _check()

```
bool Write::_check (
    std::string * errorMessage ) [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 109 of file [Write.cpp](#).

Here is the call graph for this function:



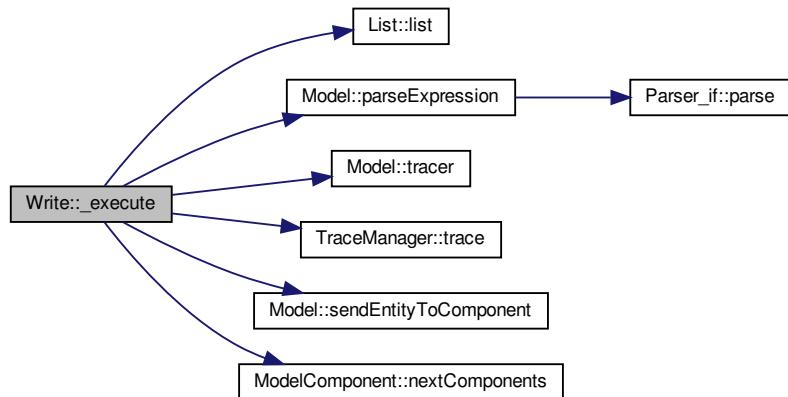
6.176.4.2 `_execute()`

```
void Write::_execute (
    Entity * entity) [protected], [virtual]
```

Implements [ModelComponent](#).

Definition at line 56 of file [Write.cpp](#).

Here is the call graph for this function:



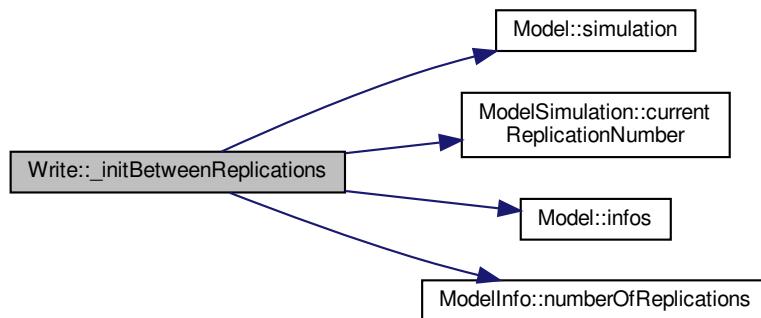
6.176.4.3 `_initBetweenReplications()`

```
void Write::_initBetweenReplications () [protected], [virtual]
```

Reimplemented from [ModelElement](#).

Definition at line 92 of file [Write.cpp](#).

Here is the call graph for this function:



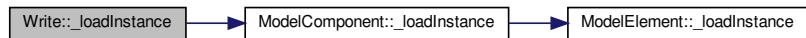
6.176.4.4 `_loadInstance()`

```
bool Write::_loadInstance (
    std::map< std::string, std::string * > * fields ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 84 of file [Write.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



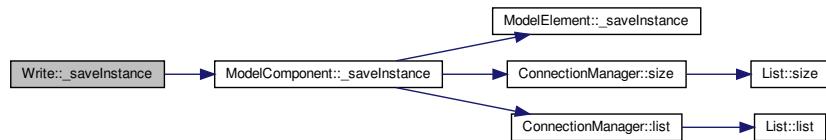
6.176.4.5 `_saveInstance()`

```
std::map< std::string, std::string * > * Write::_saveInstance ( ) [protected], [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 103 of file [Write.cpp](#).

Here is the call graph for this function:



6.176.4.6 filename()

```
std::string Write::filename() const
```

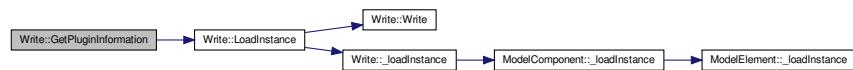
Definition at line 44 of file [Write.cpp](#).

6.176.4.7 GetPluginInformation()

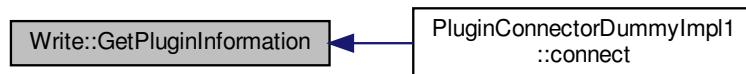
```
PluginInformation * Write::GetPluginInformation() [static]
```

Definition at line 126 of file [Write.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.176.4.8 LoadInstance()

```
ModelComponent * Write::LoadInstance(
    Model * model,
    std::map< std::string, std::string > * fields ) [static]
```

Definition at line 26 of file [Write.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

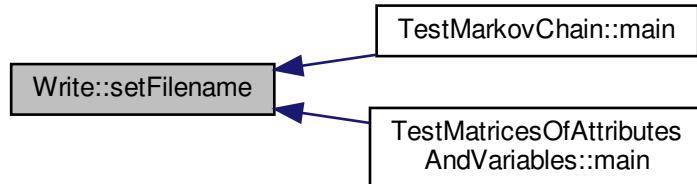


6.176.4.9 setFilename()

```
void Write::setFilename ( std::string _filename )
```

Definition at line 40 of file [Write.cpp](#).

Here is the caller graph for this function:

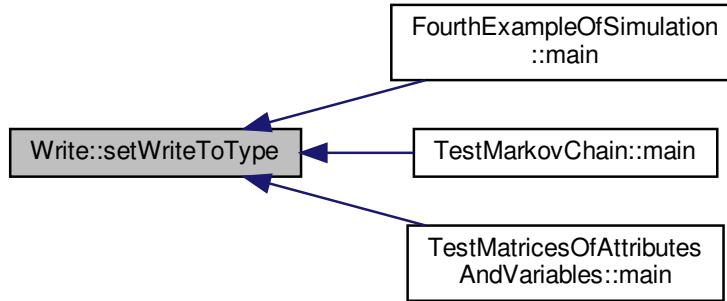


6.176.4.10 setWriteToType()

```
void Write::setWriteToType ( WriteToType _writeToType )
```

Definition at line 48 of file [Write.cpp](#).

Here is the caller graph for this function:



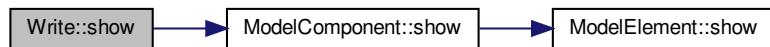
6.176.4.11 show()

```
std::string Write::show ( ) [virtual]
```

Reimplemented from [ModelComponent](#).

Definition at line 22 of file [Write.cpp](#).

Here is the call graph for this function:

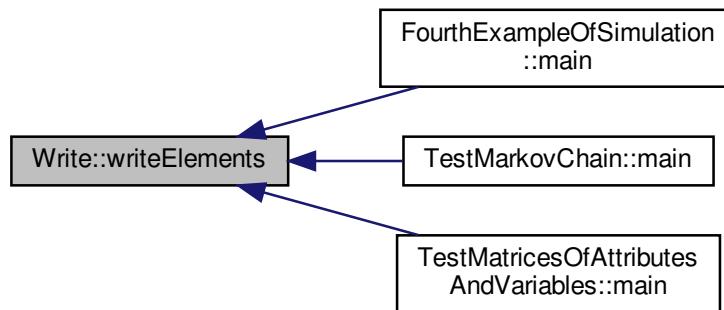


6.176.4.12 writeElements()

```
List< WriteElement * > * Write::writeElements ( ) const
```

Definition at line 36 of file [Write.cpp](#).

Here is the caller graph for this function:



6.176.4.13 writeToType()

`Write::WriteToType Write::writeToType () const`

Definition at line 52 of file [Write.cpp](#).

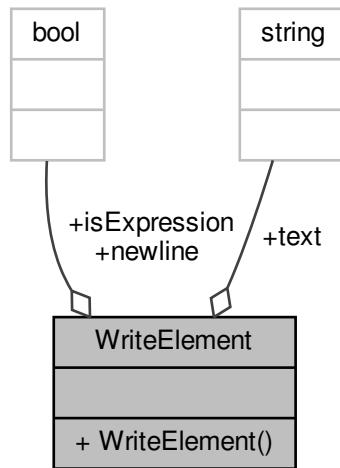
The documentation for this class was generated from the following files:

- [Write.h](#)
- [Write.cpp](#)

6.177 WriteElement Class Reference

```
#include <Write.h>
```

Collaboration diagram for WriteElement:



Public Member Functions

- `WriteElement (std::string text, bool isExpression=false, bool newline=false)`

Public Attributes

- `std::string text`
- `bool isExpression`
- `bool newline`

6.177.1 Detailed Description

Definition at line 19 of file [Write.h](#).

6.177.2 Constructor & Destructor Documentation

6.177.2.1 WriteElement()

```
WriteElement::WriteElement (
    std::string text,
    bool isExpression = false,
    bool newline = false ) [inline]
```

Definition at line 22 of file [Write.h](#).

6.177.3 Member Data Documentation

6.177.3.1 isExpression

```
bool WriteElement::isExpression
```

Definition at line 28 of file [Write.h](#).

6.177.3.2 newline

```
bool WriteElement::newline
```

Definition at line 29 of file [Write.h](#).

6.177.3.3 text

```
std::string WriteElement::text
```

Definition at line 27 of file [Write.h](#).

The documentation for this class was generated from the following file:

- [Write.h](#)

7 File Documentation

7.1 .dep.inc File Reference

7.2 .dep.inc

```
00001 # This code depends on make tool being used
00002 DEPFILES=$(wildcard ${addsuffix .d, ${OBJECTFILES} ${TESTOBJECTFILES}})
00003 ifneq (${DEPFILES},)
00004 include ${DEPFILES}
00005 endif
```

7.3 Access.cpp File Reference

```
#include "Access.h"
#include "Model.h"
```

7.4 Access.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Access.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:14
00012 */
00013
00014 #include "Access.h"
00015
00016 #include "Model.h"
00017
00018 Access::Access(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<Access>(), name) {
00019 }
00020
00021
00022 std::string Access::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Access::_loadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00027     Access* newComponent = new Access(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031
00032     }
00033     return newComponent;
00034 }
00035
00036 void Access::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00039 }
00040
00041 bool Access::_loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelComponent::_loadInstance(fields);
00043     if (res) {
00044         //...
00045     }
00046     return res;
00047 }
00048
00049 void Access::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* Access::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
();
00054     //...
00055     return fields;
00056 }
00057
00058 bool Access::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Access::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Access>(
), &Access::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069
00070

```

7.5 Access.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Access](#)

7.6 Access.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Access.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:14
00012 */
00013
00014 #ifndef ACCESS_H
00015 #define ACCESS_H
00016
00017 #include "ModelComponent.h"
00018
00053 class Access : public ModelComponent {
00054 public: // constructors
00055     Access(Model* model, std::string name="");
00056     virtual ~Access() = default;
00057 public: // virtual
00058     virtual std::string show();
00059 public: // static
00060     static PluginInformation* GetPluginInformation();
00061     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00062                                         std::string>* fields);
00062 protected: // virtual
00063     virtual void _execute(Entity* entity);
00064     virtual void _initBetweenReplications();
00065     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00066     virtual std::map<std::string, std::string>* _saveInstance();
00067     virtual bool _check(std::string* errorMessage);
00068 private: // methods
00069 private: // attributes 1:1
00070 private: // attributes 1:n
00071 };
00072
00073
00074 #endif /* ACCESS_H */
00075

```

7.7 Allocate.cpp File Reference

```
#include "Allocate.h"
#include "Model.h"
```

7.8 Allocate.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Alocate.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:16
00012 */
00013
00014 #include "Allocate.h"
00015

```

```

00016 #include "Model.h"
00017
00018 Allocate::Allocate(Model* model, std::string name) :
00019     ModelComponent(model, Util::TypeOf<Allocate>(), name) {
00020
00021
00022     std::string Allocate::show() {
00023         return ModelComponent::show() + "";
00024     }
00025
00026     ModelComponent* Allocate::LoadInstance(Model* model,
00027         std::map<std::string, std::string>* fields) {
00028         Allocate* newComponent = new Allocate(model);
00029         try {
00030             newComponent->_loadInstance(fields);
00031         } catch (const std::exception& e) {
00032
00033             return newComponent;
00034         }
00035
00036     void Allocate::_execute(Entity* entity) {
00037         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038             entity forward");
00039         this->_parentModel->sendEntityToComponent(entity, this->
00040             nextComponents()->frontConnection(), 0.0);
00041
00042     bool Allocate::_loadInstance(std::map<std::string, std::string>* fields) {
00043         bool res = ModelComponent::_loadInstance(fields);
00044         if (res) {
00045             //...
00046             return res;
00047         }
00048
00049     void Allocate::_initBetweenReplications() {
00050
00051
00052     std::map<std::string, std::string>* Allocate::_saveInstance() {
00053         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00054         ();
00055         //...
00056         return fields;
00057
00058     bool Allocate::_check(std::string* errorMessage) {
00059         bool resultAll = true;
00060         //...
00061         return resultAll;
00062     }
00063
00064     PluginInformation* Allocate::GetPluginInformation() {
00065         PluginInformation* info = new PluginInformation(
00066             Util::TypeOf<Allocate>(), &Allocate::LoadInstance);
00067         // ...
00068         return info;
00069     }
00070

```

7.9 Allocate.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Allocate](#)

7.10 Allocate.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Alocate.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:16
00012 */
00013
00014 #ifndef ALLOCATE_H
00015 #define ALLOCATE_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Allocate : public ModelComponent {
00020 public: // constructors
00021     Allocate(Model* model, std::string name="");
00022     virtual ~Allocate() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040
00041
00042 #endif /* ALLOCATE_H */
00043

```

7.11 AnalysisOfVariance_if.h File Reference

Classes

- class [AnalysisOfVariance_if](#)

7.12 AnalysisOfVariance_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: AnalysisOfVariance_if.h
00009  * Author: rlcancian
00010 *
00011 * Created on 21 de Junho de 2019, 16:08
00012 */
00013
00014 #ifndef ANALYSISOFVARIANCE_IF_H
00015 #define ANALYSISOFVARIANCE_IF_H
00016
00017 class AnalysisOfVariance_if {
00018 public:
00019     virtual void setCollector() = 0;
00020 };
00021
00022 #endif /* ANALYSISOFVARIANCE_IF_H */
00023

```

7.13 Assign.cpp File Reference

```
#include "Assign.h"
#include <string>
#include "Model.h"
#include "Variable.h"
#include "Attribute.h"
#include "Resource.h"
```

7.14 Assign.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 *
00008  * File: Assign.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Agosto de 2018, 10:10
00012 */
00013
00014 #include "Assign.h"
00015 #include <string>
00016 #include "Model.h"
00017 #include "Variable.h"
00018 #include "Attribute.h"
00019 #include "Resource.h"
00020
00021 Assign::Assign(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<Assign>(), name) {
00022 }
00023
00024 std::string Assign::show() {
00025     return ModelComponent::show() +
00026         "";
00027 }
00028
00029 List<Assign::Assignment*>* Assign::assignments() const {
00030     return _assignments;
00031 }
00032
00033 PluginInformation* Assign::GetPluginInformation() {
00034     PluginInformation* info = new PluginInformation(Util::TypeOf<Assign>(
00035         ), &Assign::LoadInstance);
00036     info->insertDynamicLibFileDependence("attribute.so");
00037     info->insertDynamicLibFileDependence("variable.so");
00038     return info;
00039 }
00040 ModelComponent* Assign::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00041     Assign* newComponent = new Assign(model);
00042     try {
00043         newComponent->_loadInstance(fields);
00044     } catch (const std::exception& e) {
00045     }
00046 }
00047 return newComponent;
00048 }
00049
00050 void Assign::_execute(Entity* entity) {
00051     Assignment* let;
00052     std::list<Assignment*>* lets = this->assignments->list();
00053     for (std::list<Assignment*>::iterator it = lets->begin(); it != lets->end(); it++) {
00054         let = (*it);
00055         double value = _parentModel->parseExpression(let->
00056             getExpression());
00057         _parentModel->parseExpression(let->getDestination() + "=" +
00058             std::to_string(value));
00059         _parentModel->tracer()->trace("Let \"\" + let->
00060             getDestination() + "\\" = " + std::to_string(value) + " // " + let->
00061             getExpression());
00062     }
00063 }
```

```

00060     this->_parentModel->sendEntityToComponent(entity, this->
00061         nextComponents()->frontConnection(), 0.0);
00062
00063 void Assign::_initBetweenReplications() {
00064 }
00065
00066 bool Assign::_loadInstance(std::map<std::string, std::string>* fields) {
00067     bool res = ModelComponent::_loadInstance(fields);
00068     if (res) {
00069         unsigned int nv = std::stoi((*(fields->find("assignments"))).second);
00070         for (unsigned int i = 0; i < nv; i++) {
00071             //DestinationType dt = static_cast<DestinationType> (std::stoi((*(fields->find("destinationType" +
00072                 std::to_string(i)))).second));
00073             std::string dest = ((*(fields->find("destination" + std::to_string(i)))).second);
00074             std::string exp = ((*(fields->find("expression" + std::to_string(i)))).second);
00075             Assignment* assmt = new Assignment(dest, exp);
00076             this->_assignments->insert(assmt);
00077         }
00078     }
00079     return res;
00080 }
00081 std::map<std::string, std::string>* Assign::_saveInstance() {
00082     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00083     (); //Util::TypeOf<Assign>());
00084     Assignment* let;
00085     fields->emplace("assignments", std::to_string(_assignments->size()));
00086     unsigned short i = 0;
00087     for (std::list<Assignment*>::iterator it = _assignments->list()->begin(); it != _assignments->list()->
00088         end(); it++) {
00089         let = (*it);
00090         //fields->emplace("destinationType" + std::to_string(i), std::to_string(static_cast<int>
00091         (let->getDestinationType())));
00092         fields->emplace("destination" + std::to_string(i), let->getDestination());
00093         fields->emplace("expression" + std::to_string(i), let->getExpression());
00094     }
00095     return fields;
00096 }
00097 bool Assign::_check(std::string* errorMessage) {
00098     Assignment* let;
00099     bool resultAll = true;
00100     // \todo: Reimplement it. Since 201910, attributes may have index, just like "attrrib1[2]" or
00101     // "att[10,1]". Because of that, the string may contain not only the name of the attribute, but also its index and
00102     // therefore, fails on the test bellow.
00103     for (std::list<Assignment*>::iterator it = _assignments->list()->begin(); it != _assignments->list()->
00104         end(); it++) {
00105         let = (*it);
00106         resultAll &= _parentModel->checkExpression(let->
00107             getExpression(), "assignment", errorMessage);
00108     }
00109     return resultAll;
00110 }
```

7.15 Assign.h File Reference

```
#include "ModelComponent.h"
#include "Model.h"
#include "Plugin.h"
```

Classes

- class [Assign](#)
- class [Assign::Assignment](#)

7.16 Assign.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
```

```

00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Assign.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Agosto de 2018, 10:10
00012 */
00013
00014 #ifndef ASSIGN_H
00015 #define ASSIGN_H
00016
00017 #include "ModelComponent.h"
00018 #include "Model.h"
00019 #include "Plugin.h"
00020
00021 class Assign : public ModelComponent {
00022 public:
00023     class Assignment {
00024     public:
00025         Assignment(std::string destination, std::string expression) {
00026             this->_destination = destination;
00027             this->_expression = expression;
00028             // an assignment is always in the form:
00029             // (destinationType) destination = expression
00030         };
00031         void setDestination(std::string _destination) {
00032             this->_destination = _destination;
00033         }
00034         std::string getDestination() const {
00035             return _destination;
00036         }
00037         void setExpression(std::string _expression) {
00038             this->_expression = _expression;
00039         }
00040         std::string getExpression() const {
00041             return _expression;
00042         }
00043     private:
00044         std::string _destination = "";
00045         std::string _expression = "";
00046     };
00047     public:
00048         Assign(Model* model, std::string name="");
00049         virtual ~Assign() = default;
00050     public:
00051         virtual std::string show();
00052     public:
00053         static PluginInformation* GetPluginInformation();
00054         static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00055                                         std::string>* fields);
00056     public:
00057         List<Assignment*>* assignments() const;
00058     protected:
00059         virtual void _execute(Entity* entity);
00060         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00061         virtual void _initBetweenReplications();
00062         virtual std::map<std::string, std::string>* _saveInstance();
00063         virtual bool _check(std::string* errorMessage);
00064     private:
00065     private:
00066         List<Assignment*>* _assignments = new List<Assignment*>();
00067     };
00068 #endif /* ASSIGN_H */
00069

```

7.17 Attribute.cpp File Reference

```
#include "Attribute.h"
```

7.18 Attribute.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Attribute.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 25 de Setembro de 2018, 16:37
00012 */
00013
00014 #include "Attribute.h"
00015
00016 //Attribute::Attribute(Model* model) : ModelElement(model, Util::TypeOf<Attribute>()) {
00017 //}
00018
00019 Attribute::Attribute(Model* model, std::string name) :
00020     ModelElement(model, Util::TypeOf<Attribute>(), name)
00021
00022
00023 std::string Attribute::show() {
00024     return ModelElement::show();
00025 }
00026
00027 bool Attribute::_loadInstance(std::map<std::string, std::string>* fields) {
00028     return ModelElement::_loadInstance(fields);
00029 }
00030
00031 PluginInformation* Attribute::GetPluginInformation() {
00032     PluginInformation* info = new PluginInformation(
00033         Util::TypeOf<Attribute>(), &Attribute::LoadInstance);
00034     return info;
00035 }
00036
00037 ModelElement* Attribute::LoadInstance(Model* model,
00038     std::map<std::string, std::string>* fields) {
00039     Attribute* newElement = new Attribute(model);
00040     try {
00041         newElement->_loadInstance(fields);
00042     } catch (const std::exception& e) {
00043     }
00044     return newElement;
00045 }
00046
00047 std::map<std::string, std::string>* Attribute::_saveInstance() {
00048     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00049     //Util::TypeOf<Attribute>());
00050     return fields;
00051
00052 bool Attribute::_check(std::string* errorMessage) {
00053     return true;
00054 }

```

7.19 Attribute.h File Reference

```

#include <string>
#include <list>
#include "List.h"
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"

```

Classes

- class [Attribute](#)

7.20 Attribute.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Attribute.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 25 de Setembro de 2018, 16:37
00012 */
00013
00014 #ifndef ATTRIBUTE_H
00015 #define ATTRIBUTE_H
00016
00017 #include <string>
00018 #include <list>
00019 #include "List.h"
00020 #include "ModelElement.h"
00021 #include "ElementManager.h"
00022 #include "Plugin.h"
00023
00024 class Attribute : public ModelElement {
00025 public:
00026     //Attribute(Model* model);
00027     Attribute(Model* model, std::string name="");
00028     virtual ~Attribute() = default;
00029 public:
00030     virtual std::string show();
00031 public:
00032     static PluginInformation* GetPluginInformation();
00033     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00034                                         std::string>* fields);
00035 protected:
00036     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00037     virtual std::map<std::string, std::string>* _saveInstance();
00038     virtual bool _check(std::string* errorMessage);
00039 private:
00040     //List<unsigned int>* _dimensionSizes = new List<unsigned int>();
00041 };
00042
00043 #endif /* ATTRIBUTE_H */
00044

```

7.21 BaseConsoleGenesysApplication.cpp File Reference

```

#include "BaseConsoleGenesysApplication.h"
#include <iostream>
#include "Simulator.h"

```

7.22 BaseConsoleGenesysApplication.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: BaseConsoleGenesysApplication.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 3 de Setembro de 2019, 16:25
00012 */
00013
00014 #include "BaseConsoleGenesysApplication.h"
00015 #include <iostream>
00016
00017 // GEnSyS Simulator
00018 #include "Simulator.h"

```

```
00020
00021
00022 BaseConsoleGenesysApplication::BaseConsoleGenesysApplication
00023 {
00024
00025
00026 // default Trace Handlers
00027
00028 void BaseConsoleGenesysApplication::traceHandler(
00029     TraceEvent e) {
00030     std::cout << e.text() << std::endl;
00031 }
00032 void BaseConsoleGenesysApplication::traceErrorHandler(
00033     TraceErrorEvent e) {
00034     std::cout << e.text() << std::endl;
00035 }
00036 void BaseConsoleGenesysApplication::traceReportHandler(
00037     TraceEvent e) {
00038     std::cout << "" << e.text() << "" << std::endl;
00039 }
00040 void BaseConsoleGenesysApplication::traceSimulationHandler(
00041     TraceSimulationEvent e) {
00042     std::cout << e.text() << std::endl;
00043 }
00044
00045 // default Event Handlers
00046
00047 void BaseConsoleGenesysApplication::onSimulationStartHandler
00048     (SimulationEvent* re) {
00049     std::cout << "(Event Handler) " << "Simulation is starting" << std::endl;
00050 }
00051 void BaseConsoleGenesysApplication::onReplicationStartHandler
00052     (SimulationEvent* re) {
00053     std::cout << "(Event Handler) " << "Replication " << re->
00054         getReplicationNumber() << " starting." << std::endl;
00055 }
00056 void BaseConsoleGenesysApplication::onProcessEventHandler
00057     (SimulationEvent* re) {
00058     std::cout << "(Event Handler) " << "Processing event " << re->
00059         getEventProcessed()->show() << std::endl;
00060 }
00061
00062 void BaseConsoleGenesysApplication::onReplicationEndHandler
00063     (SimulationEvent* re) {
00064     std::cout << "(Event Handler) " << "Replication " << re->
00065         getReplicationNumber() << " ending." << std::endl;
00066 }
00067 void BaseConsoleGenesysApplication::onSimulationEndHandler
00068     (SimulationEvent* re) {
00069     std::cout << "Event (Handler) " << "Replication " << re->
00070         getReplicationNumber() << " ending." << std::endl;
00071
00072 void BaseConsoleGenesysApplication::setDefaultEventHandlers
00073     (OnEventManager* oem) {
00074     //oem->addOnSimulationStartHandler(this, &BaseConsoleGenesysApplication::onSimulationStartHandler);
00075     //oem->addOnReplicationStartHandler(this, BaseConsoleGenesysApplication::onReplicationStartHandler);
00076     oem->AddOnProcessEventHandler(this, &
00077         BaseConsoleGenesysApplication::onProcessEventHandler);
00078     //oem->addOnReplicationEndHandler(this, &BaseConsoleGenesysApplication::onReplicationEndHandler);
00079     //oem->addOnSimulationEndHandler(this, &BaseConsoleGenesysApplication::onSimulationEndHandler);
00080     //oem->addOnEntityRemoveHandler(this, &BaseConsoleGenesysApplication::onEntityRemoveHandler);
00081 }
00082
00083 void BaseConsoleGenesysApplication::setDefaultTraceHandlers
00084     (TraceManager* tm) {
00085     tm->addTraceHandler<BaseConsoleGenesysApplication>(this, &
00086         BaseConsoleGenesysApplication::traceHandler);
00087     tm->addTraceErrorHandler<BaseConsoleGenesysApplication
00088         >(this, &BaseConsoleGenesysApplication::traceErrorHandler);
00089     tm->addTraceReportHandler<BaseConsoleGenesysApplication
00090         >(this, &BaseConsoleGenesysApplication::traceReportHandler)
00091 ;
```

```

00084     tm->addTraceSimulationHandler<
00085         BaseConsoleGenesysApplication>(this, &
00086             BaseConsoleGenesysApplication::traceSimulationHandler)
00087     ;
00088 }
00089
00087 void BaseConsoleGenesysApplication::insertFakePluginsByHand
00088     (Simulator* simulator) {
00089     // model components
00090     // arena basic process
00091     simulator->plugins()->insert("create.so");
00092     simulator->plugins()->insert("dispose.so");
00093     simulator->plugins()->insert("decide.so");
00094     simulator->plugins()->insert("batch.so");
00095     simulator->plugins()->insert("separate.so");
00096     simulator->plugins()->insert("assign.so");
00097     simulator->plugins()->insert("record.so");
00098     simulator->plugins()->insert("submodel.so");
00099     simulator->plugins()->insert("entitytype.so");
00100     simulator->plugins()->insert("entitygroup.so");
00101     simulator->plugins()->insert("attribute.so");
00102     simulator->plugins()->insert("counter.so");
00103     simulator->plugins()->insert("queue.so");
00104     simulator->plugins()->insert("set.so");
00105     simulator->plugins()->insert("resource.so");
00106     simulator->plugins()->insert("variable.so");
00107     simulator->plugins()->insert("schedule.so");
00108     simulator->plugins()->insert("entitygroup.so");
00109     // arena advanced process
00110     simulator->plugins()->insert("delay.so");
00111     simulator->plugins()->insert("dropoff.so");
00112     simulator->plugins()->insert("hold.so");
00113     simulator->plugins()->insert("match.so");
00114     simulator->plugins()->insert("pickup.so");
00115     simulator->plugins()->insert("read.so");
00116     simulator->plugins()->insert("write.so");
00117     simulator->plugins()->insert("release.so");
00118     simulator->plugins()->insert("remove.so");
00119     simulator->plugins()->insert("seize.so");
00120     simulator->plugins()->insert("search.so");
00121     simulator->plugins()->insert("signal.so");
00122     simulator->plugins()->insert("store.so");
00123     simulator->plugins()->insert("unstore.so");
00124     simulator->plugins()->insert("expression.so");
00125     simulator->plugins()->insert("failure.so");
00126     simulator->plugins()->insert("file.so");
00127     simulator->plugins()->insert("statisticscollector.so");
00128     simulator->plugins()->insert("storage.so");
00129     // arena transfer station
00130     simulator->plugins()->insert("enter.so");
00131     simulator->plugins()->insert("leave.so");
00132     simulator->plugins()->insert("pickstation.so");
00133     simulator->plugins()->insert("route.so");
00134     simulator->plugins()->insert("sequence.so");
00135     simulator->plugins()->insert("station.so");
00136     // arena transfer conveyour
00137     simulator->plugins()->insert("access.so");
00138     simulator->plugins()->insert("exit.so");
00139     simulator->plugins()->insert("start.so");
00140     simulator->plugins()->insert("stop.so");
00141     simulator->plugins()->insert("conveyour.so");
00142     simulator->plugins()->insert("segment.so");
00143     // arena transfer transport
00144     simulator->plugins()->insert("alocate.so");
00145     simulator->plugins()->insert("free.so");
00146     simulator->plugins()->insert("halt.so");
00147     simulator->plugins()->insert("move.so");
00148     simulator->plugins()->insert("request.so");
00149     simulator->plugins()->insert("transporter.so");
00150     simulator->plugins()->insert("distance.so");
00151     simulator->plugins()->insert("network.so");
00152     simulator->plugins()->insert("networklink.so");
00153     // others
00154     simulator->plugins()->insert("dummy.so");
00155     simulator->plugins()->insert("lsode.so");
00156     simulator->plugins()->insert("biochemical.so");
00157     simulator->plugins()->insert("markovchain.so");
00158     simulator->plugins()->insert("cellularautomata.so");
00159 }
```

7.23 BaseConsoleGenesysApplication.h File Reference

```
#include "GenesysApplication_if.h"
```

```
#include "TraceManager.h"
#include "OnEventManager.h"
```

Classes

- class [BaseConsoleGenesysApplication](#)

7.24 BaseConsoleGenesysApplication.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  BaseConsoleGenesysApplication.h
00009  * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 16:25
00012 */
00013
00014 #ifndef BASECONSOLEGESYSAPPLICATION_H
00015 #define BASECONSOLEGESYSAPPLICATION_H
00016
00017 #include "GenesysApplication_if.h"
00018 #include "TraceManager.h"
00019 #include "OnEventManager.h"
00020
00021 class BaseConsoleGenesysApplication : public
00022     GenesysApplication_if {
00023     public:
00024         BaseConsoleGenesysApplication();
00025         virtual ~BaseConsoleGenesysApplication() = default;
00026     public:
00027         virtual int main(int argc, char** argv) = 0;
00028         void setDefaultTraceHandlers(TraceManager* tm);
00029         void setDefaultEventHandlers(OnEventManager* oem);
00030         void insertFakePluginsByHand(Simulator* simulator);
00031     protected:
00032         // default Trace Handlers
00033         virtual void traceHandler(TraceEvent e);
00034         virtual void traceErrorHandler(TraceErrorEvent e);
00035         virtual void traceReportHandler(TraceEvent e);
00036         virtual void traceSimulationHandler(
00037             TraceSimulationEvent e);
00038         // default Event Handlers
00039         virtual void onSimulationStartHandler(
00040             SimulationEvent* re);
00041         virtual void onReplicationStartHandler(
00042             SimulationEvent* re);
00043         virtual void onProcessEventHandler(SimulationEvent* re);
00044         virtual void onReplicationEndHandler(SimulationEvent* re);
00045         virtual void onSimulationEndHandler(SimulationEvent* re);
00046     private:
00047
00048 #endif /* BASECONSOLEGESYSAPPLICATION_H */
00049
```

7.25 Batch.cpp File Reference

```
#include "Batch.h"
#include "Model.h"
```

7.26 Batch.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Batch.cpp
00009  * Author:  rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Batch.h"
00015
00016 #include "Model.h"
00017
00018 Batch::Batch(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Batch>(), name)
00020
00021
00022 std::string Batch::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Batch::_loadInstance(Model* model,
00027     std::map<std::string, std::string>* fields) {
00028     Batch* newComponent = new Batch(model);
00029     try {
00030         newComponent->_loadInstance(fields);
00031     } catch (const std::exception& e) {
00032     }
00033     return newComponent;
00034 }
00035
00036 void Batch::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038     entity forward");
00039     this->_parentModel->sendEntityToComponent(entity, this->
00040         nextComponents()->frontConnection(), 0.0);
00041 }
00042
00043 bool Batch::_loadInstance(std::map<std::string, std::string>* fields) {
00044     bool res = ModelComponent::_loadInstance(fields);
00045     if (res) {
00046         //...
00047     }
00048     return res;
00049 }
00050
00051 void Batch::_initBetweenReplications() {
00052
00053     std::map<std::string, std::string>* Batch::_saveInstance() {
00054         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00055         ();
00056         //...
00057         return fields;
00058     }
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Batch::GetPluginInformation() {
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Batch>()
00066         , &Batch::_loadInstance);
00067     // ...
00068     return info;
00069 }
00070

```

7.27 Batch.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Batch](#)

7.28 Batch.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Batch.h
00009  * Author:  rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef BATCH_H
00015 #define BATCH_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Batch : public ModelComponent {
00020 public: // constructors
00021     Batch(Model* model, std::string name="");
00022     virtual ~Batch() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074 #endif /* BATCH_H */
00075

```

7.29 BuildSimulationModel03.cpp File Reference

```

#include "BuildSimulationModel03.h"
#include "GenesysConsole.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Dummy.h"
#include "ElementManager.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"
#include "ProbDistrib.h"
#include "EntityGroup.h"
#include "Formula.h"
#include "OLD_ODElement.h"

```

Functions

- void [_buildMostCompleteModel \(Model *model\)](#)

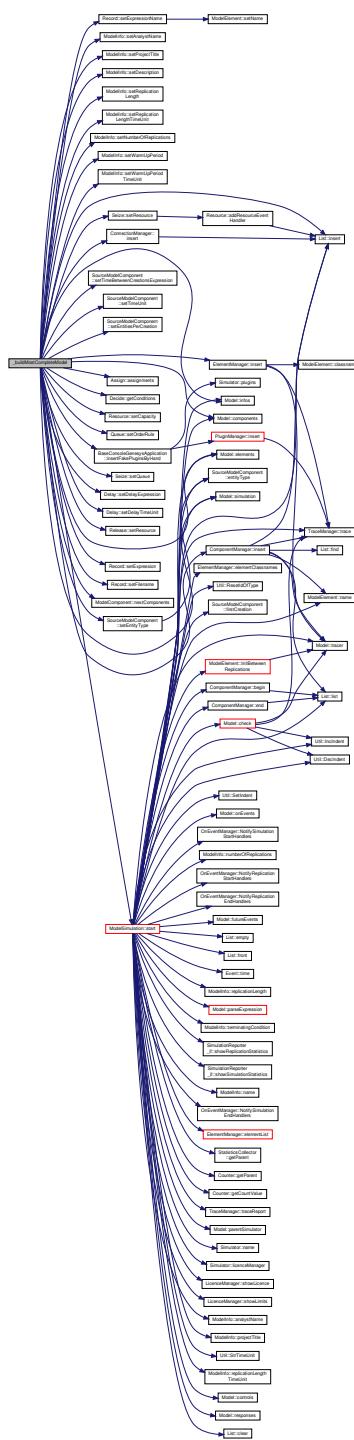
7.29.1 Function Documentation

7.29.1.1 [_buildMostCompleteModel\(\)](#)

```
void _buildMostCompleteModel (
    Model * model )
```

Definition at line [44](#) of file [BuildSimulationModel03.cpp](#).

Here is the call graph for this function:



7.30 BuildSimulationModel03.cpp

```
00001 /*  
00002 * To change this license header, choose License Headers in Project Properties.  
00003 * To change this template file, choose Tools | Templates  
00004 * and open the template in the editor.  
00005 */  
00006  
00007 */
```

```

00008 * File: MyReGenESYsApplication.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 20 de Maio de 2019, 21:01
00012 */
00013
00014 #include "BuildSimulationModel03.h"
00015 #include "GenesysConsole.h"
00016
00017 // GEnSyS Simulator
00018 #include "Simulator.h"
00019
00020 // Model Components
00021 #include "Create.h"
00022 #include "Delay.h"
00023 #include "Dispose.h"
00024 #include "Seize.h"
00025 #include "Release.h"
00026 #include "Assign.h"
00027 #include "Record.h"
00028 #include "Decide.h"
00029 #include "Dummy.h"
00030
00031 // Model elements
00032 #include "ElementManager.h"
00033 #include "EntityType.h"
00034 #include "Attribute.h"
00035 #include "Variable.h"
00036 #include "ProbDistrib.h"
00037 #include "EntityGroup.h"
00038 #include "Formula.h"
00039 #include "OLD_ODElement.h"
00040
00041 BuildSimulationModel03::BuildSimulationModel03() {
00042 }
00043
00044 void _buildMostCompleteModel(Model* model) {
00045     // buildModelWithAllImplementedComponents
00046     ModelInfo* infos = model->infos();
00047     infos->setAnalystName("Your name");
00048     infos->setProjectTitle("The title of the project");
00049     infos->setDescription("The description of the project");
00050     infos->setReplicationLength(1e3);
00051     infos->setReplicationLengthTimeUnit(
00052         Util::TimeUnit::minute);
00053     infos->setNumberOfReplications(10);
00054     infos->setWarmUpPeriod(50);
00055     infos->setWarmUpPeriodTimeUnit(Util::TimeUnit::minute);
00056     infos->setDescription("./models/model99_AllTogether.txt");
00057
00058     ComponentManager* components = model->components();
00059     ElementManager* elements = model->elements();
00060
00061     EntityType* entityType1 = new EntityType(model, "Representative_EntityType");
00062     elements->insert(entityType1);
00063
00064     Create* createl = new Create(model);
00065     createl->setEntityType(entityType1);
00066     createl->setTimeBetweenCreationsExpression("EXPO(5)");
00067     createl->setTimeUnit(Util::TimeUnit::minute);
00068     createl->setEntitiesPerCreation(1);
00069     components->insert(createl);
00070
00071     Attribute* attribute1 = new Attribute(model, "Attribute_1");
00072     elements->insert(attribute1);
00073     Variable* variable1 = new Variable(model, "Variable_1");
00074     elements->insert(variable1);
00075
00076     Assign* assign1 = new Assign(model);
00077     Assign::Assignment* attrib2Assignment = new
00078         Assign::Assignment("Variable_1", "Variable_1 + 1");
00079     assign1->assignments()->insert(attrib2Assignment);
00080     Assign::Assignment* attrib1Assignment = new
00081         Assign::Assignment("Attribute_1", "Variable_1");
00082     assign1->assignments()->insert(attrib1Assignment);
00083     components->insert(assign1);
00084
00085     Decide* decide1 = new Decide(model);
00086     //decide1->getConditions()->insert("UNIF(0,1)>=0.5");
00087     decide1->getConditions()->insert("NQ(Queue_Machine_1) <= 2*NQ(Queue_Machine_2)");
00088
00089     Resource* machine1 = new Resource(model, "Machine_1");
00090     machine1->setCapacity(1);
00091     elements->insert(machine1);
00092
00093     Queue* queueSeize1 = new Queue(model, "Queue_Machine_1");
00094     queueSeize1->setOrderRule(Queue::OrderRule::FIFO);

```

```

00092     elements->insert(queueSeize1);
00093
00094     Seize* seize1 = new Seize(model);
00095     seize1->setResource(machine1);
00096     seize1->setQueue(queueSeize1);
00097     components->insert(seize1);
00098
00099     Delay* delay1 = new Delay(model);
00100     delay1->setDelayExpression("NORM(5,1.5)");
00101     delay1->setDelayTimeUnit(Util::TimeUnit::minute);
00102     components->insert(delay1);
00103
00104     Release* release1 = new Release(model);
00105     release1->setResource(machine1);
00106     components->insert(release1);
00107
00108     Record* record1 = new Record(model);
00109     record1->setExpressionName("Time in system after releasing machine 1");
00110     record1->setExpression("TNOW - Entity.ArrivalTime");
00111     record1->setFilename("./temp/TimeAfterMachine1.txt");
00112     components->insert(record1);
00113
00114     Dispose* dispose1 = new Dispose(model);
00115     components->insert(dispose1);
00116
00117     Resource* machine2 = new Resource(model, "Machine_2");
00118     machine2->setCapacity(1);
00119     elements->insert(machine2);
00120
00121     Queue* queueSeize2 = new Queue(model, "Queue_Machine_2");
00122     queueSeize2->setOrderRule(Queue::OrderRule::FIFO);
00123     elements->insert(queueSeize2);
00124
00125     Seize* seize2 = new Seize(model);
00126     seize2->setResource(machine2);
00127     seize2->setQueue(queueSeize2);
00128     components->insert(seize2);
00129
00130     Delay* delay2 = new Delay(model);
00131     delay2->setDelayExpression("NORM(5,1.5)");
00132     delay2->setDelayTimeUnit(Util::TimeUnit::minute);
00133     components->insert(delay2);
00134
00135     Release* release2 = new Release(model);
00136     release2->setResource(machine2);
00137     components->insert(release2);
00138
00139     Record* record2 = new Record(model);
00140     record2->setExpressionName("Time in system after releasing machine 2");
00141     record2->setExpression("TNOW - Entity.ArrivalTime");
00142     record2->setFilename("./temp/TimeAfterMachine2.txt");
00143     components->insert(record2);
00144
00145     Dummy* dummy1 = new Dummy(model);
00146     components->insert(dummy1);
00147
00148 // connect model components to create a "workflow" -- should always start from a SourceModelComponent
// and end at a SinkModelComponent (it will be checked)
00149     createl->nextComponents()->insert(assign1);
00150     assign1->nextComponents()->insert(decide1);
00151     decide1->nextComponents()->insert(seize1);
00152     decide1->nextComponents()->insert(seize2);
00153     seize1->nextComponents()->insert(delay1);
00154     delay1->nextComponents()->insert(release1);
00155     release1->nextComponents()->insert(record1);
00156     record1->nextComponents()->insert(dispose1);
00157     seize2->nextComponents()->insert(delay2);
00158     delay2->nextComponents()->insert(release2);
00159     release2->nextComponents()->insert(record2);
00160     record2->nextComponents()->insert(dummy1);
00161     dummy1->nextComponents()->insert(dispose1);
00162 }
00163
00170 void BuildSimulationModel03::builAndRunSimulationModel() {
00171     Simulator* simulator = new Simulator();
00172
00173     // traces handle and simulation events to output them
00174     //TraceManager* tm = simulator->getTraceManager();
00175     //tm->addTraceHandler(&traceHandlerFunction);
00176     //tm->addTraceReportHandler(&traceHandlerFunction);
00177     //tm->addTraceSimulationHandler(&traceSimulationHandlerFunction);
00178
00179     // Insert some fake plugins, since components and elements are NOT dynamic linked.
00180     // Basically all ModelComponents and ModelElements classes that may be used to buikd simulation models
00181     // and to be persisted to files, should be "declared" by plugins.
00182     this->insertFakePluginsByHand(simulator);
00183

```

```

00183     Model* model = new Model(simulator);
00184
00185     //OnEventManager* ev = model->getOnEventManager();
00186     //ev->addOnSimulationStartHandler(&onSimulationStartHandlerFunction);
00187     //ev->addOnReplicationStartHandler(&onReplicationStartHandlerFunction);
00188     //ev->addOnReplicationEndHandler(&onReplicationEndHandlerFunction);
00189     //ev->addOnProcessEventHandler(&onProcessEventHandlerFunction);
00190
00191     _buildMostCompleteModel(model);
00192
00193     //simulator->getModelManager()->insert(model);
00194     //model->saveModel(model->infos()->getDescription());
00195
00196     //simulator->getModelManager()->loadModel("./models/model199_AllTogether.txt");
00197     //simulator->getModelManager()->loadModel("./models/model01_CreDelDis.txt");
00198
00199     //model->checkModel();
00200     //model->show();
00201     model->simulation()->start();
00202 }
00203
00204 int BuildSimulationModel03::main(int argc, char** argv) {
00205     this->builAndRunSimulationModel();
00206     return 0;
00207 }
```

7.31 BuildSimulationModel03.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
#include "Simulator.h"
```

Classes

- class [BuildSimulationModel03](#)

7.32 BuildSimulationModel03.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: MyReGenESYsApplication.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 20 de Maio de 2019, 21:01
00012 */
00013
00014 #ifndef BUILDSIMULATIONMODEL03_H
00015 #define BUILDSIMULATIONMODEL03_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 #include "Simulator.h"
00020
00021 class BuildSimulationModel03 : public
00022     BaseConsoleGenesysApplication {
00023     public:
00024         BuildSimulationModel03();
00025     public:
00026         virtual int main(int argc, char** argv);
00027     private:
00028         void builAndRunSimulationModel();
00029     };
00030 #endif /* BUILDSIMULATIONMODEL03_H */
00031
```

7.33 CellularAutomata.cpp File Reference

```
#include "CellularAutomata.h"
```

7.34 CellularAutomata.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: CelularAutomata.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "CellularAutomata.h"
00015
00016 CelularAutomata::CelularAutomata() {
00017 }
00018
00019
```

7.35 CellularAutomata.h File Reference

Classes

- class [CelularAutomata](#)

7.36 CellularAutomata.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: CelularAutomata.h
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef CELULARAUTOMATA_H
00015 #define CELULARAUTOMATA_H
00016
00017 class CelularAutomata {
00018 public:
00019     CelularAutomata();
00020     virtual ~CelularAutomata() = default;
00021 private:
00022
00023 };
00024
00025 #endif /* CELULARAUTOMATA_H */
```

7.37 Collector_if.h File Reference

```
#include <string>
#include <functional>
```

Classes

- class [Collector_if](#)

TypeDefs

- `typedef std::function< void(double) > CollectorAddValueHandler`
- `typedef std::function< void() > CollectorClearHandler`

Functions

- template<typename Class >
`CollectorAddValueHandler SetCollectorAddValueHandler (void(Class::*function)(double), Class *object)`
- template<typename Class >
`CollectorClearHandler SetCollectorClearHandler (void(Class::*function)(), Class *object)`

7.37.1 Typedef Documentation

7.37.1.1 CollectorAddValueHandler

```
typedef std::function<void(double) > CollectorAddValueHandler
```

Definition at line 22 of file [Collector_if.h](#).

7.37.1.2 CollectorClearHandler

```
typedef std::function<void() > CollectorClearHandler
```

Definition at line 29 of file [Collector_if.h](#).

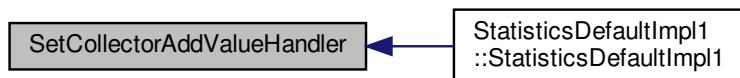
7.37.2 Function Documentation

7.37.2.1 SetCollectorAddValueHandler()

```
template<typename Class >
CollectorAddValueHandler SetCollectorAddValueHandler (
    void(Class::*)(double) function,
    Class * object )
```

Definition at line 25 of file [Collector_if.h](#).

Here is the caller graph for this function:

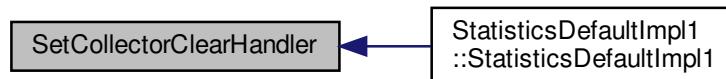


7.37.2.2 SetCollectorClearHandler()

```
template<typename Class >
CollectorClearHandler SetCollectorClearHandler (
    void(Class::* )() function,
    Class * object )
```

Definition at line 32 of file [Collector_if.h](#).

Here is the caller graph for this function:



7.38 Collector_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Collector_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 14 de Agosto de 2018, 14:16
00012 */
00013
00014 #ifndef COLLECTOR_IF_H
00015 #define COLLECTOR_IF_H
00016
00017 #include <string>
00018 #include <functional>
00019
00020 typedef std::function<void(double) > CollectorAddValueHandler;
00021
00022 template<typename Class>
00023 CollectorAddValueHandler SetCollectorAddValueHandler(
00024     void (Class::*function)(double), Class * object) {
00025     return std::bind(function, object, std::placeholders::_1);
00026 }
00027
00028
00029 typedef std::function<void() > CollectorClearHandler;
00030
00031 template<typename Class>
00032 CollectorClearHandler SetCollectorClearHandler(void (Class::*function)(), Class *
00033     object) {
00034     return std::bind(function, object);
00035 }
00036
00037 class Collector_if {
00038 public:
00039     virtual void clear() = 0;
00040     virtual void addValue(double value) = 0;
00041     virtual double getLastValue() = 0;
00042     virtual unsigned long numElements() = 0;
00043
00044 public:
00045     virtual void setAddValueHandler(CollectorAddValueHandler
00046         addValueHandler) = 0;
00047     virtual void setClearHandler(CollectorClearHandler clearHandler) = 0;
00048 };
00049
00050 #endif /* COLLECTOR_IF_H */
00051
```

7.39 CollectorDatafile_if.h File Reference

```
#include "Collector_if.h"
```

Classes

- class [CollectorDatafile_if](#)

7.40 CollectorDatafile_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    CollectorDatafile_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Agosto de 2018, 16:45
00012 */
00013
00014 #ifndef COLLECTORDATAFILE_IF_H
00015 #define COLLECTORDATAFILE_IF_H
00016
00017 #include "Collector_if.h"
00018
00019 class CollectorDatafile_if : public Collector_if {
00020 public:
00021     virtual double getValue(unsigned int rank) = 0;
00022     virtual void seekFirstValue() = 0;
00023     virtual double getNextValue() = 0;
00024     virtual std::string getDataFilename() = 0;
00025     virtual void setDataFilename(std::string filename) = 0;
00026 };
00027
00028 #endif /* COLLECTORDATAFILE_IF_H */
00029
```

7.41 CollectorDatafileDefaultImpl1.cpp File Reference

```
#include "CollectorDatafileDefaultImpl1.h"
```

7.42 CollectorDatafileDefaultImpl1.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    CollectorDatafileDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 1 de Agosto de 2018, 20:58
00012 */
00013
00014 #include "CollectorDatafileDefaultImpl1.h"
00015
00016 CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1
00017 {
00018 }
```

```

00019
00020 void CollectorDatafileDefaultImpl1::clear() {
00021 }
00022
00023 void CollectorDatafileDefaultImpl1::addValue(double value) {
00024 }
00025
00026 double CollectorDatafileDefaultImpl1::getLastValue() {
00027     return 0.0; // \todo:
00028 }
00029
00030 unsigned long CollectorDatafileDefaultImpl1::numElements() {
00031     return 0.0; // \todo:
00032 }
00033
00034 double CollectorDatafileDefaultImpl1::getValue(unsigned int num) {
00035     return 0.0; // \todo:
00036 }
00037
00038 double CollectorDatafileDefaultImpl1::getNextValue() {
00039     return 0.0; // \todo:
00040 }
00041
00042 void CollectorDatafileDefaultImpl1::seekFirstValue() {
00043 }
00044
00045 std::string CollectorDatafileDefaultImpl1::getDataFilename()
{
00046     return _filename;
00047 }
00048
00049 void CollectorDatafileDefaultImpl1::setDataFilename(
    std::string filename) {
00050     _filename = filename;
00051 }
00052
00053 void CollectorDatafileDefaultImpl1::setAddValueHandler(
    CollectorAddValueHandler addValueHandler) {
00054
00055 }
00056
00057 void CollectorDatafileDefaultImpl1::setClearHandler(
    CollectorClearHandler clearHandler) {
00058
00059 }

```

7.43 CollectorDatafileDefaultImpl1.h File Reference

```
#include <string>
#include "CollectorDatafile_if.h"
```

Classes

- class [CollectorDatafileDefaultImpl1](#)

7.44 CollectorDatafileDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  CollectorDatafileDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 1 de Agosto de 2018, 20:58
00012 */
00013
00014 #ifndef COLLECTORDATAFILEDEFAULTIMPL1_H
00015 #define COLLECTORDATAFILEDEFAULTIMPL1_H

```

```

00016
00017 #include <string>
00018
00019 #include "CollectorDatafile_if.h"
00020
00021 class CollectorDatafileDefaultImpl1 : public
00022   CollectorDatafile_if {
00023   public:
00024     CollectorDatafileDefaultImpl1();
00025     virtual ~CollectorDatafileDefaultImpl1() = default;
00026   public: // inherited from Collector_if
00027     void clear();
00028     void addValue(double value);
00029     double getLastValue();
00030     unsigned long numElements();
00031   public:
00032     double getValue(unsigned int num);
00033     double getNextValue();
00034     void seekFirstValue();
00035     std::string getDataFilename();
00036   public:
00037     void setAddValueHandler(CollectorAddValueHandler
00038      .addValueHandler);
00039     void setClearHandler(CollectorClearHandler clearHandler);
00040   private:
00041     std::string _filename;
00042   };
00043 #endif /* COLLECTORDATAFILEDEFAULTIMPL1_H */
00044

```

7.45 CollectorDefaultImpl1.cpp File Reference

```
#include "CollectorDefaultImpl1.h"
```

7.46 CollectorDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  CollectorDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 1 de Agosto de 2018, 20:43
00012 */
00013
00014 #include "CollectorDefaultImpl1.h"
00015
00016 CollectorDefaultImpl1::CollectorDefaultImpl1() {
00017 }
00018
00019
00020 void CollectorDefaultImpl1::clear() {
00021   _numElements = 0;
00022   if (_clearHandler != nullptr) {
00023     _clearHandler();
00024   }
00025 }
00026
00027 void CollectorDefaultImpl1::addValue(double value) {
00028   _lastValue = value;
00029   _numElements++;
00030   if (_addValueHandler != nullptr) {
00031     _addValueHandler(value);
00032   }
00033 }
00034
00035 double CollectorDefaultImpl1::getLastValue() {
00036   return this->_lastValue;
00037 }
00038

```

```

00039 unsigned long CollectorDefaultImpl1::numElements() {
00040     return this->_numElements;
00041 }
00042
00043 void CollectorDefaultImpl1::setAddValueHandler(
    CollectorAddValueHandler addValueHandler) {
00044     _addValueHandler = addValueHandler;
00045 }
00046
00047 void CollectorDefaultImpl1::setClearHandler(
    CollectorClearHandler clearHandler) {
00048     _clearHandler = clearHandler;
00049 }

```

7.47 CollectorDefaultImpl1.h File Reference

```
#include "Collector_if.h"
```

Classes

- class [CollectorDefaultImpl1](#)

7.48 CollectorDefaultImpl1.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: CollectorDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 1 de Agosto de 2018, 20:43
00012 */
00013
00014 #ifndef COLLECTORDEFAULTIMPL1_H
00015 #define COLLECTORDEFAULTIMPL1_H
00016
00017 #include "Collector_if.h"
00018
00019 class CollectorDefaultImpl1 : public Collector_if {
00020 public:
00021     CollectorDefaultImpl1();
00022     virtual ~CollectorDefaultImpl1() = default;
00023 public:
00024     void clear();
00025     void addValue(double value);
00026     double getLastValue();
00027     unsigned long numElements();
00028 public:
00029     void setAddValueHandler(CollectorAddValueHandler
        addValueHandler);
00030     void setClearHandler(CollectorClearHandler clearHandler);
00031 private:
00032     double _lastValue;
00033     unsigned long _numElements = 0;
00034     CollectorAddValueHandler _addValueHandler = nullptr;
00035     CollectorClearHandler _clearHandler = nullptr;
00036 };
00037
00038 #endif /* COLLECTORDEFAULTIMPL1_H */
00039

```

7.49 ComponentManager.cpp File Reference

```

#include "ComponentManager.h"
#include "List.h"
#include "Model.h"

```

7.50 ComponentManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ComponentManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 28 de Maio de 2019, 10:41
00012 */
00013
00014 #include "ComponentManager.h"
00015 #include "List.h"
00016 #include "Model.h"
00017
00018 ComponentManager::ComponentManager(Model* model) {
00019     _parentModel = model;
00020     _components = new List<ModelComponent*>();
00021     _components->setSortFunc([](const ModelComponent* a, const
00022         ModelComponent * b) {
00023         return a->id() < b->id();
00024     });
00025
00026 bool ComponentManager::insert(ModelComponent* comp) {
00027     if (_components->find(comp) == _components->list()->end()) {
00028         _components->insert(comp);
00029         _parentModel->tracer()->trace(Util::TraceLevel::componentResult
00030         , "Component \" " + comp->name() + "\" successfully inserted");
00031         _hasChanged = true;
00032         return true;
00033     }
00034     _parentModel->tracer()->trace(Util::TraceLevel::componentResult
00035     , "Component \" " + comp->name() + "\" could not be inserted");
00036     return false;
00037 }
00038
00039 void ComponentManager::remove(ModelComponent* comp) {
00040     _components->remove(comp);
00041     _parentModel->tracer()->trace(Util::TraceLevel::componentResult
00042     , "Component \" " + comp->name() + "\" successfully removed");
00043     _hasChanged = true;
00044 }
00045
00046 void ComponentManager::clear() {
00047     this->_components->clear();
00048     _hasChanged = true;
00049 }
00050
00051 //ModelComponent* ComponentManager::getComponent(Util::identification id) {
00052 //}
00053
00054 unsigned int ComponentManager::numberOfComponents() {
00055     return _components->size();
00056 }
00057
00058 std::list<ModelComponent*>::iterator ComponentManager::begin() {
00059     return _components->list()->begin();
00060 }
00061
00062 std::list<ModelComponent*>::iterator ComponentManager::end() {
00063     return _components->list()->end();
00064 }
00065
00066 ModelComponent* ComponentManager::front() {
00067     return _components->front();
00068 }
00069
00070 ModelComponent* ComponentManager::next() {
00071     return _components->next();
00072 }
00073
00074 bool ComponentManager::hasChanged() const {
00075     return _hasChanged;
00076 }
00077
00078 void ComponentManager::setHasChanged(bool _hasChanged) {
00079     this->_hasChanged = _hasChanged;
00080 }

```

7.51 ComponentManager.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [ComponentManager](#)

7.52 ComponentManager.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: ComponentManager.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 28 de Maio de 2019, 10:41
00012 */
00013
00014 #ifndef COMPONENTMANAGER_H
00015 #define COMPONENTMANAGER_H
00016
00017 #include "ModelComponent.h"
00018
00019 //class Model;
00020
00021 class ComponentManager {
00022 public:
00023     ComponentManager(Model* model);
00024     virtual ~ComponentManager() = default;
00025 public:
00026     bool insert(ModelComponent* comp);
00027     void remove(ModelComponent* comp);
00028     void clear();
00029     //bool check(ModelComponent* comp, std::string expressionName, std::string* errorMessage);
00030     //bool check(std::string compName, std::string expressionName, bool mandatory, std::string*
00031     //errorMessage);
00032     public:
00033         //ModelComponent* getComponent(Util::identification id);
00034         //ModelComponent* getComponent(std::string name);
00035         unsigned int numberComponents();
00036         std::list<ModelComponent*>::iterator begin();
00037         std::list<ModelComponent*>::iterator end();
00038         ModelComponent* front();
00039         ModelComponent* next();
00040         bool hasChanged() const;
00041         void setHasChanged(bool _hasChanged);
00042         //int getRankOf(std::string name);
00043 public:
00044     List<ModelComponent*>* _components;
00045     Model* _parentModel;
00046     bool _hasChanged = false;
00047     std::list<ModelComponent*>::iterator _componentIterator;
00048 };
00049
00050 #endif /* COMPONENTMANAGER_H */
```

7.53 ConnectionManager.cpp File Reference

```
#include "ConnectionManager.h"
```

7.54 ConnectionManager.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ConnectionManager.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 1 de Julho de 2019, 18:39
00012 */
00013
00014 #include "ConnectionManager.h"
00015
00016 ConnectionManager::ConnectionManager() {
00017 }
00018
00019
00020 unsigned int ConnectionManager::size() {
00021     return _nextConnections->size();
00022 }
00023
00024 ModelComponent* ConnectionManager::front() {
00025     return _nextConnections->front()->first;
00026 }
00027
00028 ModelComponent* ConnectionManager::atRank(unsigned int rank) {
00029     return _nextConnections->getAtRank(rank)->first;
00030 }
00031
00032 Connection* ConnectionManager::frontConnection() {
00033     return _nextConnections->front();
00034 }
00035
00036 Connection* ConnectionManager::connectionAtRank(unsigned int
rank) {
00037     return _nextConnections->getAtRank(rank);
00038 }
00039
00040 void ConnectionManager::insert(ModelComponent* component, unsigned
int inputNumber) {
00041     _nextConnections->insert(new Connection(component, inputNumber));
00042 }
00043
00044 std::list<Connection*>* ConnectionManager::list() const {
00045     return _nextConnections->list();
00046 }

```

7.55 ConnectionManager.h File Reference

```
#include <utility>
#include "List.h"
```

Classes

- class [ConnectionManager](#)

Typedefs

- [typedef std::pair< ModelComponent *, unsigned int > Connection](#)

7.55.1 Typedef Documentation

7.55.1.1 Connection

```
typedef std::pair<ModelComponent*, unsigned int> Connection
```

Definition at line 20 of file [ConnectionManager.h](#).

7.56 ConnectionManager.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ConnectionManager.h
00009  * Author: rlcancian
00010 *
00011  * Created on 1 de Julho de 2019, 18:39
00012 */
00013
00014 #ifndef CONNECTIONMANAGER_H
00015 #define CONNECTIONMANAGER_H
00016
00017 #include <utility>
00018 #include "List.h"
00019
00020 class ModelComponent;
00021
00022 typedef std::pair<ModelComponent*, unsigned int> Connection;
00023
00024 class ConnectionManager {
00025 public:
00026     ConnectionManager();
00027     virtual ~ConnectionManager() = default;
00028 public:
00029     unsigned int size();
00030     ModelComponent* front();
00031     ModelComponent* atRank(unsigned int rank);
00032     Connection* frontConnection();
00033     Connection* connectionAtRank(unsigned int rank);
00034     void insert(ModelComponent* component, unsigned int inputNumber = 0);
00035     std::list<Connection*>* list() const;
00036 private:
00037     List<Connection*>* _nextConnections = new List<Connection*>();
00038 };
00039
00040 #endif /* CONNECTIONMANAGER_H */
00041
```

7.57 Counter.cpp File Reference

```
#include "Counter.h"
#include "SimulationResponse.h"
#include "Model.h"
```

7.58 Counter.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: CounterDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 29 de Maio de 2019, 11:24
```

```

00012  */
00013
00014 #include "Counter.h"
00015 #include "SimulationResponse.h"
00016 #include "Model.h"
00017
00018 Counter::Counter(Model* model, std::string name,
00019   ModelElement* parent) : ModelElement(model, Util::TypeOf<
00020     Counter>(), name) {
00021   _parent = parent;
00022   _addSimulationResponse();
00023 }
00024 void Counter::_addSimulationResponse() {
00025   GetterMember getterMember = DefineGetterMember<Counter>(this, &
00026     Counter::getCountValue);
00027   std::string parentName = "";
00028   if (_parent != nullptr)
00029     parentName = _parent->name();
00030   SimulationResponse* resp = new SimulationResponse(
00031     Util::TypeOf<Counter>(), parentName+":"+_name, getterMember);
00032   _parentModel->responses()->insert(resp);
00033 }
00034 std::string Counter::show() {
00035   return ModelElement::show() +
00036     ", count=" + std::to_string(this->_count);
00037 }
00038 void
00039 Counter::clear() {
00040   _count = 0;
00041 }
00042
00043 void Counter::incCountValue(int value) {
00044   _count += value;
00045 }
00046
00047 unsigned long Counter::getCountValue() const {
00048   return _count;
00049 }
00050
00051 ModelElement* Counter::getParent() const {
00052   return _parent;
00053 }
00054
00055 PluginInformation* Counter::GetPluginInformation() {
00056   PluginInformation* info = new PluginInformation(Util::TypeOf<Counter>
00057   (), &Counter::LoadInstance);
00058   info->setGenerateReport(true);
00059   return info;
00060 }
00061 ModelElement* Counter::LoadInstance(Model* model,
00062   std::map<std::string, std::string>* fields) {
00063   Counter* newElement = new Counter(model);
00064   try {
00065     newElement->_loadInstance(fields);
00066   } catch (const std::exception& e) {
00067   }
00068   return newElement;
00069 }
00070
00071 bool Counter::_loadInstance(std::map<std::string, std::string>* fields) {
00072   return ModelElement::_loadInstance(fields);
00073 }
00074
00075 std::map<std::string, std::string>* Counter::_saveInstance() {
00076   return ModelElement::_saveInstance();
00077 }
00078 //std::list<std::map<std::string, std::string>>* _saveInstance(std::string type){}
00079
00080 bool Counter::_check(std::string* errorMessage) {
00081   return true;
00082 }

```

7.59 Counter.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
```

```
#include "Plugin.h"
```

Classes

- class Counter

7.60 Counter.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: CounterDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 29 de Maio de 2019, 11:24
00012 */
00013
00014 #ifndef COUNTERDEFAULTIMPL1_H
00015 #define COUNTERDEFAULTIMPL1_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020
00021 class Counter : public ModelElement {
00022 public:
00023     Counter(Model* model, std::string name="", ModelElement* parent=nullptr);
00024     virtual ~Counter() = default;
00025 public:
00026     virtual std::string show();
00027 public:
00028     static PluginInformation* GetPluginInformation();
00029     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00030                                         std::string>* fields);
00031 public:
00032     void clear();
00033     void incCountValue(int value = 1);
00034     unsigned long getCountValue() const;
00035     ModelElement* getParent() const;
00036 protected: // from ModelElement
00037     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00038     virtual std::map<std::string, std::string>* _saveInstance();
00039     virtual bool _check(std::string* errorMessage);
00040 protected:
00041     void _addSimulationResponse();
00042 private:
00043     ModelElement* _parent;
00044     unsigned long _count = 0;
00045 };
00046
00047 */
00048 #endif /* COUNTERDEFAULTIMPL1_H */
00049
```

7.61 Create.cpp File Reference

```
#include "Create.h"
#include "Model.h"
#include "EntityType.h"
#include "ElementManager.h"
#include "Attribute.h"
#include "Assign.h"
```

7.62 Create.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Create.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 20:12
00012 */
00013
00014 #include "Create.h"
00015 #include "Model.h"
00016 #include "EntityType.h"
00017 #include "ElementManager.h"
00018 #include "Attribute.h"
00019 #include "Assign.h"
00020
00021 Create::Create(Model* model, std::string name) :
    SourceModelComponent(model, Util::TypeOf<Create>(), name) {
00022     _numberOut = new Counter(_parentModel, "Count number in", this);
00023     _parentModel->elements()->insert(_numberOut);
00024     GetterMember getter = DefineGetterMember<SourceModelComponent>(this, &
        Create::entitiesPerCreation);
00025     SetterMember setter = DefineSetterMember<SourceModelComponent>(this, &
        Create::setEntitiesPerCreation);
00026     model->controls()->insert(new SimulationControl(Util::TypeOf<Create>(),
        "Entities Per Creation", getter, setter));
00027     /*
00028      model->getControls()->insert(new SimulationControl(Util::TypeOf<Create>(), "Time Between Creations",
00029          DefineGetterMember<SourceModelComponent>(this, &Create::getTimeBetweenCreationsExpression),
00030          DefineSetterMember<SourceModelComponent>(this, &Create::setTimeBetweenCreationsExpression))
00031      );
00032     */
00033 }
00034
00035
00036 std::string Create::show() {
00037     return SourceModelComponent::show();
00038 }
00039
00040 void Create::_execute(Entity* entity) {
00041     double tnow = _parentModel->simulation()->
        simulatedTime();
00042     entity->attributeValue("Entity.ArrivalTime", tnow); //
        ->find("Entity.ArrivalTime")->second->setValue(tnow);
00043     //entity->setAttributeValue("Entity.Picture", 1); //
        ->find("Entity.ArrivalTime")->second->setValue(tnow);
00044     double timeBetweenCreations, timeScale, newArrivalTime;
00045     unsigned int _maxCreations = _parentModel->parseExpression(this->
        _maxCreationsExpression);
00046     for (unsigned int i = 0; i<this->entitiesPerCreation; i++) {
00047         if (_entitiesCreatedSoFar < _maxCreations) {
00048             _entitiesCreatedSoFar++;
00049             Entity* newEntity = new Entity(_parentModel);
00050             newEntity->setEntityType(entity->entityType());
00051             _parentModel->elements()->insert(newEntity); //
        ->getEntities()->insert(newEntity);
00052             timeBetweenCreations = _parentModel->parseExpression(this->
        _timeBetweenCreationsExpression);
00053             timeScale = Util::TimeUnitConvert(this->
        _timeBetweenCreationsTimeUnit, _parentModel->
        infos()->replicationLengthTimeUnit());
00054             newArrivalTime = tnow + timeBetweenCreations*timeScale;
00055             Event newEvent = new Event(newArrivalTime, newEntity, this);
00056             _parentModel->futureEvents()->insert(newEvent);
00057             _parentModel->tracer()->trace("Arrival of entity " + std::to_string(
        newEntity->entityNumber()) + " schedule for time " + std::to_string(newArrivalTime));
00058             //_model->getTrace()->trace("Arrival of entity "+std::to_string(entity->getId()) + " schedule for
        time " +std::to_string(newArrivalTime));
00059         }
00060     }
00061     _numberOut->incCountValue();
00062     _parentModel->sendEntityToComponent(entity, this->
        nextComponents()->frontConnection(), 0.0);
00063 }
00064
00065 PluginInformation* Create::GetPluginInformation() {
00066     PluginInformation* info = new PluginInformation(Util::TypeOf<Create>(
        ), &Create::LoadInstance);
00067     info->setSource(true);
00068     info->insertDynamicLibFileDependence("attribute.so");

```

```

00069     info->insertDynamicLibFileDependence("entitytype.so");
00070     info->insertDynamicLibFileDependence("statisticscollector.so");
00071     return info;
00072 }
00073
00074 ModelComponent* Create::LoadInstance(Model* model,
00075     std::map<std::string, std::string>* fields) {
00076     Create* newComponent = new Create(model);
00077     try {
00078         newComponent->_loadInstance(fields);
00079     } catch (const std::exception& e) {
00080     }
00081     return newComponent;
00082 }
00083
00084 bool Create::_loadInstance(std::map<std::string, std::string>* fields) {
00085     return SourceModelComponent::_loadInstance(fields);
00086 }
00087
00088 void Create::_initBetweenReplications() {
00089     SourceModelComponent::_initBetweenReplications();
00090 }
00091
00092 std::map<std::string, std::string>* Create::_saveInstance() {
00093     std::map<std::string, std::string>* fields =
00094         SourceModelComponent::_saveInstance();
00095     return fields;
00096 }
00097 bool Create::_check(std::string* errorMessage) {
00098     bool resultAll = SourceModelComponent::_check(errorMessage);
00099     return resultAll;
00100 }
```

7.63 Create.h File Reference

```
#include <string>
#include <limits>
#include "SourceModelComponent.h"
#include "EntityType.h"
#include "Counter.h"
#include "Plugin.h"
```

Classes

- class [Create](#)

7.64 Create.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Create.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 21 de Junho de 2018, 20:12
00012 */
00013
00014 #ifndef CREATE_H
00015 #define CREATE_H
00016
00017 #include <string>
00018 #include <limits>
00019 #include "SourceModelComponent.h"
00020 #include "EntityType.h"
```

```

00021 #include "Counter.h"
00022 #include "Plugin.h"
00023
00024
00068 class Create : public SourceModelComponent {
00069 public:
00070     Create(Model* model, std::string name = "");
00071     virtual ~Create() = default;
00072 public:
00073     virtual std::string show();
00074 public:
00075     static PluginInformation* GetPluginInformation();
00076     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00077                                         std::string>* fields);
00077 protected:
00078     virtual void _execute(Entity* entity);
00079     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00080     virtual void _initBetweenReplications();
00081     virtual std::map<std::string, std::string>* _saveInstance();
00082     virtual bool _check(std::string* errorMessage);
00083 private:
00084     Counter* _numberOut;
00085 };
00086
00087 #endif /* CREATE_H */

```

7.65 Decide.cpp File Reference

```
#include "Decide.h"
#include "Model.h"
```

7.66 Decide.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Decide.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 9 de Agosto de 2018, 20:39
00012 */
00013
00014 #include "Decide.h"
00015 #include "Model.h"
00016
00017 Decide::Decide(Model* model, std::string name) :
00018     ModelComponent(model, Util::TypeOf<Decide>(), name) {
00019
00020 List<std::string>* Decide::getConditions() const {
00021     return _conditions;
00022 }
00023
00024
00025 std::string Decide::show() {
00026     return ModelComponent::show() + "";
00027 }
00028
00029 void Decide::_execute(Entity* entity) {
00030     double value;
00031     unsigned short i = 0;
00032     for (std::list<std::string>::iterator it = _conditions->list() ->begin(); it != _conditions->
00033         list() ->end(); it++) {
00034         value = _parentModel->parseExpression((*it));
00035         _parentModel->tracer()->traceSimulation(
00036             _parentModel->simulation()->simulatedTime(), entity, this,
00037             std::to_string(i + 1) + "th condition evaluated to " + std::to_string(value) + " // " + (*it));
00038         if (value) {
00039             _parentModel->sendEntityToComponent(entity, this->
00040                 nextComponents() ->connectionAtRank(i), 0.0);
00041         }
00042     }
00043 }
```

```

00039     i++;
00040 }
00041     _parentModel->tracer()->traceSimulation(
00042         _parentModel->simulation()->simulatedTime(), entity, this, "No condition
00043         has been evaluated true");
00044     _parentModel->sendEntityToComponent(entity, this->
00045         nextComponents()->connectionAtRank(i), 0.0);
00046 }
00047
00048 void Decide::_initBetweenReplications() {
00049
00050     bool res = ModelComponent::_loadInstance(fields);
00051     if (res) {
00052         unsigned int nv = std::stoi((*(fields->find("conditions"))).second);
00053         for (unsigned int i = 0; i < nv; i++) {
00054             this->_conditions->insert((*(fields->find("condition" + std::to_string(i)))).second);
00055         }
00056     }
00057     return res;
00058 }
00059 std::map<std::string, std::string>* Decide::_saveInstance() {
00060     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00061     (); //Util::TypeOf<Decide>());
00062     unsigned short i = 0;
00063     fields->emplace("conditions", std::to_string(_conditions->size()));
00064     for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->
00065         list()->end(); it++) {
00066         fields->emplace("condition" + std::to_string(i++), (*it));
00067     }
00068     return fields;
00069 }
00070     bool Decide::_check(std::string* errorMessage) {
00071     bool allResult = true;
00072     std::string condition;
00073     for (std::list<std::string>::iterator it = _conditions->list()->begin(); it != _conditions->
00074         list()->end(); it++) {
00075         condition = (*it);
00076         allResult &= _parentModel->checkExpression(condition, "condition",
00077             errorMessage);
00078     }
00079     return allResult;
00080 }
00081 PluginInformation* Decide::GetPluginInformation() {
00082     PluginInformation* info = new PluginInformation(Util::TypeOf<Decide>(
00083         ), &Decide::LoadInstance);
00084     return info;
00085 }
00086
00087 ModelComponent* Decide::LoadInstance(Model* model,
00088     std::map<std::string, std::string>* fields) {
00089     Decide* newComponent = new Decide(model);
00090     try {
00091         newComponent->_loadInstance(fields);
00092     } catch (const std::exception& e) {
00093     }
00094     return newComponent;
00095 }
```

7.67 Decide.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Decide](#)

7.68 Decide.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Decide.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 9 de Agosto de 2018, 20:39
00012 */
00013
00014 #ifndef DECIDE_H
00015 #define DECIDE_H
00016
00017 #include "ModelComponent.h"
00018
00073 class Decide : public ModelComponent {
00074 public:
00075     Decide(Model* model, std::string name = "");
00076     virtual ~Decide() = default;
00077 public:
00078     List<std::string>* getConditions() const;
00079
00080 public:
00081     virtual std::string show();
00082 public:
00083     static PluginInformation* GetPluginInformation();
00084     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00085                                         std::string>* fields);
00085 protected:
00086     virtual void _execute(Entity* entity);
00087     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00088     virtual void _initBetweenReplications();
00089     virtual std::map<std::string, std::string>* _saveInstance();
00090     virtual bool _check(std::string* errorMessage);
00091 private:
00092     List<std::string>* _conditions = new List<std::string>();
00093 private:
00094
00095 };
00096
00097 #endif /* DECIDE_H */
00098

```

7.69 DefaultTraceAndEventHandlers.h File Reference

7.70 DefaultTraceAndEventHandlers.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: DefaultTraceAndEventHandlers.h
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 16:12
00012 */
00013
00014 #ifndef DEFAULTTRACEANDEVENTHANDLERS_H
00015 #define DEFAULTTRACEANDEVENTHANDLERS_H
00016
00017 /*
00018 #include "TraceManager.h"
00019 #include "OnEventManager.h"
00020 #include <iostream>
00021
00022 // default Trace Handlers
00023 void traceHandlerFunction(TraceEvent e) {
00024     std::cout << e.getText() << std::endl;
00025 }
00026
00027 void traceSimulationHandlerFunction(TraceSimulationEvent e) {

```

```

00028     std::cout << e.getText() << std::endl;
00029 }
00030
00031 void onSimulationStartHandlerFunction(SimulationEvent* re) {
00032     //std::cout << "(Handler) Simulation is starting" << std::endl;
00033 }
00034
00035
00036 // default Event Handlers
00037 void onReplicationStartHandlerFunction(SimulationEvent* re) {
00038     // std::cout << "(Handler) Replication " << re->getReplicationNumber() << " starting." << std::endl;
00039 }
00040
00041 void onProcessEventHandlerFunction(SimulationEvent* re) {
00042     //std::cout << "(Handler) Processing event " << re->getEventProcessed()->show() << std::endl;
00043 }
00044
00045 void onReplicationEndHandlerFunction(SimulationEvent* re) {
00046     //std::cout << "(Handler) Replication " << re->getReplicationNumber() << " ending." << std::endl;
00047 }
00048
00049 void onEntityRemoveHandlerFunction(SimulationEvent* re) {
00050     //std::cout << "(Handler) Entity " << re->getEventProcessed()->getEntity() << " was removed." <<
00051     std::endl;
00052 */
00053 #endif /* DEFAULTTRACEANDEVENTHANDLERS_H */
00054

```

7.71 DefineGetterSetter.h File Reference

```
#include <functional>
#include "Util.h"
```

Typedefs

- `typedef std::function< double() > GetterMember`
- `typedef std::function< void(double) > SetterMember`

Functions

- `template<typename Class >`
`GetterMember DefineGetterMember (Class *object, double(Class::*function)())`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(double))`
- `template<typename Class >`
`GetterMember DefineGetterMember (Class *object, unsigned int(Class::*function)() const)`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(unsigned int))`
- `template<typename Class >`
`GetterMember DefineGetterMember (Class *object, unsigned long(Class::*function)() const)`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(unsigned long))`
- `template<typename Class >`
`GetterMember DefineGetterMember (Class *object, bool(Class::*function)() const)`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(bool))`
- `template<typename Class >`
`GetterMember DefineGetterMember (Class *object, std::string(Class::*function)() const)`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(std::string) const)`
- `template<typename Class >`
`GetterMember DefineGetterMember (Class *object, Util::TimeUnit(Class::*function)() const)`
- `template<typename Class >`
`SetterMember DefineSetterMember (Class *object, void(Class::*function)(Util::TimeUnit))`

7.71.1 Typedef Documentation

7.71.1.1 GetterMember

```
typedef std::function<double() > GetterMember
```

Definition at line 20 of file [DefineGetterSetter.h](#).

7.71.1.2 SetterMember

```
typedef std::function<void(double) > SetterMember
```

Definition at line 21 of file [DefineGetterSetter.h](#).

7.71.2 Function Documentation

7.71.2.1 DefineGetterMember() [1/6]

```
template<typename Class >
GetterMember DefineGetterMember (
    Class * object,
    double(Class::*)( ) function )
```

Definition at line 26 of file [DefineGetterSetter.h](#).

7.71.2.2 DefineGetterMember() [2/6]

```
template<typename Class >
GetterMember DefineGetterMember (
    Class * object,
    unsigned int(Class::*)( ) const function )
```

Definition at line 43 of file [DefineGetterSetter.h](#).

7.71.2.3 DefineGetterMember() [3/6]

```
template<typename Class >
GetterMember DefineGetterMember (
    Class * object,
    unsigned long(Class::*)( ) const function )
```

Definition at line 56 of file [DefineGetterSetter.h](#).

7.71.2.4 DefineGetterMember() [4/6]

```
template<typename Class >
GetterMember DefineGetterMember (
    Class * object,
    bool(Class::*)() const function )
```

Definition at line 68 of file [DefineGetterSetter.h](#).

7.71.2.5 DefineGetterMember() [5/6]

```
template<typename Class >
GetterMember DefineGetterMember (
    Class * object,
    std::string(Class::*)() const function )
```

Definition at line 80 of file [DefineGetterSetter.h](#).

7.71.2.6 DefineGetterMember() [6/6]

```
template<typename Class >
GetterMember DefineGetterMember (
    Class * object,
    Util::TimeUnit(Class::*)() const function )
```

Definition at line 92 of file [DefineGetterSetter.h](#).

7.71.2.7 DefineSetterMember() [1/6]

```
template<typename Class >
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*)(double) function )
```

Definition at line 36 of file [DefineGetterSetter.h](#).

7.71.2.8 DefineSetterMember() [2/6]

```
template<typename Class >
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*)(unsigned int) function )
```

Definition at line 48 of file [DefineGetterSetter.h](#).

7.71.2.9 DefineSetterMember() [3/6]

```
template<typename Class >
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*) (unsigned long) function )
```

Definition at line 61 of file [DefineGetterSetter.h](#).

7.71.2.10 DefineSetterMember() [4/6]

```
template<typename Class >
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*) (bool) function )
```

Definition at line 73 of file [DefineGetterSetter.h](#).

7.71.2.11 DefineSetterMember() [5/6]

```
template<typename Class >
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*) (std::string) const function )
```

Definition at line 85 of file [DefineGetterSetter.h](#).

7.71.2.12 DefineSetterMember() [6/6]

```
template<typename Class >
SetterMember DefineSetterMember (
    Class * object,
    void(Class::*) (Util::TimeUnit) function )
```

Definition at line 97 of file [DefineGetterSetter.h](#).

7.72 DefineGetterSetter.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: DefineGetterSetter.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 5 de Agosto de 2018, 13:52
00012 */
00013
00014 #ifndef DEFINEGETTERSETTER_H
00015 #define DEFINEGETTERSETTER_H
00016
00017 #include <functional>
00018 #include "Util.h"
00019
00020 typedef std::function<double() > GetterMember;
00021 typedef std::function<void(double) > SetterMember;
00022
00023 // double
00024
00025 template<typename Class>
00026 GetterMember DefineGetterMember(Class * object, double (Class::*function)())
00027 {
00028     return std::bind(function, object);
00029 }
00030
00031 template<typename Class>
00032 GetterMember DefineGetterMember(Class * object, double (Class::*function)()
00033 const) {
00034     return std::bind(function, object);
00035 }
00036
00037 template<typename Class>
00038 SetterMember DefineSetterMember(Class * object, void (Class::*function)(
00039 double)) {
00040     return std::bind(function, object, std::placeholders::_1);
00041 }
00042 // unsigned int
00043
00044 template<typename Class>
00045 GetterMember DefineGetterMember(Class * object, unsigned int (Class::*function)()
00046 const) {
00047     return std::bind(function, object);
00048 }
00049
00050 template<typename Class>
00051 SetterMember DefineSetterMember(Class * object, void (Class::*function)(
00052 unsigned int)) {
00053     return std::bind(function, object, std::placeholders::_1);
00054 }
00055 // unsigned long
00056
00057 template<typename Class>
00058 GetterMember DefineGetterMember(Class * object, unsigned long (Class::*function)()
00059 const) {
00060     return std::bind(function, object);
00061 }
00062
00063 template<typename Class>
00064 SetterMember DefineSetterMember(Class * object, void (Class::*function)(
00065 unsigned long)) {
00066     return std::bind(function, object, std::placeholders::_1);
00067 }
00068 // bool
00069
00070 template<typename Class>
00071 GetterMember DefineGetterMember(Class * object, bool (Class::*function)()
00072 const) {
00073     return std::bind(function, object);
00074 }
00075
00076 template<typename Class>
00077 SetterMember DefineSetterMember(Class * object, void (Class::*function)(bool)
00078 ) {
00079     return std::bind(function, object, std::placeholders::_1);
00080 }
```

```

00076
00077 //std::string
00078
00079 template<typename Class>
00080 GetterMember DefineGetterMember(Class * object, std::string(Class::*function)
00081 () const) {
00082     return std::bind(function, object);
00083 }
00084 template<typename Class>
00085 SetterMember DefineSetterMember(Class * object, void (Class::*function)(
00086 std::string) const) {
00087     return std::bind(function, object, std::placeholders::_1);
00088 }
00089 // Util::TimeUnit
00090
00091 template<typename Class>
00092 GetterMember DefineGetterMember(Class * object,
00093 Util::TimeUnit(Class::*function)() const) {
00094     return std::bind(function, object);
00095 }
00096 template<typename Class>
00097 SetterMember DefineSetterMember(Class * object, void (Class::*function)(
00098 Util::TimeUnit)) {
00099     return std::bind(function, object, std::placeholders::_1);
00100 }
00101
00102 #endif /* DEFINEGETTERSETTER_H */
00103

```

7.73 Delay.cpp File Reference

```

#include "Delay.h"
#include "Model.h"
#include "Attribute.h"

```

7.74 Delay.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Delay.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 19:49
00012 */
00013
00014 #include "Delay.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 Delay::Delay(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Delay>(), name) {
00020
00021     std::string Delay::show() {
00022         return ModelComponent::show() +
00023             ",delayExpression=" + this->_delayExpression +
00024             ",timeUnit=" + std::to_string(static_cast<int> (this->_delayTimeUnit));
00025     }
00026
00027     ModelComponent* Delay::LoadInstance(Model* model,
00028         std::map<std::string, std::string>* fields) {
00029         Delay* newComponent = new Delay(model);
00030         try {
00031             newComponent->_loadInstance(fields);
00032         } catch (const std::exception& e) {
00033

```

```

00033     }
00034     return newComponent;
00035 }
00036
00037 void Delay::setDelayExpression(std::string _delayExpression) {
00038     this->_delayExpression = _delayExpression;
00039 }
00040
00041 std::string Delay::delayExpression() const {
00042     return _delayExpression;
00043 }
00044
00045 void Delay::setDelayTimeUnit(Util::TimeUnit _delayTimeUnit) {
00046     this->_delayTimeUnit = _delayTimeUnit;
00047 }
00048
00049 Util::TimeUnit Delay::delayTimeUnit() const {
00050     return _delayTimeUnit;
00051 }
00052
00053 void Delay::_execute(Entity* entity) {
00054     double waitTime = _parentModel->parseExpression(_delayExpression) *
00055         Util::TimeUnitConvert(_delayTimeUnit, _parentModel->
00056         infos()->replicationLengthTimeUnit());
00057     entity->entityType()->statisticsCollector(_name + "." + "Waiting_Time"
00058     )->getStatistics()->getCollector()->addValue(waitTime);
00059     entity->attributeValue("Entity.WaitTime", entity->
00060     attributeValue("Entity.WaitTime") + waitTime);
00061     double delayEndTime = _parentModel->simulation()->
00062     simulatedTime() + waitTime;
00063     Event* newEvent = new Event(delayEndTime, entity, this->
00064     nextComponents()->frontConnection());
00065     _parentModel->futureEvents()->insert(newEvent);
00066     _parentModel->tracer()->trace("End of delay of entity " + std::to_string(entity
00067     ->entityNumber()) + " scheduled to time " + std::to_string(delayEndTime));
00068 }
00069
00070 bool Delay::_loadInstance(std::map<std::string, std::string>* fields) {
00071     bool res = ModelComponent::_loadInstance(fields);
00072     if (res) {
00073         this->_delayExpression = (*fields->find("delayExpression")).second;
00074         this->_delayTimeUnit = static_cast<Util::TimeUnit> (std::stoi((*fields->find("
00075         delayExpressionTimeUnit"))).second));
00076     }
00077     return res;
00078 }
00079
00080 void Delay::_initBetweenReplications() {
00081 }
00082
00083 std::map<std::string, std::string>* Delay::_saveInstance() {
00084     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00085     ();
00086     //Util::TypeOf<Delay>());
00087     fields->emplace("delayExpression", this->_delayExpression);
00088     fields->emplace("delayExpressionTimeUnit", std::to_string(static_cast<int> (this->_delayTimeUnit)));
00089     return fields;
00090 }
00091
00092 bool Delay::_check(std::string* errorMessage) {
00093     //include attributes needed
00094     ElementManager* elements = _parentModel->elements();
00095     std::vector<std::string> neededNames = {"Entity.WaitTime"};
00096     std::string neededName;
00097     for (unsigned int i = 0; i < neededNames.size(); i++) {
00098         neededName = neededNames[i];
00099         if (elements->element(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00100             Attribute* attr1 = new Attribute(_parentModel, neededName);
00101             elements->insert(attr1);
00102         }
00103     }
00104     return _parentModel->checkExpression(_delayExpression, "Delay expression",
00105     errorMessage);
00106 }
00107
00108 void Delay::_createInternalElements() {
00109     // include StatisticsCollector needed in EntityType
00110     ElementManager* elements = _parentModel->elements();
00111     std::list<ModelElement*>* enttypes = elements->elementList(Util::TypeOf<EntityType>())->
00112     list();
00113     for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00114         EntityType* enttype = static_cast<EntityType*> ((*it));
00115         enttype->statisticsCollector(_name + "." + "Waiting_Time"); // force create
00116         this CStat before simulation starts
00117     }
00118 }
00119
00120 PluginInformation* Delay::GetPluginInformation() {
00121 }
```

```

00108     PluginInformation* info = new PluginInformation(Util::TypeOf<Delay>()
00109         , &Delay::LoadInstance);
00110     return info;
00110 }

```

7.75 Delay.h File Reference

```

#include <string>
#include "ModelComponent.h"
#include "Plugin.h"

```

Classes

- class [Delay](#)

7.76 Delay.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Delay.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 19:49
00012 */
00013
00014 #ifndef DELAY_H
00015 #define DELAY_H
00016
00017 #include <string>
00018 #include "ModelComponent.h"
00019 #include "Plugin.h"
00020
00041 class Delay : public ModelComponent {
00042 public:
00043     Delay(Model* model, std::string name(""));
00044     virtual ~Delay() = default;
00045 public:
00046     void setDelayExpression(std::string _delayExpression);
00047     std::string delayExpression() const;
00048     void setDelayTimeUnit(Util::TimeUnit _delayTimeUnit);
00049     Util::TimeUnit delayTimeUnit() const;
00050 public:
00051     virtual std::string show();
00052 public:
00053     static PluginInformation* GetPluginInformation();
00054     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00055                                         std::string>* fields);
00055 protected:
00056     virtual void _execute(Entity* entity);
00057     virtual void _initBetweenReplications();
00058     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00059     virtual std::map<std::string, std::string>* _saveInstance();
00060     virtual bool _check(std::string* errorMessage);
00061     virtual void _createInternalElements();
00062 private:
00063     std::string _delayExpression = "1.0";
00064     Util::TimeUnit _delayTimeUnit = Util::TimeUnit::second;
00065 };
00066
00067 #endif /* DELAY_H */
00068

```

7.77 Dispose.cpp File Reference

```
#include "Dispose.h"
#include "Model.h"
```

7.78 Dispose.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Dispose.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 20:13
00012 */
00013
00014 #include "Dispose.h"
00015 #include "Model.h"
00016
00017 Dispose::Dispose(Model* model, std::string name) :
00018     SinkModelComponent(model, Util::TypeOf<Dispose>(), name) {
00019     _numberOut = new Counter(_parentModel, _name + "." + "Count_number_out", this);
00020     _parentModel->elements()->insert(_numberOut);
00021
00022
00023 std::string Dispose::show() {
00024     return SinkModelComponent::show() +
00025         ",collectStatistics=" + std::to_string(this->_collectStatistics);
00026 }
00027
00028 void Dispose::_execute(Entity* entity) {
00029     _numberOut->incCountValue();
00030     if (_collectStatistics) {
00031         double timeInSystem = _parentModel->simulation()->
00032             simulatedTime() - entity->attributeValue("Entity.ArrivalTime");
00033         entity->entityType()->statisticsCollector(entity->
00034             entityType()->name() + "." + "Total_Time")->getStatistics()->getCollector()->addValue(
00035             timeInSystem);
00036     }
00037
00038     _parentModel->removeEntity(entity, this->
00039         isCollectStatistics());
00040 }
00041
00042 bool Dispose::_loadInstance(std::map<std::string, std::string>* fields) {
00043     return ModelComponent::_loadInstance(fields);
00044 }
00045
00046 std::map<std::string, std::string>* Dispose::_saveInstance() {
00047     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00048     ();
00049
00050 }
00051
00052 bool Dispose::_check(std::string* errorMessage) {
00053     //
00054     return true;
00055 }
00056
00057 void Dispose::_createInternalElements() {
00058     // include StatisticsCollector needed for each EntityType
00059     std::list<ModelElement*>* enttypes = _parentModel->elements()->
00060         elementList(Util::TypeOf<EntityType>())->list();
00061     for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {
00062         static_cast<EntityType*> ((*it))->statisticsCollector((*it)->name() + "." + "Total_Time"); //
00063         // force create this CStat before model checking
00064     }
00065 }
```

```

00065 PluginInformation* Dispose::GetPluginInformation() {
00066     PluginInformation* info = new PluginInformation(Util::TypeOf<Dispose>
00067 (), &Dispose::LoadInstance);
00068     info->setSink(true);
00069 }
00070
00071 ModelComponent* Dispose::LoadInstance(Model* model,
00072     std::map<std::string, std::string>* fields) {
00073     Dispose* newComponent = new Dispose(model);
00074     try {
00075         newComponent->_loadInstance(fields);
00076     } catch (const std::exception& e) {
00077     }
00078     return newComponent;
00079 }
00080 }
```

7.79 Dispose.h File Reference

```
#include "SinkModelComponent.h"
#include "Counter.h"
#include "Plugin.h"
```

Classes

- class [Dispose](#)

7.80 Dispose.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Dispose.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 20:13
00012 */
00013
00014 #ifndef DISPOSE_H
00015 #define DISPOSE_H
00016
00017 #include "SinkModelComponent.h"
00018 #include "Counter.h"
00019 #include "Plugin.h"
00020
00038 class Dispose : public SinkModelComponent {
00039 public:
00040     Dispose(Model* model, std::string name = "");
00041     virtual ~Dispose() = default;
00042 public:
00043     virtual std::string show();
00044 public:
00045     static PluginInformation* GetPluginInformation();
00046     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00047                                         std::string>* fields);
00047 protected:
00048     virtual void _execute(Entity* entity);
00049     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00050     virtual void _initBetweenReplications();
00051     virtual std::map<std::string, std::string>* _saveInstance();
00052     virtual bool _check(std::string* errorMessage);
00053     virtual void _createInternalElements();
00054 private:
00055     bool _collectStatistics = true;
00056 private:
00057     Counter* _numberOut;
00058 };
00059
00060 #endif /* DISPOSE_H */
00061
```

7.81 DropOff.cpp File Reference

```
#include "DropOff.h"
#include "Model.h"
```

7.82 DropOff.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 /*
00008  * File: DropOff.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #include "DropOff.h"
00015 #include "Model.h"
00016
00017 DropOff::DropOff(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<DropOff>(), name) {
00018 }
00019
00020
00021 std::string DropOff::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* DropOff::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00026     DropOff* newComponent = new DropOff(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030         }
00031     return newComponent;
00032 }
00033 }
00034
00035 void DropOff::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool DropOff::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void DropOff::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* DropOff::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
();
00053     //...
00054     return fields;
00055 }
00056
00057 bool DropOff::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* DropOff::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<DropOff>
(),
        &DropOff::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
```

7.83 DropOff.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [DropOff](#)

7.84 DropOff.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    DropOff.h
00009  * Author:  rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #ifndef DROPOFF_H
00015 #define DROPOFF_H
00016
00017 #include "ModelComponent.h"
00018
00041 class DropOff : public ModelComponent {
00042 public: // constructors
00043     DropOff(Model* model, std::string name = "");
00044     virtual ~DropOff() = default;
00045 public: // virtual
00046     virtual std::string show();
00047 public: // static
00048     static PluginInformation* GetPluginInformation();
00049     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00050                                         std::string>* fields);
00050 protected: // virtual
00051     virtual void _execute(Entity* entity);
00052     virtual void _initBetweenReplications();
00053     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00054     virtual std::map<std::string, std::string>* _saveInstance();
00055     virtual bool _check(std::string* errorMessage);
00056 private: // methods
00057 private: // attributes 1:1
00058 private: // attributes 1:n
00059 };
00060
00061
00062 #endif /* DROPOFF_H */
00063
```

7.85 Dummy.cpp File Reference

```
#include "Dummy.h"
#include "Model.h"
```

7.86 Dummy.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Dummy.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Maio de 2019, 18:41
00012 */
00013
00014 #include "Dummy.h"
00015 #include "Model.h"
00016
00017 Dummy::Dummy(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<Dummy>(), name) {
00019
00020
00021     std::string Dummy::show() {
00022         return ModelComponent::show() + "";
00023     }
00024
00025     ModelComponent* Dummy::LoadInstance(Model* model,
00026         std::map<std::string, std::string>* fields) {
00027         Dummy* newComponent = new Dummy(model);
00028         try {
00029             newComponent->_loadInstance(fields);
00030         } catch (const std::exception& e) {
00031
00032         return newComponent;
00033     }
00034
00035     void Dummy::_execute(Entity* entity) {
00036         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00037         entity forward");
00038         this->_parentModel->sendEntityToComponent(entity, this->
00039             nextComponents()->frontConnection(), 0.0);
00040
00041     bool Dummy::_loadInstance(std::map<std::string, std::string>* fields) {
00042         bool res = ModelComponent::_loadInstance(fields);
00043         if (res) {
00044             //...
00045         }
00046         return res;
00047
00048     void Dummy::_initBetweenReplications() {
00049
00050
00051     std::map<std::string, std::string>* Dummy::_saveInstance() {
00052         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053         ();
00054         //...
00055         return fields;
00056
00057     bool Dummy::_check(std::string* errorMessage) {
00058         bool resultAll = true;
00059         //...
00060         return resultAll;
00061     }
00062
00063     PluginInformation* Dummy::GetPluginInformation() {
00064         PluginInformation* info = new PluginInformation(Util::TypeOf<Dummy>()
00065         , &Dummy::LoadInstance);
00066         // ...
00067         return info;
00068     }

```

7.87 Dummy.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Dummy](#)

7.88 Dummy.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Dummy.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Maio de 2019, 18:41
00012 */
00013
00014 #ifndef DUMMY_H
00015 #define DUMMY_H
00016
00017 #include "ModelComponent.h"
00018
00022 class Dummy : public ModelComponent {
00023 public: // constructors
00024     Dummy(Model* model, std::string name = "");
00025     virtual ~Dummy() = default;
00026 public: // virtual
00027     virtual std::string show();
00028 public: // static
00029     static PluginInformation* GetPluginInformation();
00030     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00031                                         std::string>* fields);
00031 protected: // virtual
00032     virtual void _execute(Entity* entity);
00033     virtual void _initBetweenReplications();
00034     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00035     virtual std::map<std::string, std::string>* _saveInstance();
00036     virtual bool _check(std::string* errorMessage);
00037 private: // methods
00038 private: // attributes 1:1
00039 private: // attributes 1:n
00040 };
00041
00042 #endif /* DUMMY_H */
00043

```

7.89 ElementManager.cpp File Reference

```
#include "ElementManager.h"
#include "Model.h"
```

7.90 ElementManager.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    ElementManager.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 12:48
00012 */
00013
00014 #include "ElementManager.h"
00015 #include "Model.h"
00016

```

```

00017 ElementManager::ElementManager(Model* model) {
00018     _parentModel = model;
00019     /* \todo: -- Sort methods for elements should be a decorator */
00020     _elements = new std::map<std::string, List<ModelElement*>>();
00021     //_elements->setSortFunc([](const ModelElement* a, const ModelElement * b) {
00022     //    return a->getId() < b->getId();
00023     //));
00024
00025 }
00026
00027 bool ElementManager::insert(ModelElement* infra) {
00028     std::string infraTypename = infra->classname();
00029     bool res = insert(infraTypename, infra);
00030     if (res)
00031         _parentModel->tracer()->trace(Util::TraceLevel::elementResult
00032             , "Element successfully inserted");
00033     else
00034         _parentModel->tracer()->trace(Util::TraceLevel::elementResult
00035             , "Element could not be inserted");
00036     return res;
00037 }
00038
00039 bool ElementManager::insert(std::string infraTypename,
00040     ModelElement* infra) {
00041     List<ModelElement*>* listElements = elementList(infraTypename);
00042     if (listElements->find(infra) == listElements->list()->end()) { //not found
00043         listElements->insert(infra);
00044         this->_parentModel->tracer()->trace(
00045             Util::TraceLevel::toolDetailed, "Element \""
00046             + infra->name() + "\" successfully inserted.");
00047     return true;
00048     }
00049     return false;
00050 }
00051
00052 void ElementManager::remove(ModelElement* infra) {
00053     std::string infraTypename = infra->classname();
00054     List<ModelElement*>* listElements = elementList(infraTypename);
00055     listElements->remove(infra);
00056 }
00057
00058 void ElementManager::remove(std::string infraTypename,
00059     ModelElement* infra) {
00060     List<ModelElement*>* listElements = elementList(infraTypename);
00061     listElements->remove(infra);
00062 }
00063
00064 bool ElementManager::check(std::string infraTypename, std::string infraName,
00065     std::string expressionName, bool mandatory, std::string* errorMessage) {
00066     if (infraName == "" && !mandatory)
00067         return true;
00068     if (!result) {
00069         std::string msg = infraTypename + " \""
00070             + infraName + "\" for '\"" + expressionName + "' is not in the
00071             model.\"";
00072         errorMessage->append(msg);
00073     }
00074     return result;
00075 }
00076
00077 bool ElementManager::check(std::string infraTypename,
00078     ModelElement* infra, std::string expressionName, std::string* errorMessage) {
00079     bool result = infra != nullptr;
00080     if (!result) {
00081         std::string msg = infraTypename + " for '\"" + expressionName + "' is null.";
00082         errorMessage->append(msg);
00083     } else {
00084         result = check(infraTypename, infra->name(), expressionName, true, errorMessage);
00085     }
00086     return result;
00087 }
00088
00089 void ElementManager::clear() {
00090     this->_elements->clear();
00091 }
00092
00093 unsigned int ElementManager::numberOfElements(std::string infraTypename) {
00094     List<ModelElement*>* listElements = elementList(infraTypename);
00095     return listElements->size();
00096 }
00097
00098 unsigned int ElementManager::numberOfElements() {
00099     unsigned int total = 0;

```

```

00094     for (std::map<std::string, List<ModelElement*>>::iterator it = _elements->begin();
00095     it != _elements->end(); it++) {
00096         total += (*it).second->size();
00097     }
00098 }
00099
00100 void ElementManager::show() {
00101     _parentModel->tracer()->trace(Util::TraceLevel::toolDetailed,
00102     "Model Elements:");
00103     //std::map<std::string, List<ModelElement*>>* _elements;
00104     std::string key;
00105     List<ModelElement*>* list;
00106     Util::IncIndent();
00107     for (std::map<std::string, List<ModelElement*>>::iterator infraIt = _elements->
00108     begin(); infraIt != _elements->end(); infraIt++) {
00109         key = (*infraIt).first;
00110         list = (*infraIt).second;
00111         _parentModel->tracer()->trace(Util::TraceLevel::toolDetailed
00112     , key + ": (" + std::to_string(list->size()) + ")");
00113         Util::IncIndent();
00114         for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->
00115     list()->end(); it++) {
00116             _parentModel->tracer()->trace(
00117             Util::TraceLevel::toolDetailed, (*it)->show());
00118         }
00119     }
00120     Util::DecIndent();
00121 }
00122
00123 Model* ElementManager::parentModel() const {
00124     return _parentModel;
00125 }
00126
00127 bool ElementManager::hasChanged() const {
00128     return _hasChanged;
00129 }
00130
00131 void ElementManager::setHasChanged(bool _hasChanged) {
00132     this->_hasChanged = _hasChanged;
00133 }
00134
00135 List<ModelElement*>* ElementManager::elementList(std::string
00136     infraTypename) const {
00137     std::map<std::string, List<ModelElement*>>::iterator it = this->_elements->find(infraTypename);
00138     if (it == this->_elements->end()) {
00139         // list does not exists yet. Create it and set a valid iterator
00140         List<ModelElement*>* newList = new List<ModelElement*>();
00141         newList->setSortFunc([](const ModelElement* a, const
00142             ModelElement * b) {
00143                 return a->id() < b->id();
00144             });
00145         _elements->insert(std::pair<std::string, List<ModelElement*>>(infraTypename,
00146             newList));
00147         it = this->_elements->find(infraTypename);
00148     }
00149     List<ModelElement*>* infras = it->second;
00150     return infras;
00151 }
00152
00153 ModelElement* ElementManager::element(std::string infraTypename,
00154     Util::identification id) {
00155     List<ModelElement*>* list = elementList(infraTypename);
00156     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->
00157     list()->end(); it++) {
00158         if ((*it)->id() == id) { // found
00159             return (*it);
00160         }
00161     }
00162     return nullptr;
00163 }
00164
00165 int ElementManager::rankOf(std::string infraTypename, std::string name) {
00166     int rank = 0;
00167     List<ModelElement*>* list = elementList(infraTypename);
00168     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->
00169     list()->end(); it++) {
00170         if ((*it)->name() == name) { // found
00171             return rank;
00172         } else {
00173             rank++;
00174         }
00175     }
00176 }
```

```

00169      }
00170      return -1;
00171  }
00172
00173 std::list<std::string>* ElementManager::elementClassnames() const {
00174     std::list<std::string>* keys = new std::list<std::string>();
00175     for (std::map<std::string, List<ModelElement*>>::iterator it = _elements->begin();
00176         it != _elements->end(); it++) {
00177         keys->insert(keys->end(), (*it).first);
00178     }
00179     return keys;
00180 }
00181 ModelElement* ElementManager::element(std::string infraTypename,
00182                                         std::string name) {
00182     List<ModelElement*>* list = elementList(infraTypename);
00183     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->
00184         list()->end(); it++) {
00184         if ((*it)->name() == name) { // found
00185             return (*it);
00186         }
00187     }
00188     return nullptr;
00189 }
```

7.91 ElementManager.h File Reference

```
#include <list>
#include <map>
#include "List.h"
#include "ModelElement.h"
```

Classes

- class [ElementManager](#)

7.92 ElementManager.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   ElementManager.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 7 de Novembro de 2018, 12:48
00012 */
00013
00014 #ifndef ELEMENTMANAGER_H
00015 #define ELEMENTMANAGER_H
00016
00017 #include <list>
00018 #include <map>
00019 #include "List.h"
00020 #include "ModelElement.h"
00021
00022 class Model;
00023
00029 class ElementManager {
00030 public:
00031     ElementManager(Model* model);
00032     virtual ~ElementManager() = default;
00033 public:
00034     bool insert(ModelElement* infra);
00035     void remove(ModelElement* infra);
00036     bool insert(std::string infraTypename, ModelElement* infra);
00037     void remove(std::string infraTypename, ModelElement* infra);
00038     bool check(std::string infraTypename, ModelElement* infra, std::string expressionName,
```

```

        std::string* errorMessage);
00039     bool check(std::string infraTypename, std::string infraName, std::string expressionName, bool
mandatory, std::string* errorMessage);
00040     void clear();
00041 public:
00042     ModelElement* element(std::string infraTypename,
Util::identification id);
00043     ModelElement* element(std::string infraTypename, std::string name);
00044     unsigned int numberOfElements(std::string infraTypename);
00045     unsigned int numberOfElements();
00046     int rankOf(std::string infraTypename, std::string name);
00047     std::list<std::string>* elementClassnames() const;
00048
00049     //private:
00050 public:
00051     // \todo: MUST BE PRIVATE
00052     List<ModelElement*>* elementList(std::string infraTypename) const;
00053 public:
00054     void show();
00055     Model* parentModel() const;
00056     bool hasChanged() const;
00057     void setHasChanged(bool _hasChanged);
00058 private:
00059     std::map<std::string, List<ModelElement*>>* _elements;
00060     Model* _parentModel;
00061     bool _hasChanged = false;
00062 };
00063
00064 #endif /* ELEMENTMANAGER_H */
00065

```

7.93 ElementManager_if.h File Reference

Classes

- class [ElementManager_if](#)

7.94 ElementManager_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ElementManager_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Novembro de 2018, 01:06
00012 */
00013
00014 #ifndef ELEMENTMANAGER_IF_H
00015 #define ELEMENTMANAGER_IF_H
00016
00017 class ElementManager_if {
00018 public:
00019     ElementManager_if();
00020     ElementManager_if(const ElementManager_if& orig);
00021     virtual ~ElementManager_if();
00022 private:
00023
00024 };
00025
00026 #endif /* ELEMENTMANAGER_IF_H */
00027

```

7.95 Enter.cpp File Reference

```
#include "Enter.h"
#include "Model.h"
```

7.96 Enter.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Enter.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Enter.h"
00015 #include "Model.h"
00016
00017 Enter::Enter(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<Enter>(), name) {
00019
00020
00021     std::string Enter::show() {
00022         return ModelComponent::show() + ",station=" + this->_station-
00023             name();
00024
00025     ModelComponent* Enter::LoadInstance(Model* model,
00026         std::map<std::string, std::string>* fields) {
00027         Enter* newComponent = new Enter(model);
00028         try {
00029             newComponent->_loadInstance(fields);
00030         } catch (const std::exception& e) {
00031
00032             return newComponent;
00033         }
00034
00035     void Enter::setStation(Station* _station) {
00036         this->_station = _station;
00037     }
00038
00039     Station* Enter::getStation() const {
00040         return _station;
00041     }
00042
00043     void Enter::_execute(Entity* entity) {
00044         _station->enter(entity);
00045         _parentModel->sendEntityToComponent(entity, this->
00046             nextComponents()->frontConnection(), 0.0);
00047
00048     bool Enter::_loadInstance(std::map<std::string, std::string>* fields) {
00049         bool res = ModelComponent::_loadInstance(fields);
00050         if (res) {
00051             std::string stationName = ((*fields->find("stationName"))).second;
00052             Station* station = dynamic_cast<Station*>(_parentModel->
00053                 elements()->element(Util::TypeOf<Station>(),
00054                     stationName));
00055             this->_station = station;
00056         }
00057
00058     void Enter::_initBetweenReplications() {
00059
00060
00061     std::map<std::string, std::string>* Enter::_saveInstance() {
00062         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00063             ();
00064         fields->emplace("stationName", (this->_station->name()));
00065         return fields;
00066
00067     bool Enter::_check(std::string* errorMessage) {
00068         bool resultAll = true;
00069         resultAll &= _parentModel->elements()->check(Util::TypeOf<Station>(),
00070             _station, "Station", errorMessage);
00071         if (resultAll) {
00072             _station->setEnterIntoStationComponent(this); // this component will be
00073             // executed when an entity enters into the station
00074         }
00075
00076     PluginInformation* Enter::GetPluginInformation() {

```

```

00077     PluginInformation* info = new PluginInformation(Util::TypeOf<Enter>()
00078     , &Enter::LoadInstance);
00079     info->setReceiveTransfer(true);
00080     info->insertDynamicLibFileDependence("station.so");
00081 }
00082

```

7.97 Enter.h File Reference

```
#include "ModelComponent.h"
#include "Station.h"
```

Classes

- class [Enter](#)

7.98 Enter.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Enter.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef ENTER_H
00015 #define ENTER_H
00016
00017 #include "ModelComponent.h"
00018 #include "Station.h"
00019
00020 class Enter: public ModelComponent {
00021 public:
00022     Enter(Model* model, std::string name="");
00023     virtual ~Enter() = default;
00024 public:
00025     virtual std::string show();
00026 public:
00027     static PluginInformation* GetPluginInformation();
00028     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00029                                         std::string>* fields);
00030 public:
00031     void setStation(Station* _station);
00032     Station* getStation() const;
00033 protected:
00034     virtual void _execute(Entity* entity);
00035     virtual void _initBetweenReplications();
00036     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00037     virtual std::map<std::string, std::string>* _saveInstance();
00038     virtual bool _check(std::string* errorMessage);
00039 private: // association
00040     Station* _station;
00041 };
00042 #endif /* ENTER_H */
00043

```

7.99 Entity.cpp File Reference

```
#include <typeinfo>
#include "Entity.h"
#include "Attribute.h"
#include "Model.h"
```

7.100 Entity.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Entity.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 16:30
00012 */
00013
00014 #include <typeinfo>
00015 #include "Entity.h"
00016 #include "Attribute.h"
00017 #include "Model.h"
00018
00019 Entity::Entity(Model* model) : ModelElement(model,
00020     Util::TypeOf<Entity>()) {
00021     //elements = elements;
00022     _entityNumber = Util::GetLastIdOfType(Util::TypeOf<Entity>());
00023     unsigned int numAttributes = _parentModel->elements()->
00024         numberOfElements(Util::TypeOf<Attribute>());
00025     for (unsigned i = 0; i < numAttributes; i++) {
00026         std::map<std::string, double>* map = new std::map<std::string, double>();
00027         _attributeValues->insert(map);
00028     }
00029 //Entity::Entity(const Entity &orig): ModelElement(orig) {
00030 //this->_attributeValues = new List<std::map<std::string,double>>(orig._attributeValues);
00031 //this->_entityType = orig._entityType;
00032 //}
00033
00034 void Entity::setEntityTypeName(std::string
00035     entityTypename) throw () {
00036     EntityType* entitytype = dynamic_cast<EntityType*>(
00037         _parentModel->elements()->element(Util::TypeOf<EntityType>(),
00038         entityTypename));
00039     if (entitytype != nullptr) {
00040         this->_entityType = entitytype;
00041     } else {
00042         throw std::invalid_argument("EntityType does not exist");
00043     }
00044 }
00045
00046 std::string Entity::entityTypeName() const {
00047     return this->_entityType->name();
00048 }
00049
00050 void Entity::setEntityType(EntityType* entityType) {
00051     _entityType = entityType;
00052 }
00053
00054
00055 std::string Entity::show() {
00056     std::string message = ModelElement::show();
00057     if (this->_entityType != nullptr) {
00058         message += ",entityType=\"" + this->_entityType->name() + "\"";
00059     }
00060     message += ",attributes=[";
00061     _attributeValues->front();
00062     unsigned int i = 0;
00063     for (unsigned int i = 0; i < _attributeValues->size(); i++) {
00064         std::map<std::string, double>* map = _attributeValues->current();
00065         std::string attributeName = _parentModel->elements()->
00066             elementList(Util::TypeOf<Attribute>())->getAtRank(i)->name();
00067         message += attributeName + "=";
00068         if (map->size() == 0) { // scalar
00069             message += "NaN;" //std::to_string(map->begin()->second) + ";";
00070         } else if (map->size() == 1) { // scalar
00071             message += std::to_string(map->begin()->second) + ";";
00072         } else {
00073             // array or matrix
00074             message += "[";
00075             for (std::map<std::string, double>::iterator valIt = map->begin(); valIt != map->end(); valIt++) {
00076                 message += (*valIt).first + "=" + std::to_string((*valIt).second) + ",";
00077             }
00078         }
00079     }
00080 }

```

```

00079     _attributeValues->next();
00080 }
00081     message += "]";
00082     return message;
00083 }
00084
00085 double Entity::attributeValue(std::string attributeName) {
00086     return attributeValue("", attributeName);
00087 }
00088
00089 double Entity::attributeValue(std::string index, std::string attributeName) {
00090     int rank = _parentModel->elements()->rankOf(Util::TypeOf<Attribute>(),
00091         attributeName);
00092     if (rank >= 0) {
00093         std::map<std::string, double>* map = this->_attributeValues->getAtRank(rank);
00094         std::map<std::string, double>::iterator mapIt = map->find(index);
00095         if (mapIt != map->end()) { //found
00096             return (*mapIt).second;
00097         } else { // not found
00098             return 0.0;
00099         }
00100     _parentModel->tracer()->trace(
00101         Util::TraceLevel::errorRecover, "Attribute \"\" + attributeName + "\" not
00102         found");
00103     return 0.0; /* \todo: !! Never should happen. check how to report */
00104 }
00105
00106 double Entity::attributeValue(Util::identification attributeID) {
00107     return attributeValue("", attributeID);
00108 }
00109
00110 double Entity::attributeValue(std::string index,
00111     Util::identification attributeID) {
00112     ModelElement* element = _parentModel->elements()->
00113         element(Util::TypeOf<Attribute>(), attributeID);
00114     if (element != nullptr) {
00115         return attributeValue(index, element->name());
00116     }
00117     return 0.0; // attribute not found
00118 }
00119
00120 void Entity::setAttributeValue(std::string attributeName, double value) {
00121     setAttributeValue("", attributeName, value);
00122 }
00123
00124 void Entity::setAttributeValue(std::string index, std::string attributeName,
00125     double value) {
00126     int rank = _parentModel->elements()->rankOf(Util::TypeOf<Attribute>(),
00127         attributeName);
00128     if (rank >= 0) {
00129         std::map<std::string, double>* map = _attributeValues->getAtRank(rank);
00130         std::map<std::string, double>::iterator mapIt = map->find(index);
00131         if (mapIt != map->end()) { //found
00132             (*mapIt).second = value;
00133         } else { // not found
00134             map->insert({index, value}); // (map->end(), std::pair<std::string, double>(index, value));
00135         }
00136     } else
00137         _parentModel->tracer()->trace(
00138             Util::TraceLevel::errorRecover, "Attribute \"\" + attributeName + "\" not
00139             found");
00140 }
00141
00142 void Entity::setAttributeValue(Util::identification attributeID, double value) {
00143     setAttributeValue("", attributeID, value);
00144 }
00145
00146 void Entity::setAttributeValue(std::string index,
00147     Util::identification attributeID, double value) {
00148     std::string attrname = _parentModel->elements()->element(
00149         Util::TypeOf<Attribute>(), attributeID)->name();
00150     setAttributeValue(index, attrname, value);
00151 }
00152
00153 std::map<std::string, std::string>* Entity::_saveInstance() {

```

```

00154     return new std::map<std::string, std::string>();
00155 }
00156
00157 bool Entity::_check(std::string* errorMessage) {
00158     return true;
00159 }
```

7.101 Entity.h File Reference

```
#include <string>
#include <map>
#include "Util.h"
#include "List.h"
#include "ModelElement.h"
#include "EntityType.h"
```

Classes

- class Entity

7.102 Entity.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Entity.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 16:30
00012 */
00013
00014 #ifndef ENTITY_H
00015 #define ENTITY_H
00016
00017 #include <string>
00018 #include <map>
00019
00020 #include "Util.h"
00021 #include "List.h"
00022 #include "ModelElement.h"
00023 #include "EntityType.h"
00024
00025 class Entity : public ModelElement {
00026 public:
00027     Entity(Model* model);
00028     //Entity(const Entity &orig);
00029     virtual ~Entity() = default;
00030 public:
00031     virtual std::string show();
00032
00033 public: // g & s
00034     void setEntityType(std::string entityTypeName) throw(); // indirect
00035     access to EntityType
00036     std::string entityType() const;
00037     void setEntityType(EntityType* entityType); // direct access to
00038     EntityType
00039     EntityType* entityType() const;
00040 public:
00041     double attributeValue(std::string attributeName);
00042     double attributeValue(std::string index, std::string attributeName);
00043     double attributeValue(Util::identification attributeID);
00044     double attributeValue(std::string index, Util::identification
00045     attributeID);
00046     void setAttributeValue(std::string attributeName, double value);
00047     void setAttributeValue(std::string index, std::string attributeName, double value);
00048     void setAttributeValue(Util::identification attributeID, double
```

```

    value);
00096     void setAttributeValue(std::string index,
00097         Util::identification attributeID, double value);
00098 protected:
00099     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00100     virtual std::map<std::string, std::string>* _saveInstance();
00101     virtual bool _check(std::string* errorMessage);
00102 private:
00103     Util::identification _entityNumber;
00104     EntityType* _entityType = nullptr;
00105     List< std::map<std::string,double>* >* _attributeValues = new
00106     List<std::map<std::string,double>*>();
00107
00108 #endif /* ENTITY_H */
00109

```

7.103 EntityGroup.cpp File Reference

```

#include "EntityGroup.h"
#include "Model.h"
#include "Attribute.h"

```

7.104 EntityGroup.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Group.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 12 de Junho de 2019, 19:00
00012 */
00013
00014 #include "EntityGroup.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 EntityGroup::EntityGroup(Model* model, std::string name) :
00019     ModelElement(model, Util::TypeOf<EntityGroup>(), name) {
00020     _initCStats();
00021 }
00022 void EntityGroup::_initCStats() {
00023     _cstatNumberInGroup = new StatisticsCollector(
00024         _parentModel, "Number In Group", this);
00025     _childrenElements->insert({"NumberInGroup", _cstatNumberInGroup});
00026 }
00027 EntityGroup::~EntityGroup() {
00028     // _parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatNumberInGroup);
00029 }
00030
00031 std::string EntityGroup::show() {
00032     return ModelElement::show() +
00033         ",entities=" + this->_list->show();
00034 }
00035
00036 void EntityGroup::insertElement(Entity* element) {
00037     _list->insert(element);
00038     this->_cstatNumberInGroup->getStatistics()->getCollector()->
00039     addValue(_list->size());
00040 }
00041 void EntityGroup::removeElement(Entity* element) {
00042     //double tnow = this->_elements->getParentModel()->simulation()->getSimulatedTime();
00043     _list->remove(element);
00044     this->_cstatNumberInGroup->getStatistics()->getCollector()->
00045     addValue(_list->size());
00046 }

```

```

00046
00047 void EntityGroup::initBetweenReplications() {
00048     this->_list->clear();
00049 }
00050
00051 unsigned int EntityGroup::size() {
00052     return _list->size();
00053 }
00054
00055 Entity* EntityGroup::first() {
00056     return _list->front();
00057 }
00058
00059 //List<Waiting*>* Group::getList() const {
00060 //    return _list;
00061 //}
00062
00063 PluginInformation* EntityGroup::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(
00065         Util::TypeOf<EntityGroup>(), &EntityGroup::LoadInstance);
00066     return info;
00067 }
00068 ModelElement* EntityGroup::LoadInstance(
00069     Model* model, std::map<std::string, std::string>* fields) {
00070     EntityGroup* newElement = new EntityGroup(model);
00071     try {
00072         newElement->_loadInstance(fields);
00073     } catch (const std::exception& e) {
00074     }
00075     return newElement;
00076 }
00077
00078 bool EntityGroup::_loadInstance(std::map<std::string, std::string>* fields) {
00079     bool res = ModelElement::_loadInstance(fields);
00080     if (res) {
00081         try {
00082             } catch (...) {
00083         }
00084     }
00085     return res;
00086 }
00087
00088 std::map<std::string, std::string>* EntityGroup::_saveInstance() {
00089     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00090     //Util::TypeOf<Group>());
00091     return fields;
00092 }
00093 bool EntityGroup::_check(std::string* errorMessage) {
00094     std::string newNeededAttributeName = "Entity.Group";
00095     if (_parentModel->elements()->element(Util::TypeOf<Attribute>(),
00096         newNeededAttributeName) == nullptr) {
00097         Attribute* attr1 = new Attribute(_parentModel, newNeededAttributeName);
00098         //_parentModel->insert(attr1);
00099     }
00100 }
00101

```

7.105 EntityGroup.h File Reference

```

#include "ModelElement.h"
#include "Entity.h"
#include <map>
#include <list>

```

Classes

- class EntityGroup

7.106 EntityGroup.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Group.h
00009 * Author: rlcancian
00010 *
00011 * Created on 12 de Junho de 2019, 19:00
00012 */
00013
00014 #ifndef ENTITYGROUP_H
00015 #define ENTITYGROUP_H
00016
00017 #include "ModelElement.h"
00018 #include "Entity.h"
00019 #include <map>
00020 #include <list>
00021
00022 /*
00023 The EntityGroup represents a set of entities grouped together somehow. The entities in this group are
kept into a internal queue. Usually one entity representing all grouped entities is generated by
ModelComponents and the EntityGroup it represented is accessed through an attribute.
00024 */
00025 class EntityGroup : public ModelElement {
00026 public:
00027     EntityGroup(Model* model, std::string name = "");
00028     virtual ~EntityGroup();
00029 public:
00030     virtual std::string show();
00031 public:
00032     static PluginInformation* GetPluginInformation();
00033     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00034                                     std::string*>* fields);
00034 public:
00035     void insertElement(Entity* element);
00036     void removeElement(Entity* element);
00037     unsigned int size();
00038     Entity* first();
00039 public:
00040     void initBetweenReplications();
00041 protected:
00042     virtual bool _loadInstance(std::map<std::string, std::string*>* fields);
00043     virtual std::map<std::string, std::string*>* _saveInstance();
00044     virtual bool _check(std::string* errorMessage);
00045 private:
00046     void _initCStats();
00047 private: //1::n
00048     List<Entity*>* _list = new List<Entity*>();
00049 private: // child inner element
00050     StatisticsCollector* _cstatNumberInGroup;
00051 };
00052
00053 #endif /* ENTITYGROUP_H */
00054

```

7.107 EntityType.cpp File Reference

```

#include "EntityType.h"
#include "Model.h"
#include "SimulationResponse.h"

```

7.108 EntityType.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006

```

```

00007 /*
00008  * File:  EntityType.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 19:24
00012 */
00013
00014 #include "EntityType.h"
00015 #include "Model.h"
00016 #include "SimulationResponse.h"
00017
00018 //EntityType::EntityType(Model* model) : ModelElement(model, Util::TypeOf<EntityType>()) {
00019 //    _elemManager = elemManager;
00020 //    _initCostsAndStatistics();
00021 //}
00022
00023 EntityType::EntityType(Model* model, std::string name) :
00024     ModelElement(model, Util::TypeOf<EntityType>(), name) {
00025     _initCostsAndStatistics();
00026
00027 void EntityType::_initCostsAndStatistics() {
00028     _initialWaitingCost = 0.0;
00029     _initialVACost = 0.0;
00030     _initialNVACost = 0.0;
00031     _initialOtherCost = 0.0;
00032     // add cstats as elements
00033     //_cstatWaitingTime = new StatisticsCollector("Waiting Time", this);
00034     //_cstatTransferTime = new StatisticsCollector("Transfer Time", this);
00035     //_cstatOtherTime = new StatisticsCollector("Other Time", this);
00036     //_cstatVATime = new StatisticsCollector("Value Added Time", this);
00037     //_cstatNVATime = new StatisticsCollector("Non Value Added Time", this);
00038     //_cstatTotalTime = new StatisticsCollector("Time In System", this);
00039     //_elemManager->insert(_cstatNVATime);
00040     //_elemManager->insert(_cstatOtherTime);
00041     //_elemManager->insert(_cstatTotalTime);
00042     //_elemManager->insert(_cstatTransferTime);
00043     //_elemManager->insert(_cstatVATime);
00044     //_elemManager->insert(_cstatWaitingTime);
00045     // insert cstats as simulation responses
00046
00047 //    List<::SimulationResponse*>* responses = this->_elemManager->getParentModel()->getResponses();
00048 //    responses->insert(new SimulationResponse(Util::TypeOf<EntityType>(), "Waiting time",
00049 //        DefineGetterMember<StatisticsDefaultImpl1>(this->_cstatWaitingTime,
00050 //        &StatisticsDefaultImpl1::average)));
00051 EntityType::~EntityType() {
00052     //_elemManager->remove(Util::TypeOf<StatisticsCollector>(), _cstatNVATime);
00053     //_elemManager->remove(Util::TypeOf<StatisticsCollector>(), _cstatOtherTime);
00054     //_elemManager->remove(Util::TypeOf<StatisticsCollector>(), _cstatTotalTime);
00055     //_elemManager->remove(Util::TypeOf<StatisticsCollector>(), _cstatTransferTime);
00056     //_elemManager->remove(Util::TypeOf<StatisticsCollector>(), _cstatVATime);
00057     //_elemManager->remove(Util::TypeOf<StatisticsCollector>(), _cstatWaitingTime);
00058     // remove all CStats
00059     for (std::list<StatisticsCollector*>::iterator it = this->_statisticsCollectors->
00060         list()->begin(); it != this->_statisticsCollectors->list()->end(); it++) {
00061         _parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), (*it));
00062     }
00063
00064 std::string EntityType::show() {
00065     return ModelElement::show() +
00066         ",initialPicture=" + this->_initialPicture; // add more...
00067 }
00068
00069 void EntityType::setInitialWaitingCost(double _initialWaitingCost) {
00070     this->_initialWaitingCost = _initialWaitingCost;
00071 }
00072
00073 double EntityType::initialWaitingCost() const {
00074     return _initialWaitingCost;
00075 }
00076
00077 void EntityType::setInitialOtherCost(double _initialOtherCost) {
00078     this->_initialOtherCost = _initialOtherCost;
00079 }
00080
00081 double EntityType::initialOtherCost() const {
00082     return _initialOtherCost;
00083 }
00084
00085 void EntityType::setInitialNVACost(double _initialNVACost) {
00086     this->_initialNVACost = _initialNVACost;
00087 }
00088
00089 double EntityType::initialNVACost() const {

```

```

00090     return _initialNVACost;
00091 }
00092
00093 void EntityType::setInitialVACost(double _initialVACost) {
00094     this->_initialVACost = _initialVACost;
00095 }
00096
00097 double EntityType::initialVACost() const {
00098     return _initialVACost;
00099 }
00100
00101 void EntityType::setInitialPicture(std::string _initialPicture) {
00102     this->_initialPicture = _initialPicture;
00103 }
00104
00105 std::string EntityType::initialPicture() const {
00106     return _initialPicture;
00107 }
00108
00109 //StatisticsCollector* EntityType::getCstatTotalTime() const {
00110 //    return _cstatTotalTime;
00111 //}
00112
00113 //StatisticsCollector* EntityType::getCstatNVATime() const {
00114 //    return _cstatNVATime;
00115 //}
00116
00117 //StatisticsCollector* EntityType::getCstatVATime() const {
00118 //    return _cstatVATime;
00119 //}
00120
00121 //StatisticsCollector* EntityType::getCstatOtherTime() const {
00122 //    return _cstatOtherTime;
00123 //}
00124
00125 //StatisticsCollector* EntityType::getCstatTransferTime() const {
00126 //    return _cstatTransferTime;
00127 //}
00128
00129 //StatisticsCollector* EntityType::getCstatWaitingTime() const {
00130 //    return _cstatWaitingTime;
00131 //}
00132
00133 StatisticsCollector* EntityType::statisticsCollector(
00134     std::string name) {
00135     StatisticsCollector* cstat;
00136     for (std::list<StatisticsCollector*>::iterator it = _statisticsCollectors->
00137         list()->begin(); it!= _statisticsCollectors->list()->end(); it++) {
00138         cstat = (*it);
00139         if (cstat->name()==name) {
00140             return cstat;
00141         }
00142         // not found. Create it, insert it into the list of cstats, into the model element manager, and then
00143         return it;
00144         cstat = new StatisticsCollector(_parentModel, name, this);
00145         _statisticsCollectors->insert(cstat);
00146         //_parentModel->insert(cstat); // unnecessary
00147         return cstat;
00148     }
00149
00150     PluginInformation* EntityType::GetPluginInformation() {
00151         PluginInformation* info = new PluginInformation(
00152             Util::TypeOf<EntityType>(), &EntityType::LoadInstance); return info;
00153     }
00154
00155     ModelElement* EntityType::LoadInstance(Model* model,
00156         std::map<std::string, std::string>* fields) {
00157         EntityType* newElement = new EntityType(model);
00158         try {
00159             newElement->_loadInstance(fields);
00160         } catch (const std::exception& e) {
00161             return newElement;
00162         }
00163     }
00164     bool EntityType::_loadInstance(std::map<std::string, std::string>* fields) {
00165         bool res = ModelElement::_loadInstance(fields);
00166         if (res) {
00167             this->_initialNVACost = std::stod((*(fields->find("initialNVACost"))).second);
00168             this->_initialOtherCost = std::stod((*(fields->find("initialOtherCost"))).second);
00169             this->_initialPicture = ((*(fields->find("initialPicture"))).second);
00170             this->_initialVACost = std::stod((*(fields->find("initialVACost"))).second);
00171             this->_initialWaitingCost = std::stod((*(fields->find("initialWaitingCost"))).second);
00172         }
00173         return res;

```

```

00172 }
00173
00174 std::map<std::string, std::string>* EntityType::_saveInstance() {
00175     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00176     //Util::TypeOf<EntityType>());
00177     fields->emplace("initialNVACost", std::to_string(this->_initialNVACost));
00178     fields->emplace("initialOtherCost", std::to_string(this->_initialOtherCost));
00179     fields->emplace("initialPicture", this->_initialPicture);
00180     fields->emplace("initialVACost", std::to_string(this->_initialVACost));
00181     fields->emplace("initialWaitingCost", std::to_string(this->_initialWaitingCost));
00182     return fields;
00183 }
00184 bool EntityType::_check(std::string* errorMessage) {
00185     return true;
00186 }
00187
00188

```

7.109 EntityType.h File Reference

```

#include <string>
#include "ModelElement.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
#include "Plugin.h"

```

Classes

- class **EntityType**

7.110 EntityType.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    EntityType.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 19:24
00012 */
00013
00014 #ifndef ENTITYTYPE_H
00015 #define ENTITYTYPE_H
00016
00017 #include <string>
00018 #include "ModelElement.h"
00019 #include "StatisticsCollector.h"
00020 #include "ElementManager.h"
00021 #include "Plugin.h"
00022
00023 //##include "Model.h"
00024
00025 class EntityType : public ModelElement {
00026 public:
00027     //EntityType(Model* model);
00028     EntityType(Model* model, std::string name(""));
00029     virtual ~EntityType();
00030 public:
00031     virtual std::string show();
00032 public:
00033     static PluginInformation* GetPluginInformation();
00034     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00035                                         std::string>* fields);
00035 public: //get & set
00036     void setInitialWaitingCost(double _initialWaitingCost);
00037     double initialWaitingCost() const;

```

```

00038     void setInitialOtherCost(double _initialOtherCost);
00039     double initialOtherCost() const;
00040     void setInitialNVACost(double _initialNVACost);
00041     double initialNVACost() const;
00042     void setInitialVACost(double _initialVACost);
00043     double initialVACost() const;
00044     void setInitialPicture(std::string _initialPicture);
00045     std::string initialPicture() const;
00046 public: //get
00047     //StatisticsCollector* getCstatTotalTime() const;
00048     //StatisticsCollector* getCstatNVATime() const;
00049     //StatisticsCollector* getCstatVATime() const;
00050     //StatisticsCollector* getCstatOtherTime() const;
00051     //StatisticsCollector* getCstatTransferTime() const;
00052     //StatisticsCollector* getCstatWaitingTime() const;
00053     StatisticsCollector* statisticsCollector(std::string
00054         name);
00055 protected: // must be overridden by derived classes
00056     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00057     virtual std::map<std::string, std::string>* _saveInstance();
00058     virtual bool _check(std::string* errorMessage);
00059
00060 private:
00061     void _initCostsAndStatistics();
00062 private:
00063     std::string _initialPicture = "report";
00064     double _initialVACost = 0.0;
00065     double _initialNVACost = 0.0;
00066     double _initialOtherCost = 0.0;
00067     double _initialWaitingCost = 0.0;
00068 //private:
00069 //ElementManager* _elemManager;
00070 private: //1:n
00071     List<StatisticsCollector*>* _statisticsCollectors = new
00072     List<StatisticsCollector*>();
00073     //StatisticsCollector* _cstatWaitingTime; // = new StatisticsCollector("Waiting Time");
00074     //StatisticsCollector* _cstatTransferTime; // = new StatisticsCollector("Transfer Time");
00075     //StatisticsCollector* _cstatOtherTime; // = new StatisticsCollector("Other Time");
00076     //StatisticsCollector* _cstatNVATime; // = new StatisticsCollector("Value Added Time");
00077     //StatisticsCollector* _cstatVATime; // = new StatisticsCollector("Non Value Added Time");
00078 };
00079
00080 #endif /* ENTITYTYPE_H */
00081

```

7.111 Event.cpp File Reference

```
#include "Event.h"
```

7.112 Event.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Event.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:41
00012 */
00013
00014 #include "Event.h"
00015
00016 Event::Event(double time, Entity* entity, ModelComponent* component,
    unsigned int componentInputNumber) {
00017     _time = time;
00018     _entity = entity;
00019     _component = component;
00020     _componentInputNumber = componentInputNumber;
00021 }
00022

```

```

00023 Event::Event(double time, Entity* entity, Connection* connection) {
00024     _time = time;
00025     _entity = entity;
00026     _component = connection->first;
00027     _componentInputNumber = connection->second;
00028 }
00029
00030
00031 std::string Event::show() {
00032     return "time=" + std::to_string(_time) +
00033         ",entity=" + std::to_string(_entity->entityNumber()) +
00034         ",comp=\"" + _component->name() + "\""; //+std::to_string(_component->getId())+"";
00035 }
00036
00037 unsigned int Event::componentInputNumber() const {
00038     return _componentInputNumber;
00039 }
00040
00041 double Event::time() const {
00042     return _time;
00043 }
00044
00045 ModelComponent* Event::component() const {
00046     return _component;
00047 }
00048
00049 Entity* Event::entity() const {
00050     return _entity;
00051 }
00052

```

7.113 Event.h File Reference

```

#include <string>
#include "ModelElement.h"
#include "Entity.h"
#include "ModelComponent.h"

```

Classes

- class **Event**

7.114 Event.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Event.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 19:41
00012 */
00013
00014 #ifndef EVENT_H
00015 #define EVENT_H
00016
00017 #include <string>
00018
00019 #include "ModelElement.h"
00020 #include "Entity.h"
00021 #include "ModelComponent.h"
00022
00023 class Event { //: public ModelElement {
00024     public:
00025         Event(double time, Entity* entity, ModelComponent*
00026             component, unsigned int componentInputNumber = 0);
00027         Event(double time, Entity* entity, Connection* connection);

```

```

00030     virtual ~Event() = default;
00031 public:
00032     double time() const;
00033     ModelComponent* component() const;
00034     Entity* entity() const;
00035     unsigned int componentInputNumber() const;
00036 public:
00037     std::string show();
00038 private:
00039     double _time;
00040     Entity* _entity;
00041     ModelComponent* _component;
00042     unsigned int _componentInputNumber;
00043 };
00044
00045 #endif /* EVENT_H */
00046

```

7.115 Exit.cpp File Reference

```
#include "Exit.h"
#include "Model.h"
```

7.116 Exit.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:   Exit.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #include "Exit.h"
00015
00016 #include "Model.h"
00017
00018 Exit::Exit(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Exit>(), name) {
00020
00021
00022 std::string Exit::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Exit::LoadInstance(Model* model, std::map<std::string,
00027     std::string>* fields) {
00028     Exit* newComponent = new Exit(model);
00029     try {
00030         newComponent->_loadInstance(fields);
00031     } catch (const std::exception& e) {
00032     }
00033     return newComponent;
00034 }
00035
00036 void Exit::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038     entity forward");
00039     this->_parentModel->sendEntityToComponent(entity, this->
00040         nextComponents()->frontConnection(), 0.0);
00041 }
00042
00043 bool Exit::_loadInstance(std::map<std::string, std::string>* fields) {
00044     bool res = ModelComponent::_loadInstance(fields);
00045     if (res) {
00046         //...
00047     }
00048     return res;
00049 }
```

```

00048
00049 void Exit::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* Exit::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00054     ();
00055     //...
00056     return fields;
00057 }
00058 bool Exit::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Exit::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Exit>(),
00066     &Exit::LoadInstance);
00067     // ...
00068     return info;
00069 }
00070

```

7.117 Exit.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Exit](#)

7.118 Exit.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Exit.h
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #ifndef EXIT_H
00015 #define EXIT_H
00016
00017 #include "ModelComponent.h"
00018
00037 class Exit : public ModelComponent {
00038 public: // constructors
00039     Exit(Model* model, std::string name(""));
00040     virtual ~Exit() = default;
00041 public: // virtual
00042     virtual std::string show();
00043 public: // static
00044     static PluginInformation* GetPluginInformation();
00045     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00046     std::string>* fields);
00046 protected: // virtual
00047     virtual void _execute(Entity* entity);
00048     virtual void _initBetweenReplications();
00049     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00050     virtual std::map<std::string, std::string>* _saveInstance();
00051     virtual bool _check(std::string* errorMessage);
00052 private: // methods
00053 private: // attributes 1:1
00054 private: // attributes 1:n
00055 };
00056
00057
00058 #endif /* EXIT_H */
00059

```

7.119 ExperimentDesign_if.h File Reference

```
#include "FactorOrInteractionContribution.h"
#include "ProcessAnalyser_if.h"
```

Classes

- class [ExperimentDesign_if](#)

7.120 ExperimentDesign_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ExperimentDesign.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 2 de Outubro de 2018, 22:47
00012 */
00013
00014 #ifndef EXPERIMENTDESIGN_IF_H
00015 #define EXPERIMENTDESIGN_IF_H
00016
00017 #include "FactorOrInteractionContribution.h"
00018 #include "ProcessAnalyser_if.h"
00019
00020 class ExperimentDesign_if {
00021 public:
00022     virtual ProcessAnalyser_if* getProcessAnalyser() const = 0;
00023     virtual bool generate2krScenarioExperiments() = 0;
00024     virtual bool calculateContributionAndCoefficients() = 0;
00025     virtual std::list<FactorOrInteractionContribution*>* getContributions() const = 0;
00026 };
00027
00028 #endif /* EXPERIMENTDESIGN_IF_H */
00029
00030
00031
00032
```

7.121 ExperimentDesignDefaultImpl1.cpp File Reference

```
#include "ExperimentDesignDefaultImpl1.h"
```

7.122 ExperimentDesignDefaultImpl1.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ExperimentDesignDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 10 de Agosto de 2018, 10:19
00012 */
00013
00014 #include "ExperimentDesignDefaultImpl1.h"
00015
00016 ExperimentDesignDefaultImpl1::ExperimentDesignDefaultImpl1
() {
```

```

00017 }
00018
00019
00020 std::list<FactorOrInteractionContribution*>*
    ExperimentDesignDefaultImpl1::getContributions() const {
00021     return _contributions;
00022 }
00023
00024 bool ExperimentDesignDefaultImpl1::generate2krScenarioExperiments
    () {
00025     return true;
00026 }
00027
00028 bool ExperimentDesignDefaultImpl1::calculateContributionAndCoefficients
    () {
00029     return true;
00030 }
00031
00032 ProcessAnalyser_if*
    ExperimentDesignDefaultImpl1::getProcessAnalyser() const {
00033     return _processAnalyser;
00034 }

```

7.123 ExperimentDesignDefaultImpl1.h File Reference

```
#include "ExperimentDesign_if.h"
```

Classes

- class [ExperimentDesignDefaultImpl1](#)

7.124 ExperimentDesignDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   ExperimentDesignDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 10 de Agosto de 2018, 10:19
00012 */
00013
00014 #ifndef EXPERIMENTDESIGNDEFAULTIMPL1_H
00015 #define EXPERIMENTDESIGNDEFAULTIMPL1_H
00016
00017 #include "ExperimentDesign_if.h"
00018
00019 class ExperimentDesignDefaultImpl1 : public
00020     ExperimentDesign_if {
00021     public:
00022         ExperimentDesignDefaultImpl1();
00023         virtual ~ExperimentDesignDefaultImpl1() = default;
00024     public:
00025         virtual ProcessAnalyser_if* getProcessAnalyser() const;
00026         virtual bool generate2krScenarioExperiments();
00027         virtual bool calculateContributionAndCoefficients();
00028         virtual std::list<FactorOrInteractionContribution*>* getContributions() const;
00029     private:
00030         ProcessAnalyser_if* _processAnalyser; //= new
00031         Traits<ExperimentDesign_if>::ProcessAnalyserImplementation();
00032         std::list<FactorOrInteractionContribution*>* _contributions = new
00033             std::list<FactorOrInteractionContribution*>();
00034     };
00035 #endif /* EXPERIMENTDESIGNDEFAULTIMPL1_H */
00036

```

7.125 ExperimentDesignDummyImpl.cpp File Reference

```
#include "ExperimentDesignDummyImpl.h"
#include "Assign.h"
```

7.126 ExperimentDesignDummyImpl.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   ExperimentDesignDummyImpl.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:32
00012 */
00013
00014 #include "ExperimentDesignDummyImpl.h"
00015 #include "Assign.h"
00016
00017 ExperimentDesignDummyImpl::ExperimentDesignDummyImpl()
00018 {
00019
00020     std::list<FactorOrInteractionContribution*>*
00021         ExperimentDesignDummyImpl::getContributions() const {
00022             return _contributions;
00023
00024
00025     bool ExperimentDesignDummyImpl::generate2krScenarioExperiments
00026         () {
00027             return true;
00028
00029     bool ExperimentDesignDummyImpl::calculateContributionAndCoefficients
00030         () {
00031             return true;
00032
00033     ProcessAnalyser_if*
00034         ExperimentDesignDummyImpl::getProcessAnalyser() const {
00035             return _processAnalyser;
00036 }
```

7.127 ExperimentDesignDummyImpl.h File Reference

```
#include "ExperimentDesign_if.h"
```

Classes

- class [ExperimentDesignDummyImpl](#)

7.128 ExperimentDesignDummyImpl.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ExperimentDesignDummyImpl.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:32
00012 */
00013
00014 #ifndef EXPERIMENTDESIGNDUMMYIMPL_H
00015 #define EXPERIMENTDESIGNDUMMYIMPL_H
00016
00017 #include "ExperimentDesign_if.h"
00018
00022 class ExperimentDesignDummyImpl : public
00023     ExperimentDesign_if {
00024     public:
00024         ExperimentDesignDummyImpl();
00025         virtual ~ExperimentDesignDummyImpl() = default;
00026     public:
00027         virtual ProcessAnalyser_if* getProcessAnalyser() const;
00028     public:
00029         virtual bool generate2krScenarioExperiments();
00030         virtual bool calculateContributionAndCoefficients();
00031         virtual std::list<FactorOrInteractionContribution*>* getContributions() const;
00032     private:
00033         ProcessAnalyser_if* _processAnalyser; //= new
00034             Traits<ExperimentDesign_if>::ProcessAnalyserImplementation();
00034         std::list<FactorOrInteractionContribution*>* _contributions = new
00035             std::list<FactorOrInteractionContribution*>();
00035 };
00036 #endif /* EXPERIMENTDESIGNDUMMYIMPL_H */

```

7.129 FactorOrInteractionContribution.cpp File Reference

```
#include "FactorOrInteractionContribution.h"
```

7.130 FactorOrInteractionContribution.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: FactorOrInteractionContribution.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 20:08
00012 */
00013
00014 #include "FactorOrInteractionContribution.h"
00015
00016 FactorOrInteractionContribution::FactorOrInteractionContribution
00017     (double contribution, double modelCoefficient, std::list<SimulationControl*>* controls) {
00017         _contribution = contribution;
00018         _modelCoefficient = modelCoefficient;
00019         _controls = controls;
00020     }
00021
00022
00023 double FactorOrInteractionContribution::getModelCoefficient
00024     () const {
00024         return _modelCoefficient;
00025     }
00026
00027 std::list<SimulationControl*>* FactorOrInteractionContribution::getControls
00028     () const {

```

```

00028     return _controls;
00029 }
00030
00031 double FactorOrInteractionContribution::getContribution()
00032     const {
00033         return _contribution;
00034     }

```

7.131 FactorOrInteractionContribution.h File Reference

```
#include <list>
#include "SimulationControl.h"
```

Classes

- class [FactorOrInteractionContribution](#)

7.132 FactorOrInteractionContribution.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: FactorOrInteractionContribution.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 10 de Outubro de 2018, 20:08
00012 */
00013
00014 #ifndef FACTORORINTERACTIONCONTRIBUTION_H
00015 #define FACTORORINTERACTIONCONTRIBUTION_H
00016
00017 #include <list>
00018 #include "SimulationControl.h"
00019
00023 class FactorOrInteractionContribution {
00024 public:
00025     FactorOrInteractionContribution(double contribution, double
00026 modelCoefficient, std::list<SimulationControl*>* controls);
00026     ~FactorOrInteractionContribution();
00027 public:
00028     double getModelCoefficient() const;
00029     std::list<SimulationControl*>* getControls() const;
00030     double getContribution() const;
00031 private:
00032     double _contribution;
00033     double _modelCoefficient;
00034     std::list<SimulationControl*>* _controls;
00035 };
00036
00037 #endif /* FACTORORINTERACTIONCONTRIBUTION_H */
00038

```

7.133 Failure.cpp File Reference

```
#include "Failure.h"
```

7.134 Failure.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: Failure.cpp
00008 * Author: rlcancian
00009 *
00010 *
00011 * Created on 20 de Failureembro de 2019, 20:07
00012 */
00013
00014 #include "Failure.h"
00015
00016 Failure::Failure(Model* model, std::string name) :
00017     ModelElement(model, Util::TypeOf<Failure>(), name) {
00018
00019     std::string Failure::show() {
00020         return ModelElement::show() +
00021             "";
00022     }
00023
00024     PluginInformation* Failure::GetPluginInformation() {
00025         PluginInformation* info = new PluginInformation(Util::TypeOf<Failure>
00026             (), &Failure::LoadInstance);
00027         return info;
00028     }
00029     ModelElement* Failure::LoadInstance(Model* model,
00030         std::map<std::string, std::string>* fields) {
00031         Failure* newElement = new Failure(model);
00032         try {
00033             newElement->_loadInstance(fields);
00034         } catch (const std::exception& e) {
00035         }
00036         return newElement;
00037     }
00038
00039     bool Failure::_loadInstance(std::map<std::string, std::string>* fields) {
00040         bool res = ModelElement::_loadInstance(fields);
00041         if (res) {
00042             try {
00043                 //this->_attributeName = (*fields->find("attributeName")).second;
00044                 //this->_orderRule = static_cast<OrderRule> (std::stoi((*fields->find("orderRule")).second));
00045             } catch (...) {
00046             }
00047         }
00048         return res;
00049     }
00050
00051     std::map<std::string, std::string>* Failure::_saveInstance() {
00052         std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00053         //Util::TypeOf<Failure>());
00054         //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00055         //fields->emplace("attributeName", this->_attributeName);
00056         return fields;
00057     }
00058     bool Failure::_check(std::string* errorMessage) {
00059         bool resultAll = true;
00060         // resultAll |= ...
00061         return resultAll;
00062     }
00063
00064     ParserChangesInformation*
00065         Failure::_getParserChangesInformation() {
00066         ParserChangesInformation* changes = new
00067             ParserChangesInformation();
00068         //changes->getProductionToAdd()->insert(...);
00069         //changes->getTokensToAdd()->insert(...);
00070         return changes;
00071     }

```

7.135 Failure.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
```

```
#include "ParserChangesInformation.h"
#include "PluginInformation.h"
```

Classes

- class [Failure](#)

7.136 Failure.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Failure.h
00009  * Author: rlcancian
00010 *
00011  * Created on 20 de Failureembro de 2019, 20:07
00012 */
00013
00014 #ifndef FAILURE_H
00015 #define FAILURE_H
00016
00017
00018 #include "ModelElement.h"
00019 #include "ElementManager.h"
00020 #include "ParserChangesInformation.h"
00021 #include "PluginInformation.h"
00022
00023 class Failure: public ModelElement {
00024 public:
00025     Failure(Model* model, std::string name="");
00026     virtual ~Failure() = default;
00027 public: // static
00028     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00029                                         std::string>* fields);
00030     static PluginInformation* GetPluginInformation();
00031 public:
00032     virtual std::string show();
00033
00034 protected: // must be overriden by derived classes
00035     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00036     virtual std::map<std::string, std::string>* _saveInstance();
00037 protected: // could be overriden by derived classes
00038     virtual bool _check(std::string* errorMessage);
00039     virtual ParserChangesInformation*
00040     _getParserChangesInformation();
00041
00042 };
00043
00044 #endif /* FAILURE_H */
```

7.137 File.cpp File Reference

```
#include "File.h"
```

7.138 File.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: File.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Fileembro de 2019, 20:07
00012 */
00013
00014 #include "File.h"
00015
00016
00017 File::File(Model* model, std::string name) : ModelElement(model,
00018     Util::TypeOf<File>(), name) {
00019     //_elems = elems;
00020 }
00021 std::string File::show() {
00022     return ModelElement::show() +
00023         "";
00024 }
00025
00026 PluginInformation* File::GetPluginInformation() {
00027     PluginInformation* info = new PluginInformation(Util::TypeOf<File>(),
00028         &File::LoadInstance);
00029     return info;
00030 }
00031 ModelElement* File::LoadInstance(Model* model, std::map<std::string,
00032     std::string>* fields) {
00033     File* newElement = new File(model);
00034     try {
00035         newElement->_loadInstance(fields);
00036     } catch (const std::exception& e) {
00037     }
00038     return newElement;
00039 }
00040
00041 bool File::_loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelElement::_loadInstance(fields);
00043     if (res) {
00044         try {
00045             //this->_attributeName = (*fields->find("attributeName")).second;
00046             //this->_orderRule = static_cast<OrderRule> (std::stoi((*fields->find("orderRule")).second));
00047         } catch (...) {
00048         }
00049     }
00050     return res;
00051 }
00052
00053 std::map<std::string, std::string>* File::_saveInstance() {
00054     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00055     //Util::TypeOf<File>());
00056     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00057     //fields->emplace("attributeName", this->_attributeName);
00058     return fields;
00059 }
00060 bool File::_check(std::string* errorMessage) {
00061     bool resultAll = true;
00062     // resultAll |= ...
00063     return resultAll;
00064 }
00065
00066 ParserChangesInformation*
00067     File::_getParserChangesInformation() {
00068     ParserChangesInformation* changes = new
00069         ParserChangesInformation();
00070     //changes->getProductionToAdd()->insert(...);
00071     //changes->getTokensToAdd()->insert(...);
00072     return changes;
00073 }
```

7.139 File.h File Reference

```
#include "ModelElement.h"
```

```
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"
```

Classes

- class [File](#)

7.140 File.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: File.h
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Fileembro de 2019, 20:07
00012 */
00013
00014 #ifndef FILE_H
00015 #define FILE_H
00016
00017
00018 #include "ModelElement.h"
00019 #include "ElementManager.h"
00020 #include "ParserChangesInformation.h"
00021 #include "PluginInformation.h"
00022
00064 class File: public ModelElement {
00065 public:
00066     File(Model* model, std::string name = "");
00067     virtual ~File() = default;
00068 public: // static
00069     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00070                                         std::string>* fields);
00070     static PluginInformation* GetPluginInformation();
00071 public:
00072     virtual std::string show();
00073
00074 protected: // must be overridden by derived classes
00075     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00076     virtual std::map<std::string, std::string>* _saveInstance();
00077 protected: // could be overridden by derived classes
00078     virtual bool _check(std::string* errorMessage);
00079     virtual ParserChangesInformation*
00080         _getParserChangesInformation();
00081 };
00082
00083 #endif /* FILE_H */
00084
```

7.141 FirstExampleOfSimulation.cpp File Reference

```
#include "FirstExampleOfSimulation.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "EntityType.h"
```

7.142 FirstExampleOfSimulation.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FirstExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 18:34
00012 */
00013
00014 #include "FirstExampleOfSimulation.h"
00015
00016 // you have to included need libs
00017
00018 // GEnSys Simulator
00019 #include "Simulator.h"
00020
00021 // Model Components
00022 #include "Create.h"
00023 #include "Delay.h"
00024 #include "Dispose.h"
00025
00026 // Model model
00027 #include "EntityType.h"
00028
00029 FirstExampleOfSimulation::FirstExampleOfSimulation() {
00030 }
00031
00032 int FirstExampleOfSimulation::main(int argc, char** argv) {
00033     Simulator* simulator = new Simulator();
00034     // Handle traces and simulation events to output them
00035     this->setDefaultTraceHandlers(simulator->tracer());
00036     simulator->tracer()->setTraceLevel(
00037         Util::TraceLevel::componentDetailed);
00038     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00039     this->insertFakePluginsByHand(simulator);
00040     bool wantToCreateNewModelAndSaveInsteadOfJustLoad = false;//true;
00041     Model* model;
00042     if (wantToCreateNewModelAndSaveInsteadOfJustLoad) {
00043         // creates an empty model
00044         model = new Model(simulator);
00045         //
00046         // build the simulation model
00047         // if no ModelInfo is provided, then the model will be simulated once (one replication) and the
00048         // replication length will be 60 seconds (simulated time)
00049         // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00050         EntityType* entityType1 = new EntityType(model, "Type_of_Representative_Entity");
00051         // create a ModelComponent of type Create, used to insert entities into the model
00052         Create* createl = new Create(model);
00053         createl->setEntityType(entityType1);
00054         createl-> setTimeBetweenCreationsExpression("1.5"); // create one new
00055         entity every 1.5 seconds
00056         // create a ModelComponent of type Delay, used to represent a time delay
00057         Delay* delay1 = new Delay(model);
00058         // if nothing else is set, the delay will take 1 second
00059         // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00060         Dispose* dispose1 = new Dispose(model); // insert the component into the model
00061         // connect model components to create a "workflow" -- should always start from a SourceModelComponent
00062         // and end at a SinkModelComponent (it will be checked)
00063         createl->nextComponents()->insert(delay1);
00064         delay1->nextComponents()->insert(dispose1);
00065         // insert the model into the simulator
00066         simulator->models()->insert(model);
00067         // if the model is ok then save the model into a text file
00068         if (model->check())
00069             model->save("./temp/firstExampleOfSimulation.txt");
00070     } else {
00071         simulator->models()->loadModel("./temp/firstExampleOfSimulation.txt");
00072         model = simulator->models()->current();
00073     }
00074
00075     // execute the simulation until completed and show the report
00076     model->simulation()->start();
00077     simulator->~Simulator();
00078     return 0;
00079 };
00080
```

7.143 FirstExampleOfSimulation.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [FirstExampleOfSimulation](#)

7.144 FirstExampleOfSimulation.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: FirstExampleOfSimulation.h
00009  * Author: rlcancian
00010 *
00011  * Created on 3 de Setembro de 2019, 18:34
00012 */
00013
00014 #ifndef FIRSTEXAMPLEOFSIMULATION_H
00015 #define FIRSTEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class FirstExampleOfSimulation: public
00020     BaseConsoleGenesysApplication {
00021     public:
00022         FirstExampleOfSimulation();
00023     public:
00024         virtual int main(int argc, char** argv);
00025     };
00026 #endif /* FIRSTEXAMPLEOFSIMULATION_H */
00027
```

7.145 Fitter_if.h File Reference

```
#include <string>
```

Classes

- class [Fitter_if](#)

7.146 Fitter_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Fitter_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 14 de Agosto de 2018, 14:05
00012 */
```

```

00013
00014 #ifndef FITTER_IF_H
00015 #define FITTER_IF_H
00016
00017 #include <string>
00018
00019 class Fitter_if {
00020 public:
00021     virtual bool isNormalDistributed(double confidencelevel) = 0;
00022     virtual void fitUniform(double *sqrerror, double *min, double *max) = 0;
00023     virtual void fitTriangular(double *sqrerror, double *min, double *mo, double *max) = 0;
00024     virtual void fitNormal(double *sqrerror, double *avg, double *stddev) = 0;
00025     virtual void fitExpo(double *sqrerror, double *avg1) = 0;
00026     virtual void fitErlang(double *sqrerror, double *avg, double *m) = 0;
00027     virtual void fitBeta(double *sqrerror, double *alpha, double *beta, double *infLimit, double *
supLimit) = 0;
00028     virtual void fitWeibull(double *sqrerror, double *alpha, double *scale) = 0;
00029     virtual void fitAll(double *sqrerror, std::string *name) = 0;
00030 public:
00031     virtual void setDataFilename(std::string datafilename) = 0;
00032     virtual std::string getDataFilename() = 0;
00033 };
00034
00035 #endif /* FITTER_IF_H */
00036

```

7.147 FitterDefaultImpl1.cpp File Reference

```
#include "FitterDefaultImpl1.h"
```

7.148 FitterDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   FitterDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 20 de Maio de 2019, 20:48
00012 */
00013
00014 #include "FitterDefaultImpl1.h"
00015
00016 FitterDefaultImpl1::FitterDefaultImpl1() {
00017 }
00018
00019
00020 bool FitterDefaultImpl1::isNormalDistributed(double confidencelevel)
00021 {
00022     return true;
00023 }
00024 void FitterDefaultImpl1::fitUniform(double *sqrerror, double *min, double *
max) {
00025
00026 }
00027
00028 void FitterDefaultImpl1::fitTriangular(double *sqrerror, double *min,
double *mo, double *max) {
00029
00030 }
00031
00032 void FitterDefaultImpl1::fitNormal(double *sqrerror, double *avg, double *
stddev) {
00033
00034 }
00035
00036 void FitterDefaultImpl1::fitExpo(double *sqrerror, double *avg1) {
00037 }
00038
00039 void FitterDefaultImpl1::fitErlang(double *sqrerror, double *avg, double *m) {
00040

```

```

00041 }
00042
00043 void FitterDefaultImpl1::fitBeta(double *sqrerror, double *alpha, double *beta,
00044     double *infLimit, double *supLimit) {
00045 }
00046
00047 void FitterDefaultImpl1::fitWeibull(double *sqrerror, double *alpha, double *
00048     scale) {
00049 }
00050
00051 void FitterDefaultImpl1::fitAll(double *sqrerror, std::string *name) {
00052
00053 }
00054
00055 void FitterDefaultImpl1::setDataFilename(std::string dataFilename) {
00056
00057 }
00058
00059 std::string FitterDefaultImpl1::getDataFilename() {
00060     return ""; //todo
00061 }

```

7.149 FitterDefaultImpl1.h File Reference

```
#include "Fitter_if.h"
```

Classes

- class [FitterDefaultImpl1](#)

7.150 FitterDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  FitterDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:48
00012 */
00013
00014 #ifndef FITTERDEFAULTIMPL1_H
00015 #define FITTERDEFAULTIMPL1_H
00016
00017 #include "Fitter_if.h"
00018
00019 class FitterDefaultImpl1 : public Fitter_if {
00020 public:
00021     FitterDefaultImpl1();
00022     virtual ~FitterDefaultImpl1() = default;
00023 public:
00024     bool isNormalDistributed(double confidencelevel);
00025     void fitUniform(double *sqrerror, double *min, double *max);
00026     void fitTriangular(double *sqrerror, double *min, double *mo, double *max);
00027     void fitNormal(double *sqrerror, double *avg, double *stddev);
00028     void fitExpo(double *sqrerror, double *avg1);
00029     void fitErlang(double *sqrerror, double *avg, double *m);
00030     void fitBeta(double *sqrerror, double *alpha, double *beta, double *infLimit, double *supLimit);
00031     void fitWeibull(double *sqrerror, double *alpha, double *scale);
00032     void fitAll(double *sqrerror, std::string *name);
00033 public:
00034     void setDataFilename(std::string dataFilename);
00035     std::string getDataFilename();
00036 private:
00037     std::string _dataFilename = "";
00038 };
00039
00040 #endif /* FITTERDEFAULTIMPL1_H */
00041

```

7.151 Formula.cpp File Reference

```
#include "Formula.h"
#include <iostream>
#include "ElementManager.h"
#include "Model.h"
#include "Traits.h"
```

7.152 Formula.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Formula.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 20 de Junho de 2019, 00:56
00012 */
00013
00014 #include "Formula.h"
00015 #include <iostream>
00016 #include "ElementManager.h"
00017 #include "Model.h"
00018 #include "Traits.h"
00019
00020 Formula::Formula(Model* model, std::string name) :
    ModelElement(model, Util::TypeOf<Formula>(), name) {
00021     //_myPrivateParser = new Traits<Parser_if>::Implementation(_parentModel);
00022 }
00023
00024 std::string Formula::show() {
00025     std::string expressions = "";
00026     unsigned int i = 0;
00027     // for (std::list<std::string>::iterator it = _formulaExpressions->list()->begin(); it != _formulaExpressions->list()->end(); it++) {
00028         //expressions += "expression[" + std::to_string(i++) + "]=" + (*it) + "\n";
00029     //}
00030     return ModelElement::show() + expressions;
00031 }
00032
00033 unsigned int Formula::size() {
00034     return _formulaExpressions->size();
00035 }
00036
00037 void Formula::setExpression(std::string index, std::string formulaExpression) {
00038     std::map<std::string, std::string>::iterator mapIt = _formulaExpressions->find(index);
00039     if (mapIt != _formulaExpressions->end()) {//found
00040         (*mapIt).second = formulaExpression;
00041     } else {//not found
00042         _formulaExpressions->insert({index, formulaExpression});
00043     }
00044 }
00045
00046 std::string Formula::expression(std::string index) {
00047     std::map<std::string, std::string>::iterator it = _formulaExpressions->find(index);
00048     if (it == _formulaExpressions->end()) {
00049         return ""; // index does not exist. No formula expressions returned. \todo: Should it be traced?.
00050     } else {
00051         return it->second;
00052     }
00053 }
00054
00055 void Formula::setExpression(std::string formulaExpression) {
00056     setExpression("", formulaExpression);
00057 }
00058
00059 std::string Formula::expression() {
00060     return expression("");
00061 }
00062
00063 double Formula::value(std::string index) {
00064     std::string strexpression = this->expression(index);
00065     double value = 0.0;
```

```

00066     try {
00067         value = _parentModel->parseExpression(strexpression);
00068     } catch (const std::exception& e) {
00069         _parentModel->tracer()->traceError(e, "Error parsing formula \"+
00070             strexpression+\"");
00071     }
00072 }
00073
00074 double Formula::value() {
00075     return value("");
00076 }
00077
00078 PluginInformation* Formula::GetPluginInformation() {
00079     PluginInformation* info = new PluginInformation(Util::TypeOf<Formula>
00080     (), &Formula::LoadInstance);
00081     return info;
00082 }
00083
00084 ModelElement* Formula::LoadInstance(Model* model,
00085     std::map<std::string, std::string>* fields) {
00086     Formula* newElement = new Formula(model);
00087     try {
00088         newElement->_loadInstance(fields);
00089     } catch (const std::exception& e) {
00090     }
00091     return newElement;
00092 }
00093 bool Formula::_loadInstance(std::map<std::string, std::string>* fields) {
00094     return ModelElement::_loadInstance(fields);
00095 }
00096
00097 std::map<std::string, std::string>* Formula::_saveInstance() {
00098     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00099     //fields->emplace("...", std::to_string(this->...));
00100     return fields;
00101 }
00102
00103 bool Formula::_check(std::string* errorMessage) {
00104     std::string errorMsg = "";
00105     bool res, resAll = true;
00106     //unsigned int i = 0;
00107     for (std::map<std::string, std::string>::iterator it = _formulaExpressions->begin(); it != _formulaExpressions->end(); it++) {
00108         res = _parentModel->checkExpression((*it).second, "formula expression[" +
00109             (*it).first + "]", &errorMsg);
00110         if (!res) {
00111             _parentModel->tracer()->trace(
00112                 Util::TraceLevel::errorFatal, "Error parsing expression \"+ " + (*it).second + "
00113                     +"");
00114         }
00115     }
00116     resAll &= res;
00117 }
00118
00119     return resAll;
00120 }
```

7.153 Formula.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
#include "Parser_if.h"
```

Classes

- class [Formula](#)

7.154 Formula.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Formula.h
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Junho de 2019, 00:56
00012 */
00013
00014 #ifndef FORMULA_H
00015 #define FORMULA_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020 #include "Parser_if.h"
00021
00022 class Formula : public ModelElement {
00023 public:
00024     Formula(Model* model, std::string name = "");
00025     virtual ~Formula() = default;
00026 public: // virtual
00027     virtual std::string show();
00028 public:
00029     unsigned int size();
00030     void setExpression(std::string index, std::string formulaExpression);
00031     void setExpression(std::string formulaExpression);
00032     std::string expression(std::string index);
00033     std::string expression();
00034     double value();
00035     double value(std::string index);
00036 public: // statics
00037     static PluginInformation* GetPluginInformation();
00038     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00039                                         std::string>* fields);
00039 protected: // must be overridden by derived classes
00040     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00041     virtual std::map<std::string, std::string>* _saveInstance();
00042     virtual bool _check(std::string* errorMessage);
00043 private:
00044 private:
00045     std::map<std::string, std::string>* _formulaExpressions = new std::map<std::string, std::string>(); // map<index, formula>
00046 };
00047
00048 #endif /* FORMULA_H */
00049

```

7.155 FourthExampleOfSimulation.cpp File Reference

```

#include "FourthExampleOfSimulation.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Write.h"
#include "ElementManager.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"
#include "ProbDistrib.h"
#include "EntityGroup.h"
#include "Set.h"

```

7.156 FourthExampleOfSimulation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FourthExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 24 de Setembro de 2019, 20:56
00012 */
00013
00014 #include "FourthExampleOfSimulation.h"
00015
00016 // GEnSyS Simulator
00017 #include "Simulator.h"
00018
00019 // Model Components
00020 #include "Create.h"
00021 #include "Delay.h"
00022 #include "Dispose.h"
00023 #include "Seize.h"
00024 #include "Release.h"
00025 #include "Assign.h"
00026 #include "Record.h"
00027 #include "Decide.h"
00028 #include "Write.h"
00029
00030 // Model elements
00031 #include "ElementManager.h"
00032 #include "EntityType.h"
00033 #include "Attribute.h"
00034 #include "Variable.h"
00035 #include "ProbDistrib.h"
00036 #include "EntityGroup.h"
00037 #include "Set.h"
00038
00039 FourthExampleOfSimulation::FourthExampleOfSimulation()
{
00040 }
00041
00042 int FourthExampleOfSimulation::main(int argc, char** argv) {
00043     Simulator* simulator = new Simulator();
00044     // simulator->tracer()->setTraceLevel(Util::TraceLevel::blockArrival);
00045     this->setDefaultTraceHandlers(simulator->tracer());
00046     this->insertFakePluginsByHand(simulator);
00047     bool wantToCreateNewModelAndSaveInsteadOfJustLoad = true;
00048     Model* model;
00049     if (wantToCreateNewModelAndSaveInsteadOfJustLoad) {
00050         model = new Model(simulator);
00051         // build the simulation model
00052         ModelInfo* infos = model->infos();
00053         infos->setNumberOfReplications(1);
00054         EntityType* part = new EntityType(model, "Part");
00055         // model->insert(part);
00056         Create* createl = new Create(model);
00057         createl->setEntityType(part);
00058         createl->setTimeBetweenCreationsExpression("norm(1.5,0.5)");
00059         createl->setTimeUnit(Util::TimeUnit::second);
00060         createl->setEntitiesPerCreation(1);
00061         // model->insert(createl);
00062         Assign* assignl = new Assign(model);
00063         assignl->assignments()->insert(new Assign::Assignment("varNextIndex", "
00064             varNextIndex+1"));
00065         assignl->assignments()->insert(new Assign::Assignment("index", "
00066             varNextIndex"));
00067         // model->insert(assignl);
00068         Attribute* attrl = new Attribute(model, "index");
00069         // model->insert(attrl);
00070         Variable* varl = new Variable(model, "varNextIndex");
00071         // model->insert(varl);
00072         Write* writel = new Write(model);
00073         writel->setWriteToType(Write::WriteToType::SCREEN);
00074         writel->writeElements()->insert(new WriteElement("Atributo index: "));
00075         writel->writeElements()->insert(new WriteElement("index", true, true));
00076         writel->writeElements()->insert(new WriteElement("Variável nextIndex: "));
00077         writel->writeElements()->insert(new WriteElement("NR(Machine_1)", true));
00078
;
```

```

00078     write1->writeElements()->insert(new WriteElement(" ", ""));
00079     write1->writeElements()->insert(new WriteElement("NR(Machine_2)", true))
00080     ;
00081     write1->writeElements()->insert(new WriteElement(" ", ""));
00082     write1->writeElements()->insert(new WriteElement("NR(Machine_3)", true,
00083     true));
00084     write1->writeElements()->insert(new WriteElement("Estado das máquinas: "
00085     ));
00086     write1->writeElements()->insert(new WriteElement("STATE(Machine_1)", true));
00087     write1->writeElements()->insert(new WriteElement("STATE(Machine_2)", true));
00088     write1->writeElements()->insert(new WriteElement("STATE(Machine_3)", true,
00089     true));
00090     write1->writeElements()->insert(new WriteElement("Quantidade de máquinas
00091     ocupadas no Set: "));
00092     write1->writeElements()->insert(new WriteElement("SETSUM(Machine_Set)", true,
00093     true));
00094     write1->writeElements()->insert(new WriteElement("Quantidade de
00095     entidades na fila 3: "));
00096     write1->writeElements()->insert(new WriteElement("NQ(Queue_Seize_3)", true,
00097     true));
00098     write1->writeElements()->insert(new WriteElement("Somatório do atributo
00099     'index' das entidades na fila 3: "));
00100     write1->writeElements()->insert(new WriteElement("SAQUE(Queue_Seize_3,index)", true,
00101     true));
00102     write1->writeElements()->insert(new WriteElement("Valor do atributo
00103     'index' da 2ª entidade na fila 3: "));
00104     write1->writeElements()->insert(new WriteElement("AQUE(Queue_Seize_3,2,index)", true,
00105     true));
00106     write1->writeElements()->insert(new WriteElement("Tempo médio das
00107     entidades na fila 3: "));
00108     write1->writeElements()->insert(new WriteElement("TAVG(Queue_Seize_3.Time_In_Queue)", true,
00109     true));
00110     // model->insert(write1);
00111     //
00112     Resource* machine1 = new Resource(model, "Machine_1");
00113     machine1->setCapacity(1);
00114     // model->insert(machine1);
00115     Resource* machine2 = new Resource(model, "Machine_2");
00116     machine2->setCapacity(2);
00117     // model->insert(machine2);
00118     Resource* machine3 = new Resource(model, "Machine_3");
00119     machine3->setCapacity(3);
00120     // model->insert(machine3);
00121     Set* machSet = new Set(model, "Machine_Set");
00122     machSet->setSetOfType(Util::TypeOf<Resource>());
00123     machSet->getElementSet()->insert(machine1);
00124     machSet->getElementSet()->insert(machine2);
00125     machSet->getElementSet()->insert(machine3);
00126     // model->insert(machSet);
00127     Decide* decide1 = new Decide(model);
00128     decide1->getConditions()->insert("NR(Machine_1)<MR(Machine_1)");
00129     decide1->getConditions()->insert("NR(Machine_2)<MR(Machine_2)");
00130     // model->insert(decide1);
00131     Queue* queueSeize1 = new Queue(model, "Queue_Seize_1");
00132     queueSeize1->setOrderRule(Queue::OrderRule::FIFO);
00133     // model->insert(queueSeize1);
00134     Seize* seize1 = new Seize(model);
00135     seize1->setResource(machine1);
00136     seize1->setQueue(queueSeize1);
00137     // model->insert(seize1);
00138     Delay* delay1 = new Delay(model);
00139     delay1->setDelayExpression("norm(15,1)");
00140     delay1->setDelayTimeUnit(Util::TimeUnit::second);
00141     // model->insert(delay1);
00142     Release* release1 = new Release(model);
00143     release1->setResource(machine1);
00144     // model->insert(release1);
00145     Queue* queueSeize2 = new Queue(model, "Queue_Seize_2");
00146     queueSeize2->setOrderRule(Queue::OrderRule::FIFO);
00147     // model->insert(queueSeize2);
00148     Seize* seize2 = new Seize(model);
00149     seize2->setResource(machine2);
00150     seize2->setQueue(queueSeize2);
00151     // model->insert(seize2);
00152     Delay* delay2 = new Delay(model);
00153     delay2->setDelayExpression("norm(15,1)");
00154     delay2->setDelayTimeUnit(Util::TimeUnit::second);
00155     // model->insert(delay2);
00156     Release* release2 = new Release(model);
00157     release2->setResource(machine2);
00158     // model->insert(release2);
00159     Queue* queueSeize3 = new Queue(model, "Queue_Seize_3");
00160     queueSeize3->setOrderRule(Queue::OrderRule::FIFO);

```

```

00149 // model->insert(queueSeize3);
00150 Seize* seize3 = new Seize(model);
00151 seize3->setResource(machine3);
00152 seize3->setQueue(queueSeize3);
00153 // model->insert(seize3);
00154 Delay* delay3 = new Delay(model);
00155 delay3->setDelayExpression("norm(15,1)");
00156 delay3->setDelayTimeUnit(Util::TimeUnit::second);
00157 // model->insert(delay3);
00158 Release* release3 = new Release(model);
00159 release3->setResource(machine3);
00160 // model->insert(release3);
00161 Dispose* dispose1 = new Dispose(model);
00162 // model->insert(dispose1);
00163 //
00164 createl->nextComponents()->insert(assign1);
00165 assign1->nextComponents()->insert(writel);
00166 writel->nextComponents()->insert(decide1);
00167 decide1->nextComponents()->insert(seize1);
00168 decide1->nextComponents()->insert(seize2);
00169 decide1->nextComponents()->insert(seize3);
00170 seize1->nextComponents()->insert(delay1);
00171 delay1->nextComponents()->insert(release1);
00172 release1->nextComponents()->insert(dispose1);
00173 seize2->nextComponents()->insert(delay2);
00174 delay2->nextComponents()->insert(release2);
00175 release2->nextComponents()->insert(dispose1);
00176 seize3->nextComponents()->insert(delay3);
00177 delay3->nextComponents()->insert(release3);
00178 release3->nextComponents()->insert(dispose1);
00179 //
00180 simulator->models()->insert(model);
00181 if (model->check()) {
00182     model->save("./temp/forthExampleOfSimulation.txt");
00183 }
00184 } else { // load previously saved model
00185 simulator->models()->loadModel("./temp/forthExampleOfSimulation.txt");
00186 model = simulator->models()->current();
00187 }
00188 this->setDefaultEventHandlers(model->onEvents());
00189 model->simulation()->start();
00190 return 0;
00191 }
00192

```

7.157 FourthExampleOfSimulation.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [FourthExampleOfSimulation](#)

7.158 FourthExampleOfSimulation.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: FourthExampleOfSimulation.h
00009 * Author: rlcancian
00010 *
00011 * Created on 24 de Setembro de 2019, 20:56
00012 */
00013
00014 #ifndef FOURTHEXAMPLEOFSIMULATION_H
00015 #define FOURTHEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018

```

```

00019 class FourthExampleOfSimulation : public
00020     BaseConsoleGenesysApplication {
00021     public:
00022         FourthExampleOfSimulation();
00023     public:
00024         virtual int main(int argc, char** argv);
00025     };
00026 #endif /* FOURTHEXAMPLEOFSIMULATION_H */
00027

```

7.159 Free.cpp File Reference

```
#include "Free.h"
#include "Model.h"
```

7.160 Free.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Free.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 13:16
00012 */
00013
00014 #include "Free.h"
00015
00016 #include "Model.h"
00017
00018 Free::Free(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Free>(), name) {
00020
00021
00022     std::string Free::show() {
00023         return ModelComponent::show() + "";
00024     }
00025
00026     ModelComponent* Free::LoadInstance(Model* model, std::map<std::string,
00027         std::string>* fields) {
00028         Free* newComponent = new Free(model);
00029         try {
00030             newComponent->_loadInstance(fields);
00031         } catch (const std::exception& e) {
00032
00033         return newComponent;
00034     }
00035
00036     void Free::_execute(Entity* entity) {
00037         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038             entity forward");
00039         this->_parentModel->sendEntityToComponent(entity, this->
00040             nextComponents()->frontConnection(), 0.0);
00041     }
00042     bool Free::_loadInstance(std::map<std::string, std::string>* fields) {
00043         bool res = ModelComponent::_loadInstance(fields);
00044         if (res) {
00045             //...
00046         }
00047         return res;
00048     }
00049     void Free::_initBetweenReplications() {
00050
00051     std::map<std::string, std::string>* Free::_saveInstance() {
00052         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053         ();

```

```

00054     //...
00055     return fields;
00056 }
00057
00058 bool Free::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Free::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Free>(),
00066     &Free::LoadInstance);
00067     // ...
00068     return info;
00069 }
00070

```

7.161 Free.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Free](#)

7.162 Free.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Free.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:16
00012 */
00013
00014 #ifndef FREE_H
00015 #define FREE_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Free : public ModelComponent {
00020 public: // constructors
00021     Free(Model* model, std::string name(""));
00022     virtual ~Free() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040 #endif /* FREE_H */
00041

```

7.163 FullSimulationOfComplexModel.cpp File Reference

```
#include "FullSimulationOfComplexModel.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Dummy.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"
#include "ProbDistrib.h"
#include "EntityGroup.h"
#include "Formula.h"
#include "OLD_ODElement.h"
```

7.164 FullSimulationOfComplexModel.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 #include "FullSimulationOfComplexModel.h"
00008
00009 // GEnSyS Simulator
00010 #include "Simulator.h"
00011
00012 // Model Components
00013 #include "Create.h"
00014 #include "Delay.h"
00015 #include "Dispose.h"
00016 #include "Seize.h"
00017 #include "Release.h"
00018 #include "Assign.h"
00019 #include "Record.h"
00020 #include "Decide.h"
00021 #include "Dummy.h"
00022
00023 // Model elements
00024 #include "EntityType.h"
00025 #include "Attribute.h"
00026 #include "Variable.h"
00027 #include "ProbDistrib.h"
00028 #include "EntityGroup.h"
00029 #include "Formula.h"
00030 #include "OLD_ODElement.h"
00031
00032 FullSimulationOfComplexModel::FullSimulationOfComplexModel
00033 () {
00034 }
00035
00040 int FullSimulationOfComplexModel::main(int argc, char** argv) {
00041     Simulator* simulator = new Simulator();
00042
00043     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00044     this->insertFakePluginsByHand(simulator);
00045
00046     // creates an empty model
00047     Model* model = new Model(simulator);
00048
00049     // Handle traces and simulation events to output them
00050     TraceManager* tm = model->tracer();
00051     this->setDefaultTraceHandlers(tm);
```

```

00052 // get easy access to classes used to insert components and elements into a model
00053 ComponentManager* components = model->components();
00054 ElementManager* elements = model->elements();
00055
00056 //
00057 // build the simulation model
00058 //
00059 ModelInfo* infos = model->infos();
00060 infos->setAnalystName("Your name");
00061 infos->setProjectTitle("The title of the project");
00062 infos->setDescription("This simulation model tests the components and elements that have
been implemented so far.");
00063 infos->setReplicationLength(1e3);
00064 infos->setReplicationLengthTimeUnit(
    Util::TimeUnit::minute);
00065 infos->setNumberofReplications(30);
00066
00067 EntityType* entityType1 = new EntityType(model, "Representative_EntityType");
00068 elements->insert(entityType1);
00069
00070 Create* create1 = new Create(model);
00071 create1->setEntityType(entityType1);
00072 create1->setTimeBetweenCreationsExpression("EXPO(5)");
00073 create1->setTimeUnit(Util::TimeUnit::minute);
00074 create1->setEntitiesPerCreation(1);
00075 components->insert(create1);
00076
00077 Attribute* attribute1 = new Attribute(model, "Attribute_1");
00078 elements->insert(attribute1);
00079 Variable* variable1 = new Variable(model, "Variable_1");
00080 elements->insert(variable1);
00081
00082 Assign* assign1 = new Assign(model);
00083 Assign::Assignment* attrib2Assignment = new
    Assign::Assignment("Variable_1", "Variable_1 + 1");
00084 assign1->assignments()->insert(attrib2Assignment);
00085 Assign::Assignment* attrib1Assignment = new
    Assign::Assignment("Attribute_1", "Variable_1");
00086 assign1->assignments()->insert(attrib1Assignment);
00087 components->insert(assign1);
00088
00089
00090 Decide* decide1 = new Decide(model);
00091 decide1->getConditions()->insert("UNIF(0,1)>0.5");
00092
00093 Resource* maquinal = new Resource(model, "Máquina 1");
00094 maquinal->setCapacity(1);
00095 elements->insert(maquinal);
00096
00097 Queue* filaSeizel = new Queue(model);
00098 filaSeizel->setOrderRule(Queue::OrderRule::FIFO);
00099 elements->insert(filaSeizel);
00100
00101 Seize* seizel = new Seize(model);
00102 seize1->setResource(maquinal);
00103 seize1->setQueue(filaSeizel);
00104 components->insert(seizel);
00105
00106 Delay* delay1 = new Delay(model);
00107 delay1->setDelayExpression("NORM(5,1.5)");
00108 delay1->setDelayTimeUnit(Util::TimeUnit::minute);
00109
00110 components->insert(delay1);
00111
00112 Release* release1 = new Release(model);
00113 release1->setResource(maquinal);
00114 components->insert(release1);
00115
00116 Record* record1 = new Record(model);
00117 record1->setExpressionName("Tempo total no sistema");
00118 record1->setExpression("TNOW - Entity.ArrivalTime");
00119 record1->setFilename("./temp/TotalTimeInSystem.txt");
00120 components->insert(record1);
00121
00122 Dispose* dispose1 = new Dispose(model);
00123 components->insert(dispose1);
00124
00125 // connect model components to create a "workflow" -- should always start from a SourceModelComponent
and end at a SinkModelComponent (it will be checked)
00126 create1->nextComponents()->insert(assign1);
00127 assign1->nextComponents()->insert(decide1);
00128 decide1->nextComponents()->insert(seizel);
00129 seize1->nextComponents()->insert(delay1);
00130 delay1->nextComponents()->insert(release1);
00131 release1->nextComponents()->insert(record1);
00132 record1->nextComponents()->insert(dispose1);
00133

```

```

00134
00135 // insert the model into the simulator
00136 simulator->models()->insert(model);
00137
00138 // if the model is ok then save the model into a text file
00139 if (model->check()) {
00140 model->save("./temp/fullSimulationModel.txt");
00141 }
00142
00143 // execute the simulation
00144 model->simulation()->start();
00145
00146 return 0;
00147 };

```

7.165 FullSimulationOfComplexModel.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
#include "Model.h"
```

Classes

- class [FullSimulationOfComplexModel](#)

7.166 FullSimulationOfComplexModel.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: buildSimpleModel1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Outubro de 2018, 19:18
00012 */
00013
00014 #ifndef BUILDSIMULATIONMODEL02_H
00015 #define BUILDSIMULATIONMODEL02_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018 #include "Model.h"
00019
00020 class FullSimulationOfComplexModel: public
00021     BaseConsoleGenesysApplication {
00022 public:
00023     FullSimulationOfComplexModel();
00024 public:
00025     virtual int main(int argc, char** argv);
00026 };
00027 #endif /* BUILDSIMULATIONMODEL02_H */
00028

```

7.167 Functor.h File Reference

7.168 Functor.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006

```

```

00007 /*
00008 * File: Functor.h
00009 * Author: joao.vicentesouto
00010 *
00011 * Created on November 3, 2018, 11:04 AM
00012 */
00013
00014 #ifndef FUNCTOR_H
00015 #define FUNCTOR_H
00016
00017 #endif /* FUNCTOR_H */

```

7.169 GenesysApplication_if.h File Reference

Classes

- class [GenesysApplication_if](#)

7.170 GenesysApplication_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: GenesysApplication_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Outubro de 2018, 19:14
00012 */
00013
00014 #ifndef GENESYSAPPLICATION_IF_H
00015 #define GENESYSAPPLICATION_IF_H
00016
00017 class GenesysApplication_if {
00018 public:
00019     virtual int main(int argc, char** argv) = 0;
00020 };
00021
00022 #endif /* GENESYSAPPLICATION_IF_H */
00023

```

7.171 GenesysConsole.cpp File Reference

```

#include "GenesysConsole.h"
#include "Simulator.h"
#include "Assign.h"
#include <regex>
#include <fstream>
#include <assert.h>
#include "ProbDistrib.h"

```

7.172 GenesysConsole.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: GenesysShell.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:02
00012 */
00013
00014 #include "GenesysConsole.h"
00015 #include "Simulator.h"
00016 #include "Assign.h"
00017 #include <regex>
00018 #include <fstream>
00019 #include <assert.h>
00020
00021 #include "ProbDistrib.h"
00022
00023 GenesysConsole::GenesysConsole() {
00024     _commands->setSortFunc([](const ShellCommand* a, const ShellCommand * b) {
00025         return a->shortname < b->shortname;
00026     });
00027     _commands->insert(new ShellCommand("q", "quit", "", "Quit ReGenesys shell. Same as exit.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdQuit)));
00028     _commands->insert(new ShellCommand("x", "exit", "", "Exit ReGenesys shell. Same as quit.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdQuit)));
00029     _commands->insert(new ShellCommand("h", "help", "", "Show help for commands.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdHelp)));
00030     _commands->insert(new ShellCommand("ms", "modelsave", "<filename>", "Save model.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelSave)));
00031     _commands->insert(new ShellCommand("ml", "modelload", "<filename>", "Load Model.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelLoad)));
00032     _commands->insert(new ShellCommand("rf", "readfile", "<filename>", "Read and execute shell command from file.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdScript)));
00033     _commands->insert(new ShellCommand("v", "version", "", "Show the version.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdVersion)));
00034     _commands->insert(new ShellCommand("ss", "start", "", "Start simulation.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStart)));
00035     _commands->insert(new ShellCommand("mc", "modelcheck", "", "Check model.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelCheck)));
00036     _commands->insert(new ShellCommand("mh", "modelshow", "", "Show model.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdModelShow)));
00037     _commands->insert(new ShellCommand("ps", "step", "", "step simulation.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStep)));
00038     _commands->insert(new ShellCommand("ts", "stop", "", "Stop simulation.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdStop)));
00039     _commands->insert(new ShellCommand("sr", "showreport", "", "Show simulation report.", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdShowReport)));
00040     _commands->insert(new ShellCommand("tl", "tracelevel", "<0|1|2|...|7>", "Set the trace level (the bigger the most verbose).", DefineExecuterMember<GenesysConsole>(this, &GenesysConsole::cmdTraceLevel)));
00041 }
00042
00043
00044 void GenesysConsole::cmdTraceLevel() {
00045     Trace("Set trace level");
00046     try {
00047         int tlnum = std::stoi(_parameter);
00048         Util::TraceLevel tl = static_cast<Util::TraceLevel>(tlnum);
00049         _simulator->models()->current()->tracer()->setTraceLevel(tl);
00050     } catch (...) {
00051         Trace("Error setting trace level");
00052     }
00053 }
00054
00055 void GenesysConsole::cmdModelCheck() {
00056     Trace("Check model");
00057     try {
00058         _simulator->models()->current()->check();
00059     } catch (...) {
00060         Trace("Error checking model");
00061     }
00062 }
00063
00064 void GenesysConsole::cmdStart() {
00065     Trace("Start simulation");
00066     try {
00067         _simulator->models()->current()->simulation()->start();
00068     } catch (...) {
```

```
00069     Trace("Error starting simulation");
00070 }
00071 }
00072
00073 void GenesysConsole::cmdStep() {
00074     Trace("Step simulation");
00075     try {
00076         _simulator->models()->current()->simulation()->step();
00077     } catch (...) {
00078         Trace("Error stepping simulation");
00079     }
00080 }
00081
00082 void GenesysConsole::cmdStop() {
00083     Trace("Stop simulation");
00084     try {
00085         _simulator->models()->current()->simulation()->stop();
00086     } catch (...) {
00087         Trace("Error stopping simulation");
00088     }
00089 }
00090
00091 void GenesysConsole::cmdShowReport() {
00092     Trace("Show report");
00093     try {
00094         _simulator->models()->current()->simulation()->
00095             reporter()->showSimulationStatistics();
00096     } catch (...) {
00097         Trace("Error showing reports");
00098     }
00099 }
00100 void GenesysConsole::cmdHelp() {
00101     ShellCommand* command;
00102     Trace("List of commands:");
00103     Trace(Util::SetW("Short", 6) + Util::SetW("Long", 12) +
00104         Util::SetW("Parameters", 15) + "Description");
00105     for (std::list<ShellCommand*>::iterator it = _commands->list()->begin(); it != _commands->
00106         list()->end(); it++) {
00107         //Trace("Unknown command. Type \"-h\" or \"help\" for help on possible commands.");
00108         command = (*it);
00109         Trace(Util::SetW(command->shortname, 6) + Util::SetW(command->longname, 12) +
00110             Util::SetW(command->parameters, 15) + command->description);
00111     }
00112 }
00113
00114 void GenesysConsole::cmdQuit() {
00115     Trace("Quiting/Exiting. Goodbye");
00116     exit(0);
00117 }
00118
00119 void GenesysConsole::cmdVersion() {
00120     Trace("ReGenesys Shell version 2019.0528");
00121 }
00122
00123 void GenesysConsole::cmdModelLoad() {
00124     Trace("Model load");
00125     try {
00126         std::string filename = _parameter;
00127         Model* model = new Model(this->_simulator);
00128         model->load(filename);
00129     } catch (...) {
00130         //        _commands
00131         Trace("    Error loading");
00132     }
00133 }
00134
00135 void GenesysConsole::cmdModelShow() {
00136     Trace("Model Show");
00137     try {
00138         _simulator->models()->current()->show();
00139     } catch (...) {
00140         //        _commands
00141         Trace("    Error showing");
00142     }
00143
00144 void GenesysConsole::cmdModelSave() {
00145     //this->_parameter;
00146 }
00147
00148 void GenesysConsole::cmdScript() {
00149     std::string filename = this->_parameter;
00150     //List<std::string>* arguments = new List<std::string>();
00151     std::ifstream commandfile;
00152     std::string inputText;
00153     try {
```

```

00152     commandfile.open(filename);
00153     while (getline(commandfile, inputText)) {
00154         this->tryExecuteCommand(inputText, "", "", " ");
00155     }
00156     commandfile.close();
00157     } catch (...) {
00158     Trace("    Error scripting");
00159     }
00160 }
00161
00162 void GenesysConsole::Trace(std::string message) {
00163     std::cout << message << std::endl;
00164 }
00165
00166 void GenesysConsole::run(List<std::string>* commandlineArgs) {
00167     Trace("ReGenesys Shell is running. Type your command. For help, type the command \"h\" or \"help\".");
00168     std::string inputText; //, shortPrefix, longPrefix, separator;
00169     while (true) {
00170         if (!commandlineArgs->empty()) {
00171             inputText = commandlineArgs->front();
00172             commandlineArgs->pop_front();
00173             tryExecuteCommand(inputText, "-", "--", "=");
00174         } else {
00175             std::cout << _prompt << " ";
00176             std::cin >> inputText;
00177             tryExecuteCommand(inputText, "", "", " ");
00178         }
00179     }
00180 }
00181 }
00182
00183 void GenesysConsole::tryExecuteCommand(std::string inputText, std::string shortPrefix, std::string
    longPrefix, std::string separator) {
00184     std::regex regex(R"([=]+)"); // split on space R"([\s]+)"
00185     std::sregex_token_iterator it{inputText.begin(), inputText.end(), regex, -1};
00186     std::vector<std::string> fields{it, {}};
00187     std::string typedCommandStr = fields[0];
00188     if (fields.size() > 1) { // its a comnd and a parameter in the form command=parameter
00189         assert(fields.size() == 2);
00190         _parameter = fields[1];
00191     } else {
00192         _parameter = "";
00193     }
00194     ShellCommand* command;
00195     bool found;
00196     std::transform(typedCommandStr.begin(), typedCommandStr.end(), typedCommandStr.begin(), ::tolower);
00197     if (typedCommandStr.substr(0, 1) != "#") {
00198         found = false;
00199         for (std::list<ShellCommand*>::iterator it = _commands->list()->begin(); it != _commands->
            list()->end(); it++) {
00200             //Trace("Unknown command. Type \"-h\" or \"help\" for help on possible commands.");
00201             command = (*it);
00202             if (typedCommandStr == shortPrefix + command->shortname || typedCommandStr == longPrefix + command
                ->longname) {
00203                 command->executer();
00204                 found = true;
00205                 break;
00206             }
00207         }
00208         if (!found) {
00209             Trace("Command \"\" + typedCommandStr + "\" not found. Type \"h\" or \"help\" for help.");
00210         }
00211     }
00212 }
00213 }
00214
00215 int GenesysConsole::main(int argc, char** argv) {
00216     //double res;
00217     //for (double x = -3.0; x <= 3.0; x += 0.05) {
00218     //    res = ProbDistrib::tStudent(x, 0, 1, 100);
00219     //    std::cout << x << ": " << res << std::endl;
00220     //}
00221     //return 0;
00222     List<std::string>* commandlineArgs = new List<std::string>();
00223     for (unsigned short i = 1; i < argc; i++) {
00224         std::string arg = argv[i];
00225         commandlineArgs->insert(arg);
00226     }
00227     //commandlineArgs->insert("-rf=temp/script.txt");
00228     //commandlineArgs->insert("-ml=models/genesysmodel.txt");
00229     this->run(commandlineArgs);
00230     return 0;
00231 }

```

7.173 GenesysConsole.h File Reference

```
#include "GenesysApplication_if.h"
#include "Simulator.h"
#include "List.h"
```

Classes

- class [GenesysConsole](#)

7.174 GenesysConsole.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: GenesysConsole.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:02
00012 */
00013
00014 #ifndef GENESYSCONSOLE_H
00015 #define GENESYSCONSOLE_H
00016
00017 #include "GenesysApplication_if.h"
00018 #include "Simulator.h"
00019 #include "List.h"
00020
00021 class GenesysConsole : public GenesysApplication_if {
00022 private:
00023     typedef std::function<void() > ExecuterMember;
00024
00025     template<typename Class>
00026     ExecuterMember DefineExecuterMember(Class * object, void (Class::*function)()) {
00027         return std::bind(function, object); //, std::placeholders::_1
00028     }
00029
00030     class ShellCommand {
00031     public:
00032
00033         ShellCommand(std::string shortname, std::string longname, std::string parameters, std::string
00034             description, ExecuterMember executer) {
00035             this->description = description;
00036             this->longname = longname;
00037             this->parameters = parameters;
00038             this->shortname = shortname;
00039             this->executer = executer;
00040         }
00041         std::string shortname;
00042         std::string longname;
00043         std::string parameters;
00044         std::string description;
00045         ExecuterMember executer;
00046     public:
00047         GenesysConsole();
00048         virtual ~GenesysConsole() = default;
00049     public:
00050         virtual int main(int argc, char** argv);
00051     public: // commands
00052         void cmdScript();
00053         void cmdHelp();
00054         void cmdQuit();
00055         void cmdModelLoad();
00056         void cmdModelCheck();
00057         void cmdStart();
00058         void cmdStep();
00059         void cmdStop();
00060         void cmdShowReport();
00061         void cmdModelSave();
00062         void cmdModelShow();
```

```

00063     void cmdVersion();
00064     void cmdTraceLevel();
00065 private:
00066     void run(List<std::string>* arguments);
00067     void Trace(std::string message);
00068     void tryExecuteCommand(std::string inputText, std::string shortPrefix, std::string longPrefix,
00069     std::string separator);
00070     Simulator* _simulator = new Simulator();
00071     std::string _parameter;
00072     List<ShellCommand*>* _commands = new List<ShellCommand*>();
00073     std::string _prompt = "$ReGenesys> ";
00074 };
00075
00076 #endif /* GENESYSCONSOLE_H */
00077

```

7.175 GenesysGUI.cpp File Reference

```
#include "GenesysGUI.h"
```

7.176 GenesysGUI.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: GenesysGUI.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:03
00012 */
00013
00014 #include "GenesysGUI.h"
00015
00016 GenesysGUI::GenesysGUI() {
00017 }
00018
00019 int GenesysGUI::main(int argc, char** argv) {
00020     return 0;
00021 }

```

7.177 GenesysGUI.h File Reference

```
#include "GenesysApplication_if.h"
```

Classes

- class [GenesysGUI](#)

7.178 GenesysGUI.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: GenesysGUI.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Maio de 2019, 13:03
00012 */
00013
00014 #ifndef GENESYSGUI_H
00015 #define GENESYSGUI_H
00016
00017 #include "GenesysApplication_if.h"
00018
00019 class GenesysGUI : public GenesysApplication_if {
00020 public:
00021     GenesysGUI();
00022     ~GenesysGUI() = default;
00023 public:
00024     virtual int main(int argc, char** argv);
00025 };
00026
00027 #endif /* GENESYSGUI_H */
00028

```

7.179 GenesysShell_if.h File Reference

```
#include "GenesysApplication_if.h"
#include "Util.h"
```

Classes

- class [GenesysShell_if](#)

7.180 GenesysShell_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: GenesysShell_if.h
00009  * Author: rlcancian
00010 *
00011 * Created on 24 de Maio de 2019, 11:02
00012 */
00013
00014 #ifndef GENESYSSHELL_IF_H
00015 #define GENESYSSHELL_IF_H
00016
00017 #include "GenesysApplication_if.h"
00018 #include "Util.h"
00019
00020 class GenesysShell_if : public GenesysApplication_if {
00021 public:
00022     virtual void openModel(std::string filename) = 0;
00023     virtual void saveModelAs(std::string filename) = 0;
00024     virtual void saveModel() = 0;
00025     virtual void listElements() = 0;
00026     virtual void listComponents() = 0;
00027     virtual void listHosts() = 0;
00028     virtual void listPlugins() = 0;
00029     virtual void deleteTraceFiles() = 0;

```

```

00030     virtual void traceLevel(Util::TraceLevel tracelevel) = 0;
00031     virtual void addPlugin(std::string filename) = 0;
00032     virtual void addFromFile(std::string filename) = 0;
00033     virtual void readCommandsFromFile(std::string filename) = 0;
00034     virtual void redirectTrace(std::string trace, std::string dest, std::string filename) = 0;
00035     virtual void closeModel() = 0;
00036     virtual void createModel() = 0;
00037     virtual void execLinuxCommand(std::string command) = 0;
00038     virtual void verboseMode(bool on) = 0;
00039     virtual void check() = 0;
00040     virtual void getGenesysInfo() = 0;
00041     virtual void getCommandLine() = 0;
00042     virtual void sendFile(std::string filename, std::string hostname, std::string portname) = 0;
00043     virtual void setActivationCode(std::string code) = 0;
00044     virtual void receiveFile(std::string filename) = 0;
00045     virtual void startSimulation() = 0;
00046     virtual void stepSimulation() = 0;
00047     virtual void stopSimulation() = 0;
00048     virtual void showInit() = 0;
00049     virtual void showHelp() = 0;
00050     virtual void showHostName() = 0;
00051 //virtual void execute() = 0;
00052 //property TraceHandler : TProcTrace write aTraceHandler;
00053 };
00054
00055 #endif /* GENESYSSHLL_IF_H */
00056

```

7.181 Halt.cpp File Reference

```
#include "Halt.h"
#include "Model.h"
```

7.182 Halt.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Halt.cpp
00009  * Author:  rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 13:17
00012 */
00013
00014 #include "Halt.h"
00015 #include "Model.h"
00016
00017 Halt::Halt(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<Halt>(), name) {
00019
00020
00021 std::string Halt::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Halt::LoadInstance(Model* model, std::map<std::string,
00026     std::string>* fields) {
00027     Halt* newComponent = new Halt(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031     }
00032     return newComponent;
00033 }
00034
00035 void Halt::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00037     entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
00039     nextComponents()->frontConnection(), 0.0);

```

```

00038 }
00039
00040 bool Halt::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Halt::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Halt::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053     ();
00054     //...
00055     return fields;
00056 }
00057 bool Halt::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Halt::GetPluginInformation(){
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Halt>(),
00065     &Halt::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069

```

7.183 Halt.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Halt](#)

7.184 Halt.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Halt.h
00009  * Author: rlcancian
00010  *
00011  * Created on 11 de Setembro de 2019, 13:17
00012 */
00013
00014 #ifndef HALT_H
00015 #define HALT_H
00016
00017 #include "ModelComponent.h"
00018
00039 class Halt : public ModelComponent {
00040 public: // constructors
00041     Halt(Model* model, std::string name="");
00042     virtual ~Halt() = default;
00043 public: // virtual
00044     virtual std::string show();
00045 public: // static
00046     static PluginInformation* GetPluginInformation();
00047     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00048                                         std::string>* fields);
00048 protected: // virtual

```

```

00049     virtual void _execute(Entity* entity);
00050     virtual void _initBetweenReplications();
00051     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052     virtual std::map<std::string, std::string>* _saveInstance();
00053     virtual bool _check(std::string* errorMessage);
00054 private: // methods
00055 private: // attributes 1:1
00056 private: // attributes 1:n
00057 };
00058
00059
00060 #endif /* HALT_H */
00061

```

7.185 Hold.cpp File Reference

```
#include "Hold.h"
#include "Model.h"
```

7.186 Hold.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Hold.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Hold.h"
00015 #include "Model.h"
00016
00017 Hold::Hold(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<Hold>(), name) {
00019
00020
00021 std::string Hold::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Hold::LoadInstance(Model* model, std::map<std::string,
00026     std::string>* fields) {
00027     Hold* newComponent = new Hold(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031     }
00032     return newComponent;
00033 }
00034
00035 void Hold::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00037     entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
00039     nextComponents()->frontConnection(), 0.0);
00040 }
00041
00042 bool Hold::_loadInstance(std::map<std::string, std::string>* fields) {
00043     bool res = ModelComponent::_loadInstance(fields);
00044     if (res) {
00045         //...
00046     }
00047     return res;
00048 }
00049
00050 void Hold::_initBetweenReplications() {
00051     std::map<std::string, std::string>* Hold::_saveInstance() {

```

```

00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053     ();
00054     //...
00055     return fields;
00056
00057     bool Hold::_check(std::string* errorMessage) {
00058         bool resultAll = true;
00059         //...
00060         return resultAll;
00061     }
00062
00063     PluginInformation* Hold::GetPluginInformation() {
00064         PluginInformation* info = new PluginInformation(Util::TypeOf<Hold>(),
00065         &Hold::LoadInstance);
00066         // ...
00067         return info;
00068     }
00069

```

7.187 Hold.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Hold](#)

7.188 Hold.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Hold.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef HOLD_H
00015 #define HOLD_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Hold : public ModelComponent {
00020 public: // constructors
00021     Hold(Model* model, std::string name = "");
00022     virtual ~Hold() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040
00041 #endif /* HOLD_H */
00042

```

7.189 HypothesisTester_if.h File Reference

```
#include <string>
```

Classes

- class [HypothesisTester_if](#)

7.190 HypothesisTester_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   HypothesisTester_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Agosto de 2018, 19:04
00012 */
00013
00014 #ifndef HYPOTHESISTESTER_IF_H
00015 #define HYPOTHESISTESTER_IF_H
00016
00017 #include <string>
00018
00019 class HypothesisTester_if {
00020 public:
00021     enum H1Comparition {
00022         DIFFERENT = 1,
00023         LESS_THAN = 2,
00024         GREATER_THAN = 3
00025     };
00026 public:
00027     /* \todo: all "test" methods should return double p-value, not bool */
00028     virtual double testAverage(double confidencelevel, double avg,
00029                                H1Comparition comp) = 0;
00030     virtual double testProportion(double confidencelevel, double prop,
00031                                    H1Comparition comp) = 0;
00032     virtual double testVariance(double confidencelevel, double var,
00033                                H1Comparition comp) = 0;
00034     virtual double testAverage(double confidencelevel, std::string secondPopulationDataFilename,
00035                                H1Comparition comp) = 0;
00036     virtual double testProportion(double confidencelevel, std::string
00037                                    secondPopulationDataFilename, H1Comparition comp) = 0;
00038     virtual double testVariance(double confidencelevel, std::string
00039                                    secondPopulationDataFilename, H1Comparition comp) = 0;
00040     virtual void setDatafilename(std::string datafilename) = 0;
00041     virtual std::string getDatafilename() = 0;
00042 };
00043 #endif /* HYPOTHESISTESTER_IF_H */
```

7.191 HypothesisTesterBoostImpl.cpp File Reference

```
#include "HypothesisTesterBoostImpl.h"
```

7.192 HypothesisTesterBoostImpl.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTesterBoostImpl.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 13:31
00012 */
00013
00014 #include "HypothesisTesterBoostImpl.h"
00015
00016 HypothesisTesterBoostImpl::HypothesisTesterBoostImpl()
00017 {
00018     double HypothesisTesterBoostImpl::testAverage(double confidencelevel,
00019         double avg, H1Comparition comp) {
00020         return 0.0; //todo
00021     }
00022     double HypothesisTesterBoostImpl::testProportion(double
00023         confidencelevel, double prop, H1Comparition comp) {
00024         return 0.0; //todo
00025     }
00026     double HypothesisTesterBoostImpl::testVariance(double
00027         confidencelevel, double var, H1Comparition comp) {
00028         return 0.0; //todo
00029     }
00030     double HypothesisTesterBoostImpl::testAverage(double confidencelevel,
00031         std::string secondPopulationDataFilename, H1Comparition comp) {
00032         return 0.0; //todo
00033     }
00034     double HypothesisTesterBoostImpl::testProportion(double
00035         confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp) {
00036         return 0.0; //todo
00037     }
00038     double HypothesisTesterBoostImpl::testVariance(double
00039         confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp) {
00040         return 0.0; //todo
00041     }
00042     void HypothesisTesterBoostImpl::setDataFilename(std::string
00043         dataFilename) {
00044     }
00045     std::string HypothesisTesterBoostImpl::getDataFilename() {
00046         return ""; //todo
00047     }

```

7.193 HypothesisTesterBoostImpl.h File Reference

```
#include "HypothesisTester_if.h"
#include "Integrator_if.h"
```

Classes

- class [HypothesisTesterBoostImpl](#)

7.194 HypothesisTesterBoostImpl.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTesterBoostImpl.h
00009 * Author: rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 13:31
00012 */
00013
00014 #ifndef HYPOTHESISTESTERBOOSTIMPL_H
00015 #define HYPOTHESISTESTERBOOSTIMPL_H
00016
00017 #include "HypothesisTester_if.h"
00018 #include "Integrator_if.h"
00019
00020 class HypothesisTesterBoostImpl : public
00021     HypothesisTester_if {
00022     public:
00023         HypothesisTesterBoostImpl();
00024         virtual ~HypothesisTesterBoostImpl() = default;
00025     public:
00026         virtual double testAverage(double confidencelevel, double avg,
00027             H1Comparition comp);
00027         virtual double testProportion(double confidencelevel, double prop,
00028             H1Comparition comp);
00029         virtual double testVariance(double confidencelevel, double var,
00030             H1Comparition comp);
00031         virtual double testAverage(double confidencelevel, std::string secondPopulationDataFilename,
00032             H1Comparition comp);
00032         virtual double testProportion(double confidencelevel, std::string
00033             secondPopulationDataFilename, H1Comparition comp);
00033         virtual double testVariance(double confidencelevel, std::string
00034             secondPopulationDataFilename, H1Comparition comp);
00035         virtual void setDataFilename(std::string dataFilename);
00036         virtual std::string getDataFilename();
00037     private:
00038         Integrator_if* _integrator;
00039     };
00040
00041 #endif /* HYPOTHESISTESTERBOOSTIMPL_H */
00042

```

7.195 HypothesisTesterDefaultImpl1.cpp File Reference

```
#include "HypothesisTesterDefaultImpl1.h"
#include "Traits.h"
```

7.196 HypothesisTesterDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTesterDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Agosto de 2018, 10:27
00012 */
00013
00014 #include "HypothesisTesterDefaultImpl1.h"
00015 #include "Traits.h"
00016
00017 HypothesisTesterDefaultImpl1::HypothesisTesterDefaultImpl1
00018     () {
00019         _integrator = new Traits<HypothesisTester_if>::IntegratorImplementation
00020     ();

```

```

00019 }
00020
00021
00022 double HypothesisTesterDefaultImpl1::testAverage(double
00023     confidencelevel, double avg, H1Comparition comp) {
00024     return 0.0; //todo
00025
00026 double HypothesisTesterDefaultImpl1::testProportion(double
00027     confidencelevel, double prop, H1Comparition comp) {
00028     return 0.0; //todo
00029
00030 double HypothesisTesterDefaultImpl1::testVariance(double
00031     confidencelevel, double var, H1Comparition comp) {
00032     return 0.0; //todo
00033
00034 double HypothesisTesterDefaultImpl1::testAverage(double
00035     confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp) {
00036     return 0.0; //todo
00037
00038 double HypothesisTesterDefaultImpl1::testProportion(double
00039     confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp) {
00040     return 0.0; //todo
00041
00042 double HypothesisTesterDefaultImpl1::testVariance(double
00043     confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp) {
00044     return 0.0; //todo
00045
00046 void HypothesisTesterDefaultImpl1::setDataFilename(std::string
00047     datafilename) {
00048
00049 std::string HypothesisTesterDefaultImpl1::getDataFilename() {
00050     return ""; //todo
00051 }
```

7.197 HypothesisTesterDefaultImpl1.h File Reference

```
#include "HypothesisTester_if.h"
#include "Integrator_if.h"
```

Classes

- class [HypothesisTesterDefaultImpl1](#)

7.198 HypothesisTesterDefaultImpl1.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: HypothesisTesterDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Agosto de 2018, 10:27
00012 */
00013
00014 #ifndef HYPOTHESISTESTERDEFAULTIMPL1_H
00015 #define HYPOTHESISTESTERDEFAULTIMPL1_H
00016
00017 #include "HypothesisTester_if.h"
00018 #include "Integrator_if.h"
00019
00020 class HypothesisTesterDefaultImpl1 : public
```

```

HypothesisTester_if {
00021 public:
00022     HypothesisTesterDefaultImpl1();
00023     virtual ~HypothesisTesterDefaultImpl1() = default;
00024 public:
00025     virtual double testAverage(double confidencelevel, double avg,
H1Comparition comp);
00026     virtual double testProportion(double confidencelevel, double prop,
H1Comparition comp);
00027     virtual double testVariance(double confidencelevel, double var,
H1Comparition comp);
00028     virtual double testAverage(double confidencelevel, std::string secondPopulationDataFilename,
H1Comparition comp);
00029     virtual double testProportion(double confidencelevel, std::string
secondPopulationDataFilename, H1Comparition comp);
00030     virtual double testVariance(double confidencelevel, std::string
secondPopulationDataFilename, H1Comparition comp);
00031     virtual void setDataFilename(std::string dataFilename);
00032     virtual std::string getDataFilename();
00033 private:
00034     Integrator_if* _integrator;
00035 };
00036
00037 #endif /* HYPOTHESISTESTERDEFAULTIMPL1_H */
00038

```

7.199 Integrator_if.h File Reference

Classes

- class [Integrator_if](#)

7.200 Integrator_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Integrator_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 13:54
00012 */
00013
00014 #ifndef INTEGRATOR_IF_H
00015 #define INTEGRATOR_IF_H
00016
00021 class Integrator_if {
00022 public:
00023     virtual void setPrecision(double e) = 0;
00024     virtual double getPrecision() = 0;
00025     virtual double integrate(double min, double max, double (*f)(double, double), double p2) = 0;
00026     virtual double integrate(double min, double max, double (*f)(double, double, double), double
p2, double p3) = 0;
00027     virtual double integrate(double min, double max, double (*f)(double, double, double, double),
double p2, double p3, double p4) = 0;
00028     virtual double integrate(double min, double max, double (*f)(double, double, double, double,
double), double p2, double p3, double p4, double p5) = 0;
00029 };
00030
00031 #endif /* INTEGRATOR_IF_H */
00032

```

7.201 IntegratorDefaultImpl1.cpp File Reference

```
#include "IntegratorDefaultImpl1.h"
#include "Traits.h"
```

7.202 IntegratorDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: IntegratorDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 00:44
00012 */
00013
00014 #include "IntegratorDefaultImpl1.h"
00015 #include "Traits.h"
00016
00018
00019 IntegratorDefaultImpl1::IntegratorDefaultImpl1() {
00020     this->_precision = Traits<Integrator_if>::Precision;
00021 }
00022
00023
00024 void IntegratorDefaultImpl1::setPrecision(double e) {
00025     _precision = e;
00026 }
00027
00028 double IntegratorDefaultImpl1::getPrecision() {
00029     return _precision;
00030 }
00031
00032 double IntegratorDefaultImpl1::integrate(double min, double max, double (*
f)(double, double), double p2) {
00033     //Rosetta::GaussLegendreQuadrature<5>();
00034     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00035     double x, y, h = (max - min) / m;
00036     double I1 = 0.0;
00037     for (unsigned int j = 1; j <= m + 1; j++) {
00038         i = j - 1;
00039         if (i == 0 || i == m)
00040             c = 1;
00041         else
00042             c = 2;
00043         x = min + i*h;
00044         y = f(x, p2);
00045         I1 += c*y;
00046     }
00047     return (h / 2)*I1;
00048 }
00049
00050 double IntegratorDefaultImpl1::integrate(double min, double max, double (*
f)(double, double, double), double p2, double p3) {
00051     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00052     double x, y, h = (max - min) / m;
00053     double I1 = 0.0;
00054     for (unsigned int j = 1; j <= m + 1; j++) {
00055         i = j - 1;
00056         if (i == 0 || i == m)
00057             c = 1;
00058         else
00059             c = 2;
00060         x = min + i*h;
00061         y = f(x, p2, p3);
00062         I1 += c*y;
00063     }
00064     return (h / 2)*I1;
00065 }
00066
00067 double IntegratorDefaultImpl1::integrate(double min, double max, double (*
f)(double, double, double, double), double p2, double p3, double p4) {
00068     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00069     double x, y, h = (max - min) / m;
00070     double I1 = 0.0;
00071     for (unsigned int j = 1; j <= m + 1; j++) {
00072         i = j - 1;
00073         if (i == 0 || i == m)
00074             c = 1;
00075         else
00076             c = 2;
00077         x = min + i*h;
00078         y = f(x, p2, p3, p4);
00079         I1 += c*y;
00080     }
00081     return (h / 2)*I1;
00082 }

```

```

00083
00084 double IntegratorDefaultImpl1::integrate(double min, double max, double (*
00085     f)(double, double, double, double, double), double p2, double p3, double p4, double p5) {
00086     unsigned int c, i, m = Traits<Integrator_if>::MaxIterations;
00087     double x, y, h = (max - min) / m;
00088     double I1 = 0.0;
00089     for (unsigned int j = 1; j <= m + 1; j++) {
00090         i = j - 1;
00091         if (i == 0 || i == m)
00092             c = 1;
00093         else
00094             c = 2;
00095         x = min + i*h;
00096         y = f(x, p2, p3, p4, p5);
00097         I1 += c*y;
00098     }
00099     return (h / 2)*I1;
00100 }
00101 /*
00102 double simpsonIntegral(double a, double b, int n, const std::function<double (double)> &f) {
00103     const double width = (b-a)/n;
00104
00105     double simpson_integral = 0;
00106     for(int step = 0; step < n; step++) {
00107         const double x1 = a + step*width;
00108         const double x2 = a + (step+1)*width;
00109
00110         simpson_integral += (x2-x1)/6.0*(f(x1) + 4.0*f(0.5*(x1+x2)) + f(x2));
00111     }
00112
00113     return simpson_integral;
00114 }
00115 */

```

7.203 IntegratorDefaultImpl1.h File Reference

```
#include "Integrator_if.h"
```

Classes

- class [IntegratorDefaultImpl1](#)

7.204 IntegratorDefaultImpl1.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: IntegratorDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 00:44
00012 */
00013
00014 #ifndef INTEGRATORDEFAULTIMPL1_H
00015 #define INTEGRATORDEFAULTIMPL1_H
00016
00017 #include "Integrator_if.h"
00018
00019 class IntegratorDefaultImpl1 : public Integrator_if {
00020 public:
00021     IntegratorDefaultImpl1();
00022     virtual ~IntegratorDefaultImpl1() = default;
00023 public:
00024     virtual void setPrecision(double e);
00025     virtual double getPrecision();
00026     virtual double integrate(double min, double max, double (*f)(double, double), double p2,
00027                             double p2, double p3);

```

```

00028     virtual double integrate(double min, double max, double (*f)(double, double, double, double),
00029         double p2, double p3, double p4);
00030     virtual double integrate(double min, double max, double (*f)(double, double, double, double,
00031         double), double p2, double p3, double p4, double p5);
00032 private:
00033     double _precision;
00034 #endif /* INTEGRATORDEFAULTIMPL1_H */
00035

```

7.205 Leave.cpp File Reference

```
#include "Leave.h"
#include "Model.h"
```

7.206 Leave.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Leave.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Leave.h"
00015 #include "Model.h"
00016
00017 Leave::Leave(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<Leave>(), name) {
00019
00020
00021 std::string Leave::show() {
00022     return ModelComponent::show() + ",station=" + this->_station-
00023         name();
00024
00025 ModelComponent* Leave::LoadInstance(Model* model,
00026     std::map<std::string, std::string>* fields) {
00027     Leave* newComponent = new Leave(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031     }
00032     return newComponent;
00033 }
00034
00035 void Leave::setStation(Station* _station) {
00036     this->_station = _station;
00037 }
00038
00039 Station* Leave::getStation() const {
00040     return _station;
00041 }
00042
00043 void Leave::_execute(Entity* entity) {
00044     _station->leave(entity);
00045     _parentModel->sendEntityToComponent(entity, this->
00046         nextComponents()->frontConnection(), 0.0);
00047
00048 bool Leave::_loadInstance(std::map<std::string, std::string>* fields) {
00049     bool res = ModelComponent::_loadInstance(fields);
00050     if (res) {
00051         std::string stationName = ((*fields->find("stationName"))).second;
00052         Station* station = dynamic_cast<Station*>(_parentModel->
00053             elements()->element(Util::TypeOf<Station>(), stationName));
00054         this->_station = station;

```

```

00054     }
00055     return res;
00056 }
00057
00058 void Leave::_initBetweenReplications() {
00059 }
00060
00061 std::map<std::string, std::string>* Leave::_saveInstance() {
00062     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00063     ());
00064     fields->emplace("stationName", (this->_station->name()));
00065     return fields;
00066 }
00067
00068 bool Leave::_check(std::string* errorMessage) {
00069     bool resultAll = true;
00070     resultAll &= _parentModel->elements()->check(Util::TypeOf<Station>(),
00071     "Station", errorMessage);
00072     return resultAll;
00073 }
00074
00075 PluginInformation* Leave::GetPluginInformation() {
00076     PluginInformation* info = new PluginInformation(Util::TypeOf<Leave>()
00077     , &Leave::LoadInstance);
00078     info->insertDynamicLibFileDependence("station.so");
00079     return info;
00080 }
00081
00082 }
```

7.207 Leave.h File Reference

```
#include "ModelComponent.h"
#include "Station.h"
```

Classes

- class [Leave](#)

7.208 Leave.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Leave.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef LEAVE_H
00015 #define LEAVE_H
00016
00017 #include "ModelComponent.h"
00018 #include "Station.h"
00019
00093 class Leave: public ModelComponent {
00094 public:
00095     Leave(Model* model, std::string name(""));
00096     virtual ~Leave() = default;
00097 public:
00098     virtual std::string show();
00099 public:
00100     static PluginInformation* GetPluginInformation();
00101     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00102     std::string>* fields);
00103 public:
00104     void setStation(Station* _station);
```

```

00104     Station* getStation() const;
00105 public:
00106 protected:
00107     virtual void _execute(Entity* entity);
00108     virtual void _initBetweenReplications();
00109     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00110     virtual std::map<std::string, std::string>* _saveInstance();
00111     virtual bool _check(std::string* errorMessage);
00112 private: // association
00113     Station* _station;
00114 };
00115
00116 #endif /* LEAVE_H */
00117

```

7.209 LicenceManager.cpp File Reference

```
#include "LicenceManager.h"
```

7.210 LicenceManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: LicenceManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 29 de Maio de 2019, 11:45
00012 */
00013
00014 #include "LicenceManager.h"
00015
00016 LicenceManager::LicenceManager(Simulator* simulator) {
00017     _simulator = simulator;
00018     this->setDefaultLicenceAndLimits();
00019 }
00020
00021
00022 void LicenceManager::setDefaultLicenceAndLimits() {
00023     _licence = "LICENCE: Academic Mode. In academic mode this software has full functionality and executing
00024     training-size simulation models. This software may be duplicated and used for educational purposes only;
00025     any commercial application is a violation of the license agreement. Designed and developed by prof. Dr. Ing
00026     Rafael Luiz Cancian, 2018-2019";
00027     _activationCode = "";
00028     _components = 50;
00029     _elements = 100;
00030     _entities = 200;
00031     _hosts = 1;
00032     _threads = 1;
00033
00034     const std::string LicenceManager::showLicence() const {
00035         return _licence;
00036     }
00037
00038     const std::string LicenceManager::showLimits() const {
00039         std::string msg = "LIMITS: Based on your licence and activation code, your simulator is running under
00040         the following limits" +
00041             std::string(": ") + std::to_string(_components) + " components" +
00042             ", " + std::to_string(_elements) + " elements" +
00043             ", " + std::to_string(_entities) + " entities" +
00044             ", " + std::to_string(_hosts) + " hosts" +
00045             ", " + std::to_string(_threads) + " threads.";
00046     }
00047
00048     const std::string LicenceManager::showActivationCode() const {
00049         std::string msg = "ACTIVATION CODE: Not found.";
00050         return msg;
00051     }
00052
00053     bool LicenceManager::lookforActivationCode() {

```

```

00052     // \todo: Not implemented yet
00053     return false;
00054 }
00055
00056 bool LicenceManager::insertActivationCode() {
00057     // \todo: Not implemented yet
00058     return false;
00059 }
00060
00061 void LicenceManager::removeActivationCode() {
00062     this->setDefaultLicenceAndLimits();
00063 }
00064
00065 unsigned int LicenceManager::modelComponentsLimit() {
00066     return this->_components;
00067 }
00068
00069 unsigned int LicenceManager::modelElementsLimit() {
00070     return this->_elements;
00071 }
00072
00073 unsigned int LicenceManager::entityLimit() {
00074     return this->_entities;
00075 }
00076
00077 unsigned int LicenceManager::hostsLimit() {
00078     return this->_hosts;
00079 }
00080
00081 unsigned int LicenceManager::threadsLimit() {
00082     return this->_threads;
00083 }

```

7.211 LicenceManager.h File Reference

```
#include <string>
```

Classes

- class [LicenceManager](#)

7.212 LicenceManager.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: LicenceManager.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 29 de Maio de 2019, 11:45
00012 */
00013
00014 #ifndef LICENCEMANAGER_H
00015 #define LICENCEMANAGER_H
00016
00017 #include <string>
00018
00019 class Simulator;
00020
00021 class LicenceManager {
00022 public:
00023     LicenceManager(Simulator* simulator);
00024     virtual ~LicenceManager() = default;
00025     const std::string showLicence() const;
00026     const std::string showLimits() const;
00027     const std::string showActivationCode() const;
00028     bool lookforActivationCode();
00029     bool insertActivationCode();
00030     void removeActivationCode();

```

```

00031     unsigned int modelComponentsLimit();
00032     unsigned int modelElementsLimit();
00033     unsigned int entityLimit();
00034     unsigned int hostsLimit();
00035     unsigned int threadsLimit();
00036 private:
00037     void setDefaultLicenceAndLimits();
00038 private:
00039     Simulator* _simulator;
00040     std::string _licence;
00041     std::string _activationCode;
00042     unsigned int _components, _elements, _entities, _hosts, _threads;
00043 };
00044
00045 #endif /* LICENCEMANAGER_H */
00046

```

7.213 LinkedBy.cpp File Reference

```
#include "LinkedBy.h"
```

7.214 LinkedBy.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: LinkedBy.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Agosto de 2018, 07:35
00012 */
00013
00014 #include "LinkedBy.h"
00015
00016 LinkedBy::LinkedBy() {
00017 }
00018
00019 LinkedBy::LinkedBy(const LinkedBy& orig) {
00020 }
00021
00022 LinkedBy::~LinkedBy() {
00023 }
00024
00025 void LinkedBy::addLink() {
00026     this->_linkedBy++;
00027 }
00028
00029 void LinkedBy::removeLink() {
00030     this->_linkedBy--;
00031 }
00032
00033 bool LinkedBy::isLinked() {
00034     return this->_linkedBy > 0;
00035 }

```

7.215 LinkedBy.h File Reference

Classes

- class [LinkedBy](#)

7.216 LinkedBy.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: LinkedBy.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Agosto de 2018, 07:35
00012 */
00013
00014 #ifndef LINKEDBYCOMPONENT_H
00015 #define LINKEDBYCOMPONENT_H
00016
00017 class LinkedBy {
00018 public:
00019     LinkedBy();
00020     LinkedBy(const LinkedBy& orig);
00021     virtual ~LinkedBy();
00022 private:
00023     void addLink();
00024     void removeLink();
00025     bool isLinked();
00026 };
00027
00028 unsigned int _linkedBy = 0;
00029
00030 #endif /* LINKEDBYCOMPONENT_H */
00031

```

7.217 List.h File Reference

```

#include <string>
#include <list>
#include <map>
#include <iterator>
#include <functional>
#include <algorithm>
#include "Util.h"
#include "ModelElement.h"

```

Classes

- class [List< T >](#)

7.218 List.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: TManager.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:55
00012 */
00013
00014 #ifndef LISTMANAGER_H
00015 #define LISTMANAGER_H
00016
00017 #include <string>

```

```

00018 #include <list>
00019 #include <map>
00020 #include <iterator>
00021 #include <functional>
00022 #include <algorithm>
00023 #include "Util.h"
00024 #include "ModelElement.h"
00025
00026 class Simulator;
00027
00031 template <typename T>
00032 class List {
00033 public:
00034     using CompFunct = std::function<bool(const T, const T) >;
00035 public:
00036     List();
00037     virtual ~List() = default;
00038 public: // direct access to list
00039     unsigned int size();
00040     bool empty();
00041     void clear();
00042     void pop_front();
00043     template<class Compare>
00044     void sort(Compare comp);
00045     std::list<T>* list() const;
00046 public: // new methods
00047     T create();
00048     template<typename U>
00049     T create(U arg);
00050     std::string show();
00051     typename std::list<T>::iterator find(T element);
00052     //int rankOf(T element); ///* _list;
00067     CompFunct _sortFunc;[](const T, const T) {
00068         return false;
00069     } };
00070     typename std::list<T>::iterator _it;
00071 };
00072
00073 template <typename T>
00074 List<T>::List() {
00075     //_map = new std::map<Util::identitifcation, T*>();
00076     _list = new std::list<T>();
00077     _it = _list->begin();
00078 }
00079
00080 template <typename T>
00081 std::list<T>* List<T>::list() const {
00082     return _list;
00083 }
00084
00085 template <typename T>
00086 unsigned int List<T>::size() {
00087     return _list->size();
00088 }
00089
00090 //template <typename T>
00091 //List<T>::List(const List& orig) {
00092 //}
00093
00094 //template <typename T>
00095 //List<T>::~List() {
00096 //}
00097
00098 template <typename T>
00099 std::string List<T>::show() {
00100     int i = 0;
00101     std::string text = "{";
00102     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++, i++) {
00103         text += "[" + std::to_string(i) + "]=" + (*it)->show() + ", ";
00104     }
00105     text += "}";
00106     return text;

```

```
00107 }
00108
00109 template <typename T>
00110 void List<T>::insert(T element) {
00111     _list->insert(std::upper_bound(_list->begin(), _list->end(), element, _sortFunc), element);
00112 }
00113
00114 template <typename T>
00115 bool List<T>::empty() {
00116     return _list->empty();
00117 }
00118
00119 template <typename T>
00120 void List<T>::pop_front() {
00121     typename std::list<T>::iterator itTemp = _list->begin();
00122     _list->pop_front();
00123     if (_it == itTemp) { /* \todo: +: check this */
00124         _it = _list->begin(); // if it points to the removed element, then changes to begin
00125     }
00126 }
00127
00128 template <typename T>
00129 void List<T>::remove(T element) {
00130     _list->remove(element);
00131     if ((*_it) == element) { /* \todo: +: check this */
00132         _it = _list->begin(); // if it points to the removed element, then changes to begin
00133     }
00134 }
00135
00136 template <typename T>
00137 T List<T>::create() {
00138     return new T();
00139 }
00140
00141 template <typename T>
00142 void List<T>::clear() {
00143     _list->clear();
00144 }
00145
00146 template <typename T>
00147 T List<T>::getAtRank(unsigned int rank) {
00148     unsigned int thisRank = 0;
00149     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00150         if (rank == thisRank) {
00151             return(*it);
00152         } else {
00153             thisRank++;
00154         }
00155     }
00156     return 0; /* \todo: Invalid return depends on T. If T is pointer, nullptr works fine. If T is double,
00157     it does not. I just let (*it), buut it is not nice*/
00158 }
00159
00160 template <typename T>
00161 void List<T>::setAtRank(unsigned int rank, T element) {
00162     unsigned int thisRank = 0;
00163     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00164         if (rank == thisRank) {
00165             *it = element;
00166         } else {
00167             thisRank++;
00168         }
00169     }
00170 }
00171
00172 template <typename T>
00173 T List<T>::next() {
00174     _it++;
00175     if (_it != _list->end())
00176         return(*_it);
00177     else
00178         return nullptr;
00179 }
00180
00181
00182 template <typename T>
00183 typename std::list<T>::iterator List<T>::find(T element) {
00184     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00185         if ((*it) == element) {
00186             return it;
00187         }
00188     }
00189     return _list->end(); /* \todo:+: check nullptr or invalid iterator when not found */
00190     //return nullptr;
00191 }
```

```

00193 /*
00194 template <typename T>
00195 int List<T>::rankOf(T element) {
00196     int rank = 0;
00197     for (typename std::list<T>::iterator it = _list->begin(); it != _list->end(); it++) {
00198         if ((*it) == element) {
00199             return rank;
00200         } else
00201             rank++;
00202     }
00203     return -1; // not found -> negative rank
00204 }
00205 */
00206
00207 template <typename T>
00208 T List<T>::front() {
00209     _it = _list->begin();
00210     //if (_it != _list->end())
00211     return(*_it);
00212     //else
00213     //return dynamic_cast<T>(nullptr);
00214 }
00215
00216 template <typename T>
00217 T List<T>::last() {
00218     _it = _list->end();
00219     _it--;
00220     //if (_it != _list->end()) // \todo: CHECK!!!
00221     return(*_it);
00222     //else return nullptr;
00223 }
00224
00225 template <typename T>
00226 T List<T>::previous() {
00227     _it--; // \todo: CHECK!!!
00228     return(*_it);
00229 }
00230
00231 template <typename T>
00232 T List<T>::current() {
00233     /* \todo: To implement (i think it's just to check). Must actualize _it on other methods when other
        elements are accessed */
00234     return(*_it);
00235 }
00236
00237 template <typename T>
00238 void List<T>::setSortFunc(CompFunct _sortFunc) {
00239     this->_sortFunc = _sortFunc;
00240 }
00241
00242 template <typename T>
00243 template<typename U>
00244 T List<T>::create(U arg) {
00245     return T(arg);
00246 }
00247
00248 template <typename T>
00249 template<class Compare>
00250 void List<T>::sort(Compare comp) {
00251     _list->sort(comp);
00252 }
00253
00254 #endif /* LISTMANAGER_H */
00255

```

7.219 LSODE.cpp File Reference

```
#include "LSODE.h"
#include "Model.h"
```

7.220 LSODE.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
```

```
00005  */
00006
00007 /*
00008 * File: LSODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 22 de Outubro de 2019, 22:28
00012 */
00013
00014 #include "LSODE.h"
00015 #include "Model.h"
00016
00017 LSODE::LSODE(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<LSODE>(), name) {
00019
00020     std::string LSODE::show() {
00021         return ModelComponent::show() + "";
00022     }
00023
00024     ModelComponent* LSODE::LoadInstance(Model* model,
00025         std::map<std::string, std::string>* fields) {
00026         LSODE* newComponent = new LSODE(model);
00027         try {
00028             newComponent->_loadInstance(fields);
00029         } catch (const std::exception& e) {
00030
00031         return newComponent;
00032     }
00033
00034     void LSODE::setDiffEquations(Formula* formula) {
00035         _diffEquations = formula;
00036     }
00037
00038     Formula* LSODE::getDiffEquations() const {
00039         return _diffEquations;
00040     }
00041
00042     void LSODE:: setTimeVariable(Variable* timeVariable) {
00043         _timeVariable = timeVariable;
00044     }
00045
00046     Variable* LSODE::getTimeVariable() const {
00047         return _timeVariable;
00048     }
00049
00050     void LSODE::setStep(double step) {
00051         _step = step;
00052     }
00053
00054     double LSODE::getStep() const {
00055         return _step;
00056     }
00057
00058     void LSODE::setVariables(Variable* variables) {
00059         _variables = variables;
00060     }
00061
00062     Variable* LSODE::getVariables() const {
00063         return _variables;
00064     }
00065
00066     bool LSODE::_doStep() {
00067         double initTime, time, tnow, eqResult, halfStep;
00068         //std::list<std::string>* eqs = _diffEquations->formulaExpressions()->list();
00069         unsigned int i, numEqs = _diffEquations->size();
00070         double k1[numEqs], k2[numEqs], k3[numEqs], k4[numEqs], valVar[numEqs];
00071         time = _timeVariable->value();
00072         initTime = time;
00073         tnow = _parentModel->simulation()->simulatedTime();
00074         bool res = time + _step <= tnow + 1e-15; // \todo: numerical error treatment by just adding 1e-15
00075         if (res) {
00076             halfStep = _step * 0.5;
00077             for (i = 0; i < numEqs; i++) { // (std::list<std::string>::iterator it = eqs->begin(); it != eqs->end(); it++)
00078                 std::string expression = _diffEquations->expression(std::to_string(i));
00079                 valVar[i] = _variables->value(std::to_string(i));
00080                 eqResult = _parentModel->parseExpression(expression);
00081                 k1[i++] = eqResult;
00082             }
00083             time += halfStep;
00084             _timeVariable->setValue(time);
00085             for (i = 0; i < numEqs; i++) {
00086                 _variables->setValue(std::to_string(i), valVar[i] + k1[i] * halfStep);
00087             }
00088             for (i = 0; i < numEqs; i++) {
```

```

00089     eqResult = _parentModel->parseExpression(_diffEquations->
00090         expression(std::to_string(i)));
00091     k2[i] = eqResult;
00092 }
00093     for (i = 0; i < numEqs; i++) {
00094         _variables->setValue(std::to_string(i), valVar[i] + k2[i] * halfStep);
00095     }
00096     for (i = 0; i < numEqs; i++) {
00097         eqResult = _parentModel->parseExpression(_diffEquations->
00098             expression(std::to_string(i)));
00099         k3[i] = eqResult;
00100     }
00101     for (i = 0; i < numEqs; i++) {
00102         _variables->setValue(std::to_string(i), valVar[i] + k3[i] * halfStep);
00103     }
00104     for (i = 0; i < numEqs; i++) {
00105         eqResult = _parentModel->parseExpression(_diffEquations->
00106             expression(std::to_string(i)));
00107         k4[i] = eqResult;
00108     }
00109     _variables->setValue(std::to_string(i), eqResult);
00110 time = initTime + _step;
00111 _timeVariable->setValue(time);
00112 }
00113 return res;
00114 }
00115
00116 void LSODE::_execute(Entity* entity) {
00117     // _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00118     //for (std::list<std::string>::iterator it = _diffEquations->getFormulaExpressions()->list()->begin();
00119     it != _diffEquations->getFormulaExpressions()->list()->end(); it++) {
00120         //double value = _parentModel->parseExpression((*it));
00121         //_parentModel->tracer()->trace("Expression \"" + (*it) + "\" evaluates to " + std::to_string(value));
00122         //}
00123         while (_doStep()) { // execute solve ODE step by step until reach TNOW
00124             std::string message = "time=" + std::to_string(_timeVariable->value());
00125             for (unsigned int i = 0; i < _variables->dimensionSizes()->
00126                 front(); i++) {
00127                 message += " ,y[" + std::to_string(i) + "]=" + std::to_string(_variables->
00128                     value(std::to_string(i)));
00129             }
00130             _parentModel->tracer()->trace(message);
00131             _parentModel->sendEntityToComponent(entity,
00132                 nextComponents()->frontConnection(), 0.0);
00133         }
00134     }
00135     bool res = ModelComponent::_loadInstance(fields);
00136     if (res) {
00137         ...
00138     }
00139 }
00140 void LSODE::_initBetweenReplications() {
00141 }
00142
00143 std::map<std::string, std::string>* LSODE::_saveInstance() {
00144     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00145     ();
00146     ...
00147     return fields;
00148 }
00149 bool LSODE::_check(std::string* errorMessage) {
00150     bool resultAll = true;
00151     ...
00152     return resultAll;
00153 }
00154
00155 PluginInformation* LSODE::GetPluginInformation() {
00156     PluginInformation* info = new PluginInformation(Util::TypeOf<LSODE>()
00157     , &LSODE::LoadInstance);
00158     ...
00159     return info;
00160 }
```

7.221 LSODE.h File Reference

```
#include "ModelComponent.h"
#include "Formula.h"
#include "Variable.h"
```

Classes

- class [LSODE](#)

7.222 LSODE.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: LSODE.h
00009  * Author: rlcancian
00010 *
00011  * Created on 22 de Outubro de 2019, 22:28
00012 */
00013
00014 #ifndef LSODE_H
00015 #define LSODE_H
00016
00017 #include "ModelComponent.h"
00018 #include "Formula.h"
00019 #include "Variable.h"
00020
00021 class LSODE : public ModelComponent {
00022 public: // constructors
00023     LSODE(Model* model, std::string name = "");
00024     virtual ~LSODE() = default;
00025 public: // virtual
00026     virtual std::string show();
00027 public: // static
00028     static PluginInformation* GetPluginInformation();
00029     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00030                                         std::string>* fields);
00031 public: // g&s
00032     void setDiffEquations(Formula* formula);
00033     Formula* getDiffEquations() const;
00034     void setTimeVariable(Variable* _timeVariable);
00035     Variable* getTimeVariable() const;
00036     void setStep(double _step);
00037     double getStep() const;
00038     void setVariables(Variable* _variables);
00039     Variable* getVariables() const;
00040 protected: // virtual
00041     virtual void _execute(Entity* entity);
00042     virtual void _initBetweenReplications();
00043     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00044     virtual std::map<std::string, std::string>* _saveInstance();
00045     virtual bool _check(std::string* errorMessage);
00046     //virtual void _createInternalElements();
00047 private: // methods
00048     bool _doStep();
00049 private: // attributes 1:1
00050     Formula* _diffEquations;
00051     Variable* _variables;
00052     Variable* _timeVariable;
00053     double _step;
00054 private: // attributes 1:n
00055 };
00056
00057 */
00058 #endif /* LSODE_H */
00059
00060
```

7.223 main.cpp File Reference

```
#include <iostream>
#include <thread>
#include "Traits.h"
```

Functions

- int **main** (int argc, char **argv)

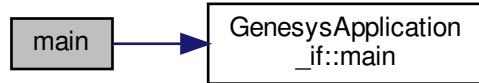
7.223.1 Function Documentation

7.223.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

Definition at line 21 of file [main.cpp](#).

Here is the call graph for this function:



7.224 main.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: main.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:47
00012 */
00013
00014 #include <iostream>
00015 #include <thread>
00016 #include "Traits.h"
00017
00018 /*
00019 * This is the MAIN application of GenESyS. It just calls the Application specified on the configuration
file.
00020 */
```

```

00021 int main(int argc, char** argv) {
00022     // do not change it. Set your own application in Traits file =>
00023     Traits<GenesysApplication_if>::Application
00024     GenesysApplication_if *app = new
00025     Traits<GenesysApplication_if>::Application();
00026     int res = app->main(argc, argv);
00027     // that's all folks!
00028     std::cout << "Quiting application." << std::endl;
00029     for (unsigned int i=0; i<1e6; i++)
00030     std::this_thread::yield(); // Give the IDE a try to output previous traces
00031     return res;
00030 }
00031

```

7.225 MarkovChain.cpp File Reference

```

#include "MarkovChain.h"
#include "Model.h"
#include "Variable.h"
#include "ProbDistrib.h"
#include "Simulator.h"

```

7.226 MarkovChain.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    MarkovChain.cpp
00009  * Author:  rlcancian
00010 *
00011 * Created on 24 de Outubro de 2019, 18:26
00012 */
00013
00014 #include "MarkovChain.h"
00015
00016 #include "Model.h"
00017 #include "Variable.h"
00018 #include "ProbDistrib.h"
00019 #include "Simulator.h"
00020
00021 MarkovChain::MarkovChain(Model* model, std::string name) :
00022     ModelComponent(model, Util::TypeOf<MarkovChain>(), name) {
00023 }
00024 std::string MarkovChain::show() {
00025     return ModelComponent::show() + "";
00026 }
00027
00028 ModelComponent* MarkovChain::LoadInstance(
00029     Model* model, std::map<std::string, std::string>* fields) {
00030     try {
00031         newComponent->_loadInstance(fields);
00032     } catch (const std::exception& e) {
00033     }
00034     return newComponent;
00035 }
00036
00037
00038 void MarkovChain::setTransitionProbabilityMatrix(
00039     Variable* _transitionMatrix) {
00040     this->_transitionProbMatrix = _transitionMatrix;
00041 }
00042 Variable* MarkovChain::getTransitionMatrix() const {
00043     return _transitionProbMatrix;
00044 }
00045
00046 Variable* MarkovChain::getCurrentState() const {
00047     return _currentState;

```

```

00048 }
00049
00050 void MarkovChain::setCurrentState(Variable* _currentState) {
00051     this->_currentState = _currentState;
00052 }
00053
00054 void MarkovChain::setInitialDistribution(
00055     Variable* _initialDistribution) {
00056     this->_initialDistribution = _initialDistribution;
00057 }
00058 Variable* MarkovChain::getInitialState() const {
00059     return _initialDistribution;
00060 }
00061
00062 void MarkovChain::setInitialized(bool _initialized) {
00063     this->_initialized = _initialized;
00064 }
00065
00066 bool MarkovChain::isInitialized() const {
00067     return _initialized;
00068 }
00069
00070 void MarkovChain::_execute(Entity* entity) {
00071     //parentModel->tracer()->trace("I'm just a dummy model and I'll just send the entity forward");
00072     unsigned int size;
00073     double rnd, sum, value;
00074     if (!_initialized) {
00075         // define the initial state based on initial probabilities
00076         size = _initialDistribution->dimensionSizes()->front();
00077         rnd = _parentModel->parentSimulator()->tools()->
00078             sampler()->random();
00079         sum = 0.0;
00080         for (unsigned int i = 0; i < size; i++) {
00081             value = _initialDistribution->value(std::to_string(i));
00082             sum += value;
00083             if (sum > rnd) {
00084                 _currentState->setValue(i); // _currentState = i;
00085                 break;
00086             }
00087             _parentModel->tracer()->trace("Initial current state=" + std::to_string(
00088                 _currentState->value()));
00089             _initialized = true;
00090         } else {
00091             size = _transitionProbMatrix->dimensionSizes()->front();
00092             rnd = _parentModel->parentSimulator()->tools()->
00093                 sampler()->random();
00094             sum = 0.0;
00095             for (unsigned int i = 0; i < size; i++) {
00096                 std::string index = std::to_string(static_cast<unsigned int>(_currentState->
00097                     value()) + "," + std::to_string(i));
00098                 value = _transitionProbMatrix->value(index);
00099                 sum += value;
00100                 if (sum > rnd) {
00101                     _currentState->setValue(i);
00102                     break;
00103                 }
00104             _parentModel->tracer()->trace("Current state=" + std::to_string(_currentState->
00105                 value()));
00106         }
00107     bool MarkovChain::_loadInstance(std::map<std::string, std::string>* fields) {
00108         bool res = ModelComponent::_loadInstance(fields);
00109         if (res) {
00110             //...
00111         }
00112         return res;
00113     }
00114
00115 void MarkovChain::_initBetweenReplications() {
00116     this->_initialized = false;
00117 }
00118
00119 std::map<std::string, std::string>* MarkovChain::_saveInstance() {
00120     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00121     ();
00122     //...
00123     return fields;
00124 }
00125 bool MarkovChain::_check(std::string* errorMessage) {
00126     bool resultAll = true;

```

```

00127     //...
00128     return resultAll;
00129 }
00130
00131 PluginInformation* MarkovChain::GetPluginInformation() {
00132     PluginInformation* info = new PluginInformation(
00133         Util::TypeOf<MarkovChain>(), &MarkovChain::LoadInstance);
00134     // ...
00135     return info;
00136 }
```

7.227 MarkovChain.h File Reference

```
#include "ModelComponent.h"
#include "Variable.h"
```

Classes

- class [MarkovChain](#)

7.228 MarkovChain.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    MarkovChain.h
00009  * Author:  rlcancian
00010 *
00011  * Created on 24 de Outubro de 2019, 18:26
00012 */
00013
00014 #ifndef MARKOVCHAIN_H
00015 #define MARKOVCHAIN_H
00016
00017 #include "ModelComponent.h"
00018 #include "Variable.h"
00019
00020 class MarkovChain : public ModelComponent {
00021 public: // constructors
00022     MarkovChain(Model* model, std::string name = "");
00023     virtual ~MarkovChain() = default;
00024 public: // virtual
00025     virtual std::string show();
00026 public: // static
00027     static PluginInformation* GetPluginInformation();
00028     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00029                                         std::string>* fields);
00030 public: // get and set
00031     void setTransitionProbabilityMatrix(Variable* _transitionMatrix);
00032     Variable* getTransitionMatrix() const;
00033     Variable* getCurrentState() const;
00034     void setInitialDistribution(Variable* _initialDistribution);
00035     Variable* getInitialState() const;
00036     void setInitialized(bool _initialized);
00037     bool isInitialized() const;
00038 protected: // virtual
00039     virtual void _execute(Entity* entity);
00040     virtual void _initBetweenReplications();
00041     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00042     virtual std::map<std::string, std::string>* _saveInstance();
00043     virtual bool _check(std::string* errorMessage);
00044 private: // methods
00045 private: // attributes 1:1
00046     Variable* _transitionProbMatrix;
00047     Variable* _initialDistribution;
00048     Variable* _currentState;
00049     bool _initialized = false;
00050 private: // attributes 1:n
00051 };
00052
00053 #endif /* MARKOVCHAIN_H */
```

7.229 Match.cpp File Reference

```
#include "Match.h"
#include "Model.h"
```

7.230 Match.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Match.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Match.h"
00015
00016 #include "Model.h"
00017
00018 Match::Match(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Match>(), name) {
00020
00021
00022 std::string Match::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Match::_loadInstance(Model* model,
00027     std::map<std::string, std::string>* fields) {
00028     Match* newComponent = new Match(model);
00029     try {
00030         newComponent->_loadInstance(fields);
00031     } catch (const std::exception& e) {
00032     }
00033     return newComponent;
00034 }
00035
00036 void Match::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038         entity forward");
00039     this->_parentModel->sendEntityToComponent(entity, this->
00040         nextComponents()->frontConnection(), 0.0);
00041 }
00042 bool Match::_loadInstance(std::map<std::string, std::string>* fields) {
00043     bool res = ModelComponent::_loadInstance(fields);
00044     if (res) {
00045         //...
00046     }
00047     return res;
00048 }
00049 void Match::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* Match::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00054     ();
00055     //...
00056     return fields;
00057 }
00058 bool Match::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Match::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Match>()
00066     , &Match::LoadInstance);
00067     // ...
00068     return info;
00069 }
```

7.231 Match.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Match](#)

7.232 Match.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Match.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef MATCH_H
00015 #define MATCH_H
00016
00017 #include "ModelComponent.h"
00018
00047 class Match : public ModelComponent {
00048 public: // constructors
00049     Match(Model* model, std::string name(""));
00050     virtual ~Match() = default;
00051 public: // virtual
00052     virtual std::string show();
00053 public: // static
00054     static PluginInformation* GetPluginInformation();
00055     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00056                                         std::string>* fields);
00056 protected: // virtual
00057     virtual void _execute(Entity* entity);
00058     virtual void _initBetweenReplications();
00059     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00060     virtual std::map<std::string, std::string>* _saveInstance();
00061     virtual bool _check(std::string* errorMessage);
00062 private: // methods
00063 private: // attributes 1:1
00064 private: // attributes 1:n
00065 };
00066
00067
00068 #endif /* MATCH_H */
```

7.233 MathMeth.cpp File Reference

```
#include "MathMeth.h"
```

7.234 MathMeth.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: MathMeth.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 17 de Outubro de 2019, 17:46
00012 */
00013
00014 #include "MathMeth.h"
00015
00016 MathMeth::MathMeth() {
00017 }
00018
00019 /*
00020 void MathMeth::RK4step(int ndep, double dx, double &x, valarray<double> &Y, valarray<double> (*F)(double,
00021 valarray<double>)) {
00022     valarray<double> dY1(ndep), dY2(ndep), dY3(ndep), dY4(ndep);
00023
00024     dY1 = F(x, Y) * dx;
00025     dY2 = F(x + 0.5 * dx, Y + 0.5 * dY1) * dx;
00026     dY3 = F(x + 0.5 * dx, Y + 0.5 * dY2) * dx;
00027     dY4 = F(x + dx, Y + dY3) * dx;
00028     Y += (dY1 + 2.0 * dY2 + 2.0 * dY3 + dY4) / 6.0;
00029
00030     x += dx;
00031 }

```

7.235 MathMeth.h File Reference

```
#include <valarray>
```

Classes

- class [MathMeth](#)

7.236 MathMeth.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: MathMeth.h
00009 * Author: rlcancian
00010 *
00011 * Created on 17 de Outubro de 2019, 17:46
00012 */
00013
00014 #ifndef MATHMETH_H
00015 #define MATHMETH_H
00016
00017 #include <valarray>
00018
00019 class MathMeth {
00020 public:
00021     MathMeth();
00022 public:
00023     //void RK4step(int ndep, double dx, double &x, valarray<double> &Y, valarray<double> (*F)(double,
00024     valarray<double>));
00025 private:
00026 };
00027
00028 #endif /* MATHMETH_H */
00029

```

7.237 Model.cpp File Reference

```
#include <typeinfo>
#include <iostream>
#include <algorithm>
#include <string>
#include "Model.h"
#include "SourceModelComponent.h"
#include "Simulator.h"
#include "StatisticsCollector.h"
#include "Traits.h"
#include "Access.h"
```

Functions

- bool [EventCompare](#) (const [Event](#) *a, const [Event](#) *b)

7.237.1 Function Documentation

7.237.1.1 [EventCompare\(\)](#)

```
bool EventCompare (
    const Event * a,
    const Event * b )
```

Definition at line [26](#) of file [Model.cpp](#).

Here is the call graph for this function:



7.238 Model.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SimulationModel.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 15:01
00012 */
```

```

00013 #include <typeinfo>
00014 #include <iostream>
00015 #include <algorithm>
00016 #include <string>
00018
00019 #include "Model.h"
00020 #include "SourceModelComponent.h"
00021 #include "Simulator.h"
00022 #include "StatisticsCollector.h"
00023 #include "Traits.h"
00024 #include "Access.h"
00025
00026 bool EventCompare(const Event* a, const Event * b) {
00027     return a->time() < b->time();
00028 }
00029
00030 Model::Model(Simulator* simulator) {
00031     _parentSimulator = simulator; // a simulator is the "parent" of a model
00032     // 1:1 associations (no Traits)
00033     _modelInfo = new ModelInfo();
00034     _eventManager = new OnEventManager(); // should be on .h (all that does not depends on
THIS)
00035     _elementManager = new ElementManager(this);
00036     _componentManager = new ComponentManager(this);
00037     _traceManager = simulator->tracer(); // every model starts with the same tracer, unless a
specific one is set
00038     // 1:1 associations (Traits)
00039     _parser = new Traits<Parser_if>::Implementation(this);
00040     _modelChecker = new Traits<ModelChecker_if>::Implementation(this
);
00041     _modelPersistence = new Traits<ModelPersistence_if>::Implementation
(this);
00042     _simulation = new ModelSimulation(this);
00043     // 1:n associations
00044     _events = new List<Event*>();
00045     //_events->setSortFunc(&EventCompare); // It works too
00046     _events->setSortFunc([](const Event* a, const Event * b) {
00047         return a->time() < b->time();
00048     });
00049     // for process analyser
00050     _responses = new List<SimulationResponse*>();
00051     _controls = new List<SimulationControl*>();
00052     // insert controls
00053     _controls->insert(new SimulationControl("Model Info", "Number of Replications",
00054         DefineGetterMember<ModelInfo>(this->_modelInfo, &
ModelInfo::numberOfReplications),
00055         DefineSetterMember<ModelInfo>(this->_modelInfo, &
ModelInfo::setNumberOfReplications)
00056     );
00057     _controls->insert(new SimulationControl("Model Info", "Replication Length",
00058         DefineGetterMember<ModelInfo>(this->_modelInfo, &ModelInfo::replicationLength),
00059         DefineSetterMember<ModelInfo>(this->_modelInfo, &
ModelInfo::setReplicationLength))
00060     );
00061     _controls->insert(new SimulationControl("Model Info", "Warmup Period",
00062         DefineGetterMember<ModelInfo>(this->_modelInfo, &ModelInfo::warmUpPeriod),
00063         DefineSetterMember<ModelInfo>(this->_modelInfo, &
ModelInfo::setWarmUpPeriod))
00064     );
00065 }
00066
00067 void Model::sendEntityToComponent(Entity* entity,
Connection* connection, double timeDelay) {
00068     this->sendEntityToComponent(entity, connection->first, timeDelay, connection->
second);
00069 }
00070
00071 void Model::sendEntityToComponent(Entity* entity,
ModelComponent* component, double timeDelay, unsigned int componentInputNumber) {
00072     this->onEvents()->NotifyEntityMoveHandlers(new
SimulationEvent(_simulation->currentReplicationNumber(), new
Event(_simulation->simulatedTime(), entity, component, componentInputNumber))); //\todo:
Event should include information about "from component" and timeDelay, but it doesn't
00073     if (timeDelay > 0) {
00074         // schedule to send it
00075         Event* newEvent = new Event(this->simulation()->simulatedTime() + timeDelay, entity
, component, componentInputNumber);
00076         this->futureEvents()->insert(newEvent);
00077     } else {
00078         // send it now
00079         /* \todo: :- supposed not to be a queue associated to a component */
00080         Util::DecIndent();
00081         ModelComponent::Execute(entity, component, componentInputNumber);
00082         Util::IncIndent();
00083     }

```

```
00084 }
00085
00086 bool Model::save(std::string filename) {
00087     bool res = this->_modelPersistence->save(filename);
00088     if (res) {
00089         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model
00090             successfully saved");
00091     } else {
00092         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model
00093             could not be saved");
00094     }
00095 }
00096
00097 bool Model::load(std::string filename) {
00098     this->clear();
00099     bool res = this->_modelPersistence->load(filename);
00100     if (res)
00101         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model
00102             successfully loaded");
00103     else
00104         this->_traceManager->trace(Util::TraceLevel::modelResult, "Model
00105             could not be loaded");
00106     return res;
00107 }
00108 double Model::parseExpression(const std::string expression) {
00109     try {
00110         return _parser->parse(expression);
00111     } catch (...) {
00112         return 0.0; // \todo: HOW SAY THERE WAS AN ERROR?
00113     }
00114
00115 bool Model::checkExpression(const std::string expression, const std::string
00116     expressionName, std::string* errorMessage) {
00117     bool result;
00118     parseExpression(expression, &result, errorMessage);
00119     if (!result) {
00120         std::string msg = "Expression \"\" + expression + "\" for '" + expressionName + "' is incorrect. ";
00121         errorMessage->append(msg);
00122     }
00123     return result;
00124 }
00125 double Model::parseExpression(const std::string expression, bool* success,
00126     std::string* errorMessage) {
00127     double value = _parser->parse(expression, success, errorMessage);
00128     return value;
00129 }
00130 void Model::show() {
00131     tracer()->trace(Util::TraceLevel::report, "Simulation Model:");
00132     Util::IncIndent();
00133     {
00134         tracer()->trace(Util::TraceLevel::report, "Information:");
00135         Util::IncIndent();
00136         tracer()->trace(Util::TraceLevel::report, this->
00137             infos()->show());
00138         Util::DecIndent();
00139         _showComponents();
00140         _showElements();
00141         _showSimulationControls();
00142         _showSimulationResponses();
00143     }
00144     Util::DecIndent();
00145     tracer()->trace(Util::TraceLevel::report, "End of Simulation Model")
00146 ;
00147 }
00148 bool Model::insert(ModelElement* elemOrComp) {
00149     ModelComponent* comp = dynamic_cast<ModelComponent*> (elemOrComp);
00150     if (comp == nullptr) // it's a ModelElement
00151         return this->elements()->insert(elemOrComp);
00152     else // it's a ModelComponent
00153         return this->components()->insert(comp);
00154 }
00155 void Model::remove(ModelElement* elemOrComp) {
00156     ModelComponent* comp = dynamic_cast<ModelComponent*> (elemOrComp);
00157     if (comp == nullptr) // it's a ModelElement
00158         this->elements()->remove(elemOrComp);
00159     else // it's a ModelComponent
00160         this->components()->remove(comp);
00161 }
00162 }
```

```

00163 void Model::_showElements() const {
00164     tracer()->trace(Util::TraceLevel::report, "Elements:");
00165     Util::IncIndent();
00166     {
00167         std::string elementType;
00168         ModelElement* element;
00169         std::list<std::string>* elementTypes = elements()->elementClassnames();
00170         for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end(); typeIt++) {
00171             elementType = (*typeIt);
00172             List<ModelElement*>* em = elements()->
00173                 elementList(elementType);
00174             tracer()->trace(Util::TraceLevel::report, elementType + ":");
00175             Util::IncIndent();
00176             {
00177                 for (std::list<ModelElement*>::iterator it = em->list()->begin(); it != em->
00178                     list()->end(); it++) {
00179                     element = (*it);
00180                     tracer()->trace(Util::TraceLevel::report, element->
00181                         show());
00182                 }
00183             }
00184             Util::DecIndent();
00185         }
00186     }
00187 void Model::_showComponents() const {
00188     tracer()->trace(Util::TraceLevel::report, "Components:");
00189     Util::IncIndent();
00190     for (std::list<ModelComponent*>::iterator it = components()->begin(); it != components()->end(); it++) {
00191         tracer()->trace(Util::TraceLevel::report, (*it)->show());
00192     }
00193     Util::DecIndent();
00194 }
00195
00196 void Model::_showSimulationControls() const {
00197     tracer()->trace(Util::TraceLevel::report, "Simulation Controls:");
00198     Util::IncIndent();
00199     for (std::list<SimulationControl*>::iterator it = _controls->list()->begin(); it != _controls->
00200         list()->end(); it++) {
00201         tracer()->trace(Util::TraceLevel::report, (*it)->show());
00202     }
00203     Util::DecIndent();
00204 }
00205 void Model::_showSimulationResponses() const {
00206     tracer()->trace(Util::TraceLevel::report, "Simulation Responses:");
00207     Util::IncIndent();
00208     for (std::list<SimulationResponse*>::iterator it = _responses->list()->begin(); it != _responses->
00209         list()->end(); it++) {
00210         tracer()->trace(Util::TraceLevel::report, (*it)->show());
00211     }
00212     Util::DecIndent();
00213 }
00214 void Model::clear() {
00215     this->_componentManager->clear();
00216     this->_elementManager->clear();
00217     //Util::ResetAllIds(); // \todo: To implement
00218 }
00219
00220 void Model::_createModelInternalElements() {
00221     for (std::list<ModelComponent*>::iterator it = _componentManager->begin(); it != _componentManager-
00222         >end(); it++) {
00223         ModelComponent::CreateInternalElements((*it));
00224     }
00225     std::list<ModelElement*>* modelElements;
00226     for (std::list<std::string>::iterator itty = elements()->getElementTypenames()->begin(); itty != el-
00227         ements()->getElementTypenames()->end(); itty++) {
00228         modelElements = elements()->getElements((itty))->list();
00229         for (std::list<ModelElement*>::iterator itel = modelElements->begin(); itel != modelElements->end();
00230             itel++) {
00231             ModelElement::CreateInternalElements((*itel));
00232         }
00233     }
00234 }
00235 bool Model::check() {
00236     tracer()->trace(Util::TraceLevel::modelInternal, "Checking
00237         model consistency");
00238     Util::IncIndent();
00239     // before checking the model, creates all necessary internal ModelElements

```

```
00239     _createModelInternalElements();
00240     bool res = this->_modelChecker->checkAll();
00241     Util::DecIndent();
00242     if (res) {
00243         tracer()->trace(Util::TraceLevel::modelResult, "End of Model
00244             checking: Success");
00245     } else {
00246         //std::exception e = new std::exception();
00247         //getTrace()->traceError();
00248         tracer()->trace(Util::TraceLevel::modelResult, "End of Model
00249             checking: Failed");
00250     }
00251     return res;
00252 //bool Model::verifySymbol(std::string componentName, std::string expressionName, std::string expression,
00253 //    std::string expressionResult, bool mandatory) {
00254 //    return this->_modelChecker->verifySymbol(componentName, expressionName, expression, expressionResult,
00255 //        mandatory);
00256 //}
00257 void Model::removeEntity(Entity* entity, bool collectStatistics) {
00258     /* todo: -: event onEntityRemove */
00259     std::string entId = std::to_string(entity->entityNumber());
00260     this->elements()->remove(Util::TypeOf<Entity>(), entity);
00261     tracer()->trace("Entity " + entId + " was removed from the system");
00262 }
00263 List<Event*>* Model::futureEvents() const {
00264     return _events;
00265 }
00266
00267 void Model::setTraceManager(TraceManager * _traceManager) {
00268     this->_traceManager = _traceManager;
00269 }
00270
00271 TraceManager * Model::tracer() const {
00272     return _traceManager;
00273 }
00274
00275 bool Model::hasChanged() const {
00276     bool changed = _hasChanged;
00277     changed &= this->_componentManager->hasChanged();
00278     changed &= this->_elementManager->hasChanged();
00279     changed &= this->_modelInfo->hasChanged();
00280     changed &= this->_modelPersistence->hasChanged();
00281     return changed;
00282 }
00283
00284 ComponentManager * Model::components() const {
00285     return _componentManager;
00286 }
00287
00288 List<SimulationControl*>* Model::controls() const {
00289     return _controls;
00290 }
00291
00292 List<SimulationResponse*>* Model::responses() const {
00293     return _responses;
00294 }
00295
00296 OnEventManager * Model::onEvents() const {
00297     return _eventManager;
00298 }
00299
00300 ElementManager * Model::elements() const {
00301     return _elementManager;
00302 }
00303
00304 ModelInfo * Model::infos() const {
00305     return _modelInfo;
00306 }
00307
00308 Simulator * Model::parentSimulator() const {
00309     return _parentSimulator;
00310 }
00311
00312 ModelSimulation * Model::simulation() const {
00313     return _simulation;
00314 }
00315
00316 Util::identification Model::id() const {
00317     return _id;
00318 }
00319
```

7.239 Model.h File Reference

```
#include <string>
#include "List.h"
#include "ModelComponent.h"
#include "Event.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "ModelPersistence_if.h"
#include "ElementManager.h"
#include "ComponentManager.h"
#include "TraceManager.h"
#include "OnEventManager.h"
#include "ModelInfo.h"
#include "ModelSimulation.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Classes

- class [Model](#)

7.240 Model.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  SimulationModel.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Junho de 2018, 15:01
00012 */
00013
00014 #ifndef SIMULATIONMODEL_H
00015 #define SIMULATIONMODEL_H
00016
00017 #include <string>
00018
00019 #include "List.h"
00020 #include "ModelComponent.h"
00021 #include "Event.h"
00022 #include "ModelChecker_if.h"
00023 #include "Parser_if.h"
00024 #include "ModelPersistence_if.h"
00025 #include "ElementManager.h"
00026 #include "ComponentManager.h"
00027 #include "TraceManager.h"
00028 #include "OnEventManager.h"
00029 #include "ModelInfo.h"
00030 #include "ModelSimulation.h"
00031 //for PAN
00032 #include "SimulationResponse.h"
00033 #include "SimulationControl.h"
00034
00035 class Simulator;
00036
00043 class Model {
00044 public:
00045     Model(Simulator* simulator);
00046     virtual ~Model() = default;
00047 public: // model control
00048     //void showReports();
00049     bool save(std::string filename);
00050     bool load(std::string filename);
00051     bool check();
```

```

00052     void clear();
00053     void show();
00054     bool insert(ModelElement* elemOrComp);
00055     void remove(ModelElement* elemOrComp);
00056     //bool verifySymbol(std::string componentName, std::string expressionName, std::string expression,
00057     //std::string expressionResult, bool mandatory); ///< Verifies if a symbol defined in a component (ModelComponent)
00058     // or element is syntactically valid and addresses existing components or elements. It's used only by and
00059     // directed by the component that defines the symbol.
00060     void removeEntity(Entity* entity, bool collectStatistics);
00061     void sendEntityToComponent(Entity* entity,
00062     Connection* connection, double timeDelay);
00063     void sendEntityToComponent(Entity* entity,
00064     ModelComponent* component, double timeDelay, unsigned int componentInputNumber = 0);
00065     double parseExpression(const std::string expression);
00066     double parseExpression(const std::string expression, bool* success, std::string*
00067     errorMessage);
00068     bool checkExpression(const std::string expression, const std::string expressionName,
00069     std::string* errorMessage);
00070     public: // only gets
00071     Util::identification id() const;
00072     // 1:1
00073     List<SimulationControl*>* controls() const;
00074     List<SimulationResponse*>* responses() const;
00075     OnEventManager* onEvents() const;
00076     ElementManager* elements() const;
00077     ComponentManager* components() const;
00078     ModelInfo* infos() const;
00079     Simulator* parentSimulator() const;
00080     ModelSimulation* simulation() const;
00081     // 1:n
00082     //List<ModelComponent*>* getComponents() const; ///< Returns the list of components (such as Create,
00083     //Delay, Dispose, etc.) that make up the simulation model.
00084     List<Event*>* futureEvents() const;
00085     void setTraceManager(TraceManager* _traceManager);
00086     TraceManager* tracer() const;
00087     bool hasChanged() const;
00088     /*
00089      * PRIVATE
00090      */
00091     private:
00092     void _showComponents() const;
00093     void _showElements() const;
00094     void _showSimulationControls() const;
00095     void _showSimulationResponses() const;
00096     void _createModelInternalElements();
00097     private:
00098     bool _hasChanged = false;
00099     private: // read only public access (gets)
00100     Util::identification _id;
00101     Simulator* _parentSimulator;
00102     // 1:1 (associated classes)
00103     TraceManager* _traceManager;
00104     OnEventManager* _eventManager;
00105     ElementManager* _elementManager;
00106     ComponentManager* _componentManager;
00107     ModelInfo* _modelInfo;
00108     ModelSimulation* _simulation;
00109     // 1:n
00110     //List<ModelComponent*>* _components;
00111     List<Event*>* _events;
00112     // for process analyser
00113     List<SimulationResponse*>* _responses;
00114     List<SimulationControl*>* _controls;
00115
00116     private: // no public access (no gets / sets)
00117     ModelChecker_if* _modelChecker;
00118     ModelPersistence_if* _modelPersistence;
00119     Parser_if* _parser;
00120 };
00121 */
00122 #endif /* SIMULATIONMODEL_H */
00123

```

7.241 ModelChecker_if.h File Reference

```
#include <typeinfo>
#include <string>
```

Classes

- class ModelChecker_if

7.242 ModelChecker_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelChecker_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Agosto de 2018, 15:48
00012 */
00013
00014 #ifndef MODELCHECKER_IF_H
00015 #define MODELCHECKER_IF_H
00016
00017 #include <typeinfo>
00018 #include <string>
00019
00020 //#include "Model.h"
00021 //class Model;
00022
00023 class ModelChecker_if {
00024 public:
00025     virtual bool checkAll() = 0;
00026     virtual bool checkConnected() = 0;
00027     virtual bool checkSymbols() = 0;
00028     virtual bool checkActivationCode() = 0;
00029     virtual bool checkLimits() = 0;
00030     //virtual bool verifySymbol(std::string componentName, std::string expressionName, std::string
00031     //expression, std::string expressionResult, bool mandatory) = 0;
00032
00033 };
00034
00035 #endif /* MODELCHECKER_IF_H */
00036
00037 #endif /* MODELCHECKER_IF_H */
00038

```

7.243 ModelCheckerDefaultImpl1.cpp File Reference

```

#include "ModelCheckerDefaultImpl1.h"
#include "SourceModelComponent.h"
#include "SinkModelComponent.h"
#include "ComponentManager.h"
#include "Simulator.h"
#include <assert.h>

```

7.244 ModelCheckerDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelCheckerDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 8 de Agosto de 2018, 18:44
00012 */
00013
00014 #include "ModelCheckerDefaultImpl1.h"
00015 #include "SourceModelComponent.h"
00016 #include "SinkModelComponent.h"

```

```

00017 #include "ComponentManager.h"
00018 #include "Simulator.h"
00019
00020 #include <assert.h>
00021
00022 ModelCheckerDefaultImpl1::ModelCheckerDefaultImpl1(
    Model* model) {
00023     _model = model;
00024 }
00025
00026
00027 bool ModelCheckerDefaultImpl1::checkAll() {
00028     bool res = true;
00029     res &= checkSymbols();
00030     //res &= checkAndAddInternalLiterals();
00031     //res &= checkActivationCode();
00032     res &= checkLimits();
00033     res &= checkConnected();
00034     return res;
00035 }
00036
00037 //bool ModelCheckerDefaultImpl1::checkAndAddInternalLiterals() {
00038 //    /* \todo: +-: not implemented yet */
00039 //    return true;
00040 //}
00041
00042 void ModelCheckerDefaultImpl1::_recursiveConnectedTo(PluginManager* pluginManager,
    ModelComponent* comp, List<ModelComponent*>* visited,
    List<ModelComponent*>* unconnected, bool* drenoFound) {
00043     visited->insert(comp);
00044     _model->tracer()->trace(Util::TraceLevel::toolDetailed, "
Connected to component \" + comp->name() + "\");
00045     Plugin* plugin = pluginManager->find(comp->classname());
00046     assert(plugin!= nullptr);
00047     if (plugin->pluginInfo()->isSink() || (plugin->pluginInfo()->
isSendTransfer() && comp->nextComponents()->size()==0)) //(
(dynamic_cast<SinkModelComponent*> (comp) != nullptr) {
00048         // it is a sink OR it can send entities throught a transfer and has no nextConnections
00049         *drenoFound = true;
00050     } else { // it is not a sink
00051         if (comp->nextComponents()->size() == 0) {
00052             unconnected->insert(comp);
00053             _model->tracer()->trace(Util::TraceLevel::errorFatal, "
Component \" + comp->name() + "\" is unconnected (not a sink with no next componentes connected to)");
00054             *drenoFound = false;
00055         } else {
00056             ModelComponent* nextComp;
00057             for (std::list<Connection*>::iterator it = comp->nextComponents()->
list()->begin(); it != comp->nextComponents()->list()->end(); it++) {
00058                 nextComp = (*it)->first;
00059                 if (visited->find(nextComp) == visited->list()->end()) { // not visited yet
00060                     *drenoFound = false;
00061                     Util::IncIndent();
00062                     this->_recursiveConnectedTo(pluginManager, nextComp, visited, unconnected, drenoFound);
00063                     Util::DecIndent();
00064                 } else {
00065                     Util::IncIndent();
00066                     _model->tracer()->trace(
Util::TraceLevel::toolDetailed, "Connected to " + nextComp->
name());
00067                     Util::DecIndent();
00068                     *drenoFound = true;
00069                 }
00070             }
00071         }
00072     }
00073 }
00074
00075 /*
00076 procedure TModel._RecursiveConnectedTo(thisModule:TModule; var visited:TStringList; var unconnected:
TStringList; var drenofound:boolean);
00077 var plugin: TPlugin;
00078     index: integer;
00079     nextModule: TModule;
00080     i: integer;
00081 begin
00082     visited.Add(IntToStr(thisModule.ID)); {include this module as visited}
00083     index := Genesys.PlugInIndexOf(thisModule.Kind);
00084     Trace(cTLModuleDetail, Genesys.AuxFunctions.StrIdent(2)+'- Connected to ' + thisModule.Kind +': '+
thisModule.Name+ ' ('+IntToStr(thisModule.ID)+')');
00085     if index = -1 then begin
00086         {the module has no associated plugin. Impossible. Should do more, but will mark module as
unconnected}
00087         unconnected.Add(IntToStr(thisModule.ID));
00088         Trace(cTLError, 'Error: Module named "'+thisModule.Name+'" id "'+IntToStr(thisModule.ID)+'" has no
valid plugin "'+thisModule.Kind+'".');
00089     end

```

```

00090     else begin
00091         plugin := Genesys.PluginIn[index];
00092         if plugin.Dreno then begin
00093             index := 0; {singal to stop searching. Found a dreno}
00094             i := 0;
00095             while (i < thisModule.NextCount) and (index = 0) do begin
00096                 if thisModule.NextID[i] <> thisModule.ID then
00097                     index := 1; {need continue searching}
00098                     i := i + 1;
00099             end;
00100             if index = 0 then begin
00101                 {found a dreno that has no connection with other modules. Stop walking}
00102                 drenoFound := true;
00103             end;
00104             end;
00105             if not drenoFound then begin
00106                 if thisModule.NextCount = 0 then begin
00107                     //this module is not a DRENO and it has no next modules - it is a dead end
00108                     unconnected.Add(IntToStr(thisModule.ID));
00109                     Trace(cTLError,'Error: Module named "'+thisModule.Name+'" id "'+IntToStr(thisModule.ID)+'" is
00110                     unconnected (no next module).');
00111                     end
00112                     else begin
00113                         //continue the pathway thought the next modules
00114                         for index := 0 to thisModule.NextCount - 1 do begin
00115                             drenoFound := false;
00116                             nextModule := ModuleByID(thisModule.NextID[index]);
00117                             if (nextModule = nil) then begin
00118                                 {can be an invalid ID or a queue associated with a module}
00119                                 i := SIMAN.QueueIndex(thisModule.NextID[index]);
00120                                 if (i >= 0) then begin
00121                                     {is a queue. Try to get the module associated}
00122                                     nextModule := ModuleByID(SIMAN.Queue[i].ModuleID);
00123                                 end;
00124                                 if (nextModule = nil) then begin
00125                                     {invalid ID}
00126                                     unconnected.Add(IntToStr(thisModule.ID));
00127                                     Trace(cTLError,'Error: Module named "'+thisModule.Name+'" id "'+IntToStr(thisModule.ID)+'" is
00128                                     unconnected (invalid next id "'+IntToStr(thisModule.NextID[index])+'").';
00129                                     DrenoFound := true; {only one error is suficient}
00130                                 end;
00131                                 end;
00132                                 if (nextModule <> nil) then begin
00133                                     if visited.IndexOf(IntToStr(nextModule.ID)) = -1 then
00134                                         _RecursiveConnectedTo(nextModule, visited, unconnected, drenoFound) {continue searching}
00135                                     else
00136                                         drenoFound := true; {arrived at an already visited module}
00137                                     end;
00138                                 if not DrenoFound then begin
00139                                     Trace(cTLError,'Error: Module named "'+thisModule.Name+'" id "'+IntToStr(thisModule.ID)+'" is
00140                                     unconnected (takes to no dreno).';
00141                                     DrenoFound := true;
00142                                 end;
00143                             end;
00144                         end;
00145                     */
00146
00147     bool ModelCheckerDefaultImpl::checkConnected() {
00148         /* \todo: +-: not implemented yet */
00149         _model->tracer()->trace(Util::TraceLevel::toolInternal, "
00150             Checking connected");
00151         bool resultAll = true;
00152         PluginManager* pluginManager = this->_model->parentSimulator()->
00153             plugins();
00154         Plugin* plugin;
00155         Util::IncIndent();
00156         {
00157             List<ModelComponent*>* visited = new
00158                 List<ModelComponent*>();
00159             List<ModelComponent*>* unconnected = new
00160                 List<ModelComponent*>();
00161             ModelComponent* comp;
00162             for (std::list<ModelComponent*>::iterator it = _model->components()->
00163                 begin(); it != _model->components()->end(); it++) {
00164                 comp = (*it);
00165                 plugin = pluginManager->find(comp->classname());
00166                 assert(plugin != nullptr);
00167                 if (plugin->pluginInfo()->isSource() || plugin->
00168                     pluginInfo()->isReceiveTransfer()) { //dynamic_cast<SourceModelComponent*>
00169                     (comp) != nullptr) {
00170                         // it is a source component OR it can receive entities from transfer
00171                         bool drenoFound = false;
00172                         _recursiveConnectedTo(pluginManager, comp, visited, unconnected, &drenoFound);
00173                     }

```

```

00167      }
00168      // check if any component remains unconnected
00169      for (std::list<ModelComponent*>::iterator it = _model->components()->
00170          begin(); it != _model->components()->end(); it++) {
00171          comp = (*it);
00172          if (visited->find(comp) == visited->list()->end()) { //not found
00173              resultAll = false;
00174              _model->tracer()->trace(Util::TraceLevel::errorFatal, "
00175                  Component \" + comp->name() + "\" is unconnected.");
00176          }
00177      }
00178      Util::DecIndent();
00179      return resultAll;
00180 }
00181 */
00182 /*
00183 function TModel._CheckConnected: integer;
00184 var i, j: integer;
00185     kind: string;
00186     moduleID: word;
00187     criacoes,
00188     visited, unconnected, nonvisual: TStringList;
00189     thisModule: TModule;
00190     drenoFound: boolean;
00191 begin
00192     { Returns 0 if model is correctly connected, or the -ID of the first unconnected module}
00193     result := -1; {is wrong by default}
00194     Trace(cTLMModuleIntern, Genesys.AuxFunctions.StrIdent(1)+'- Verifying modules connections');
00195     visited := TStringList.Create;
00196     visited.Sorted := true;
00197     unconnected := TStringList.Create;
00198     unconnected.Sorted := true;
00199     {takes all source modules in the model}
00200     i := 0;
00201     while (i < Genesys.PlugInCount) do begin
00202         if Genesys.PlugIn[i].Source then begin
00203             kind := Genesys.PlugIn[i].Kind;
00204             //A SOURCE block has been found. Start searching a pathway
00205             Criacoes := ModulesOfKind[kind];
00206             if Criacoes <> nil then begin
00207                 for j := 0 to Criacoes.Count - 1 do begin
00208                     //for each SOURCE block, searchs for a pathway throught a DRENO
00209                     drenoFound := false;
00210                     thisModule := TModule(Criacoes.Objects[j]);
00211                     _RecursiveConnectedTo(thisModule, visited, unconnected, drenoFound);
00212                 end;
00213             end;
00214         end;
00215         // Non-visual blocks dont need to be connected
00216         if not Genesys.PlugIn[i].Visual then begin
00217             kind := Genesys.PlugIn[i].Kind;
00218             nonvisual := ModulesOfKind[kind];
00219             if nonvisual <> nil then begin
00220                 for j := 0 to nonvisual.Count - 1 do begin
00221                     visited.Add(IntToStr(TModule(nonvisual.Objects[j]).ID));
00222                 end;
00223             end;
00224         end;
00225         i := i + 1;
00226     end;
00227     {finished to walk throught the model}
00228     {verify if there are modules that were not visited}
00229     for i := 0 to aModules.Count - 1 do begin
00230         criacoes := TStringList(aModules.Objects[i]);
00231         for j := 0 to criacoes.Count - 1 do begin
00232             moduleID := TModule(criacoes.Objects[j]).ID;
00233             if visited.IndexOf(IntToStr(moduleID)) = -1 then begin
00234                 {this module couldn't be visited by any source module. It is inaccessible. Mark as unconnected}
00235                 unconnected.Add(IntToStr(moduleID));
00236                 Trace(cTLMError,'Error: Module id "'+IntToStr(moduleID)+'" is unconnected (not accessible).');
00237             end;
00238         end;
00239     end;
00240     if unconnected.Count = 0 then
00241         result := 0 {ok}
00242     else
00243         result := -1; {model has problems}
00244     end;
00245 */
00246 bool ModelCheckerDefaultImpl1::checkSymbols() {
00247     bool res = true;
00248     _model->tracer()->trace(Util::TraceLevel::toolInternal, "
00249         Checking symbols");
00250     Util::IncIndent();

```

```

00251      {
00252          // check components
00253          _model->tracer()->trace(Util::TraceLevel::toolDetailed, "
00254              Components:");
00255          {
00256              //List<ModelComponent*>* components = _model->getComponents();
00257              for (std::list<ModelComponent*>::iterator it = _model->components()->
00258                  begin(); it != _model->components()->end(); it++) {
00259                  res &= (*it)->Check((*it));
00260              }
00261          Util::DecIndent();
00262
00263          // check elements
00264          _model->tracer()->trace(Util::TraceLevel::toolDetailed, "
00265              Elements:");
00266          {
00267              std::string elementType;
00268              bool result;
00269              ModelElement* element;
00270              std::string* errorMessage = new std::string();
00271              std::list<std::string>* elementTypes = _model->elements()->
00272                  elementClassnames();
00273              for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end();
00274                  typeIt++) {
00275                  elementType = (*typeIt);
00276                  List<ModelElement*>* elements = _model->elements()->
00277                      elementList(elementType);
00278                  for (std::list<ModelElement*>::iterator it = elements->list()->begin(); it != elements->
00279                      list()->end(); it++) {
00280                      element = (*it);
00281                      // copied from modelComponent. It is not inside the ModelElement::Check because ModelElement
00282                      has no access to Model to call Tracer
00283                      _model->tracer()->trace(Util::TraceLevel::toolDetailed
00284                          , "Checking " + element->classname() + ":" + "\"" + element->name() + "\" (id " + std::to_string(
00285                              element->id()) + ")");
00286                      Util::IncIndent();
00287                      {
00288                          try {
00289                              result = element->Check((*it), errorMessage);
00290                              res &= result;
00291                              if (!result) {
00292                                  _model->tracer()->trace(Util::TraceLevel::errorFatal
00293                                      , "Error: Checking has failed with message '" + *errorMessage + "'");
00294                                  }
00295                              } catch (const std::exception& e) {
00296                                  _model->tracer()->traceError(e, "Error verifying component " + element->
00297                                      show());
00298                              }
00299                          }
00300                      }
00301                  Util::DecIndent();
00302
00303                  /*
00304                      _model->getTraceManager()->trace(Util::TraceLevel::mostDetailed, "Verifying symbols of " +
00305                          elementType + " " + element->getName()); //std::to_string(component->_id));
00306                      result = element->Check((*it), errorMessage);
00307                      res &= result;
00308                      if (!result) {
00309                          _model->getTraceManager()->trace(Util::TraceLevel::mostDetailed, "Verification of symbols of
00310                          component \" " + element->getName() + "\" has failed with message " + *errorMessage);
00311                      }
00312                  */
00313              }
00314          Util::DecIndent();
00315
00316
00317      }
00318  Util::DecIndent();
00319  return true;
00320 }
00321
00322 bool ModelCheckerDefaultImpl::checkLimits() {

```

```

00323     bool res = true;
00324     std::string text;
00325     unsigned int value, limit;
00326     LicenceManager *licence = _model->parentSimulator()->
00327         licenceManager();
00328     _model->tracer()->trace(Util::TraceLevel::toolInternal, "
00329         Checking model limits");
00330     Util::IncIndent();
00331     {
00332         value = _model->components()->numberOfComponents();
00333         limit=licence->modelComponentsLimit();
00334         res &= value <= limit;
00335         _model->tracer()->trace("Model has "+std::to_string(value)+"/"+std::to_string(limit)+""
00336             "components");
00337         if (!res) {
00338             text = "Model has " + std::to_string(_model->components()->
00339                 numberOfComponents()) + " components, exceeding the limit of " + std::to_string(licence->
00340                     modelComponentsLimit()) + " components imposed by the current activation code";
00341             // _model->getTraceManager()->trace(Util::TraceLevel::errors, text);
00342         } else {
00343             value = _model->elements()->numberOfElements();
00344             limit = licence->modelElementsLimit();
00345             res &= value <= limit;
00346             _model->tracer()->trace("Model has "+std::to_string(value)+"/"+std::to_string(limit)+""
00347                 "elements");
00348             if (!res) {
00349                 text = "Model has " + std::to_string(_model->elements()->
00350                     numberOfElements()) + " elements, exceeding the limit of " + std::to_string(licence->
00351                         modelElementsLimit()) + " elements imposed by the current activation code";
00352                 // _model->getTraceManager()->trace(Util::TraceLevel::errors, text);
00353             }
00354         }
00355     }
00356     if (!res) {
00357         _model->tracer()->trace(Util::TraceLevel::errorFatal, "
00358             Error: Checking has failed with message '" + text + "'");
00359     }
00360     Util::DecIndent();
00361     return res;
00362 }
00363 */
00364
00365 bool ModelCheckerDefaultImpl1::verifySymbol(std::string componentName, std::string expressionName,
00366     std::string expression, std::string expressionResult, bool mandatory) {
00367     bool res = true;
00368     if (mandatory && expression == "") {
00369         _model->getTraceManager()->getErrorMessages()->insert("Error verifying symbol \""
00370             + expressionName + "\\" + componentName + "\n: Mandatory symbol is empty");
00371         res = false;
00372     }
00373     // \todo: Not implemented yet
00374     return res;
00375 }
00376 */

```

7.245 ModelCheckerDefaultImpl1.h File Reference

```
#include "ModelChecker_if.h"
#include "Model.h"
#include "PluginManager.h"
```

Classes

- class [ModelCheckerDefaultImpl1](#)

7.246 ModelCheckerDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */

```

```

00006
00007 /*
00008 * File: ModelCheckerDefaultImpl1.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 8 de Agosto de 2018, 18:44
00012 */
00013
00014 #ifndef MODELCHECKERDEFAULTIMPL1_H
00015 #define MODELCHECKERDEFAULTIMPL1_H
00016
00017 #include "ModelChecker_if.h"
00018 #include "Model.h"
00019 #include "PluginManager.h"
00020
00021 class ModelCheckerDefaultImpl1 : public ModelChecker_if {
00022 public:
00023     ModelCheckerDefaultImpl1(Model* model);
00024     virtual ~ModelCheckerDefaultImpl1() = default;
00025 public:
00026     virtual bool checkAll();
00027     virtual bool checkConnected();
00028     virtual bool checkSymbols();
00029     virtual bool checkActivationCode();
00030     virtual bool checkLimits();
00031     //virtual bool verifySymbol(std::string componentName, std::string expressionName, std::string
00032     //expression, std::string expressionResult, bool mandatory);
00032 private:
00033     void _recursiveConnectedTo(PluginManager* pluginManager,
00034         ModelComponent* comp, List<ModelComponent*>* visited,
00035         List<ModelComponent*>* unconnected, bool* drenoFound);
00034 private:
00035     Model* _model;
00036 };
00037
00038 #endif /* MODELCHECKERDEFAULTIMPL1_H */
00039

```

7.247 ModelComponent.cpp File Reference

```
#include "ModelComponent.h"
#include "Model.h"
```

7.248 ModelComponent.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Element.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 15:56
00012 */
00013
00014 #include "ModelComponent.h"
00015 #include "Model.h"
00016
00017 ModelComponent::ModelComponent(Model* model, std::string
00018     componentTypename, std::string name) : ModelElement(model, componentTypename, name, false) {
00019     model->components()->insert(this);
00020 }
00021 ModelComponent::~ModelComponent(){
00022     _parentModel->components()->remove(this);
00023 }
00024
00025 void ModelComponent::Execute(Entity* entity,
00026     ModelComponent* component, unsigned int inputNumber) {
00027     std::string msg = "Entity " + std::to_string(entity->entityNumber()) + " has arrived at
00028     component \"" + component->_name + "\";
00029     // \todo: How can I know the number of inputs?
00030     if (inputNumber > 0)

```

```
00029     msg += " by input " + std::to_string(inputNumber);
00030     component->_parentModel->tracer()->trace(
00031         Util::TraceLevel::componentArrival, msg);
00032     Util::IncIndent();
00033     try {
00034         component->_execute(entity);
00035     } catch (const std::exception& e) {
00036         component->_parentModel->tracer()->traceError(e, "Error executing component
00037             " + component->show());
00038     }
00039     Util::DecIndent();
00040 */
00041 void ModelComponent::InitBetweenReplications(ModelComponent* component) {
00042     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component \\" + component->_name + "\\"); //std::to_string(component->_id));
00043     try {
00044         component->_initBetweenReplications();
00045     } catch (const std::exception& e) {
00046         component->_parentModel->tracer()->traceError(e, "Error initing component " + component->show());
00047     };
00048 }
00049 */
00050
00051 void ModelComponent::CreateInternalElements(
00052     ModelComponent* component) {
00053     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component \\" + component->_name + "\\"); //std::to_string(component->_id));
00054     try {
00055         component->_createInternalElements();
00056     } catch (const std::exception& e) {
00057         component->_parentModel->tracer()->traceError(e, "Error creating elements
00058             of component " + component->show());
00059     };
00060
00061 std::map<std::string, std::string>* ModelComponent::SaveInstance(
00062     ModelComponent* component) {
00063     component->_parentModel->tracer()->trace(
00064         Util::TraceLevel::componentDetailed, "Writing component \\" + component-
00065             >_name + "\\"); //std::to_string(component->_id));
00066     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00067     try {
00068         fields = component->_saveInstance();
00069     } catch (const std::exception& e) {
00070         component->_parentModel->tracer()->traceError(e, "Error executing component
00071             " + component->show());
00072     }
00073     return fields;
00074 }
00075
00076 bool ModelComponent::Check(ModelComponent* component) {
00077     component->_parentModel->tracer()->trace(
00078         Util::TraceLevel::componentDetailed, "Checking " + component->
00079             _typename + ": \\" + component->_name + "\\"); //std::to_string(component->_id));
00080     bool res = false;
00081     std::string* errorMessage = new std::string();
00082     Util::IncIndent();
00083     {
00084         try {
00085             res = component->_check(errorMessage);
00086             if (!res) {
00087                 component->_parentModel->tracer()->trace(
00088                     Util::TraceLevel::errorFatal, "Error: Checking has failed with message '" +
00089                         errorMessage + "'");
00090             }
00091         } catch (const std::exception& e) {
00092             component->_parentModel->tracer()->traceError(e, "Error verifying
00093                 component " + component->show());
00094         }
00095     }
00096     Util::DecIndent();
00097     return res;
00098 }
00099
00100 ConnectionManager* ModelComponent::nextComponents() const {
00101     return _nextComponents;
00102 }
00103
00104 std::string ModelComponent::show() {
00105     return ModelElement::show(); // "{id=" + std::to_string(this->_id) +
00106     ",name=\""+this->_name + "\\\"}"; // , nextComponents[]=( " + _nextComponents->show() + ")");
00107 }
00108
00109 bool ModelComponent::_loadInstance(std::map<std::string, std::string>* fields)
```

```

00099     bool res = ModelElement::_loadInstance(fields);
00100     if (res) {
00101         // Now it should load nextComponents. The problem is that the nextComponent may not be loaded yet.
00102         // So, what can be done is to temporarily load the ID of the nextComponents, and to wait until all the
00103         // components have been loaded to update nextComponents based on the temporarilyIDs now being loaded
00104         //unsigned short nextSize = std::stoi((*fields->find("nextSize")).second);
00105         //this->_tempLoadNextComponentsIDs = new List<Util::identification>();
00106         //for (unsigned short i = 0; i < nextSize; i++) {
00107             // Util::identification nextId = std::stoi((*fields->find("nextId" + std::to_string(i))).second);
00108             // this->_tempLoadNextComponentsIDs->insert(nextId);
00109         //}
00110     }
00111     return res;
00112 }
00113 std::map<std::string, std::string>* ModelComponent::_saveInstance() {
00114     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00115     fields->emplace("nextSize", std::to_string(this->_nextComponents->size()));
00116     unsigned short i = 0;
00117     for (std::list<Connection*>::iterator it = _nextComponents->list()->begin(); it != _nextComponents
00118 ->list()->end(); it++) {
00119         fields->emplace("nextId" + std::to_string(i), std::to_string((*it)->first->_id));
00120         fields->emplace("nextInputNumber" + std::to_string(i), std::to_string((*it)->second));
00121         i++;
00122     }
00123 }
00124
00125 void ModelComponent::_createInternalElements() {
00126
00127 }
00128
00129 */
00130 std::list<std::map<std::string, std::string>>* ModelComponent::_saveInstance(std::string type) {
00131     std::list<std::map<std::string, std::string>>* fields = ModelComponent::_saveInstance();
00132     fields->push_back(std::to_string(this->_nextComponents->size()));
00133     for (std::list<ModelComponent*>::iterator it=_nextGetComponentManager()->begin();
00134     it!=_nextGetComponentManager()->end(); it++) {
00135         fields->push_back((*it)->_name);
00136     }
00137 }
00138 */

```

7.249 ModelComponent.h File Reference

```

#include <string>
#include <list>
#include "Plugin.h"
#include "List.h"
#include "Entity.h"
#include "ModelElement.h"
#include "ConnectionManager.h"

```

Classes

- class [ModelComponent](#)

7.250 ModelComponent.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Element.h
00009  * Author: rafael.luiz.cancian
00010 */

```

```

00011 * Created on 21 de Junho de 2018, 15:56
00012 */
00013
00014 #ifndef MODELCOMPONENT_H
00015 #define MODELCOMPONENT_H
00016
00017 #include <string>
00018 #include <list>
00019
00020 #include "Plugin.h"
00021 #include "List.h"
00022 #include "Entity.h"
00023 #include "ModelElement.h"
00024 #include "ConnectionManager.h"
00025
00026 class Model;
00027
00028 class ModelComponent : public ModelElement {
00029 public:
00030     ModelComponent(Model* model, std::string componentTypename, std::string
00031         name = "");
00032     virtual ~ModelComponent();
00033 public:
00034     virtual std::string show();
00035 public:
00036     ConnectionManager* nextComponents() const;
00037 public:
00038     static void Execute(Entity* entity, ModelComponent* component, unsigned int
00039         inputNumber);
00040     //static void InitBetweenReplications(ModelComponent* component);
00041     static void CreateInternalElements(ModelComponent* component);
00042     static bool Check(ModelComponent* component);
00043     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00044         std::string>* fields);
00045     static std::map<std::string, std::string>* SaveInstance(
00046         ModelComponent* component);
00047 private:
00048     ConnectionManager* _nextComponents = new ConnectionManager();
00049     //List<Util::identification>* _tempLoadNextComponentsIDs; // initialize only when loading
00050 protected: // pure virtual methods
00051     virtual void _execute(Entity* entity) = 0;
00052 protected:
00053     virtual std::map<std::string, std::string>* _saveInstance();
00054     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00055     virtual void _createInternalElements();
00056
00057 protected:
00058     //Model* _parentModel;
00059 };
00060
00061 #endif /* MODELCOMPONENT_H */
00062

```

7.251 ModelElement.cpp File Reference

```

#include <iostream>
#include "ModelElement.h"
#include "Model.h"

```

7.252 ModelElement.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelElement.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 21 de Junho de 2018, 19:40
00012 */
00013
00014 // #include <typeinfo>

```

```

00015 #include <iostream>
00016 #include "ModelElement.h"
00017 #include "Model.h"
00018
00019 ModelElement::ModelElement(Model* model, std::string thistypename,
00020     std::string name, bool insertIntoModel) {
00020     _id = Util::GenerateNewId(); //GenerateNewIdOfType(thistypename);
00021     _typename = thistypename;
00022     _parentModel = model;
00023     if (name == "") {
00024         _name = thistypename + "_" + std::to_string(Util::GenerateNewIdOfType(
00024             thistypename));
00025     } else {
00026         _name = name;
00027     }
00028     if (insertIntoModel)
00029         model->insert(this);
00030
00031 //ModelElement::ModelElement(const ModelElement &orig) {
00032 //this->_parentModel = orig->_parentModel;
00033 //this->_name = "copy_of_" + orig->_name;
00034 //this->_typename = orig->_typename;
00035 //}
00036
00037 ModelElement::~ModelElement() {
00038     _parentModel->tracer()->trace(
00039         Util::TraceLevel::everythingMostDetailed, "Removing Element \""
00039         + this->_name + "\" from the model");
00040     {
00041         for (std::map<std::string, ModelElement*>::iterator it = _childrenElements->begin(); it
00041             != _childrenElements->end(); it++) {
00042             (*it).second->~ModelElement();
00043         }
00044     }
00045     Util::DecIndent();
00046     _parentModel->elements()->remove(this);
00047 }
00048
00049 bool ModelElement::_loadInstance(std::map<std::string, std::string>* fields) {
00050     bool res = true;
00051     std::map<std::string, std::string>::iterator it;
00052     it = fields->find("id");
00053     if (it != fields->end()) ? this->_id = std::stoi((*it).second) : res = false;
00054     it = fields->find("name");
00055     if (it != fields->end()) {
00056         this->_name = (*it).second;
00057     } else {
00058         res = false;
00059     }
00060     if (it != fields->end()) ? this->_name = (*it).second : res = false;
00061     return res;
00062 }
00063 std::map<std::string, std::string>* ModelElement::_saveInstance() {
00064     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00065     fields->emplace("typename", this->_typename);
00066     fields->emplace("id", std::to_string(this->_id));
00067     fields->emplace("name", this->_name);
00068     return fields;
00069 }
00070
00071 bool ModelElement::_check(std::string* errorMessage) {
00072     return true; // if there is no override, return true
00073 }
00074
00075 ParserChangesInformation*
00076     ModelElement::_getParserChangesInformation() {
00076     return new ParserChangesInformation(); // if there is no override, return no
00076     changes
00077 }
00078
00079 void ModelElement::_initBetweenReplications() {
00080
00081 }
00082
00083 /*
00084     std::list<std::map<std::string, std::string>>* ModelElement::_saveInstance(std::string type) {
00085         std::list<std::map<std::string, std::string>>* fields = ModelElement::_saveInstance();
00086         fields->push_back(type);
00087         return fields;
00088     }
00089 */
00090
00091 std::string ModelElement::show() {
00092     return "id=" + std::to_string(_id) + ",name=\"" + _name + "\"";
00093 }
00094

```

```
00095 std::list<std::string>* ModelElement::childrenElementKeys() const {
00096     std::list<std::string>* result = new std::list<std::string>();
00097     return result;
00098 }
00099
00100 Util::identification ModelElement::id() const {
00101     return _id;
00102 }
00103
00104 void ModelElement::setName(std::string _name) {
00105     this->_name = _name;
00106 }
00107
00108 std::string ModelElement::name() const {
00109     return _name;
00110 }
00111
00112 std::string ModelElement::classname() const {
00113     return _typename;
00114 }
00115
00116 //std::list<std::map<std::string, std::string>*>* ModelElement::_saveInstance() { /* \todo: REMOVE - IS PURE
00117     VIRTUAL TEMP */
00118     std::list<std::map<std::string, std::string>*>* fields = new std::map<std::string, std::string>();
00119     // return fields;
00120 }
00121
00122 void ModelElement::InitBetweenReplications(
00123     ModelElement* element) {
00124     //component->_model->getTraceManager()->trace(Util::TraceLevel::blockArrival, "Writing component \" +
00125     component->_name + "\""); //std::to_string(component->_id));
00126     try {
00127         element->_initBetweenReplications();
00128     } catch (const std::exception& e) {
00129         element->_parentModel->tracer()->traceError(e, "Error initing component " +
00130         element->show());
00131     }
00132 }
00133
00134 ModelElement* ModelElement::LoadInstance(
00135     Model* model, std::map<std::string, std::string>* fields, bool insertIntoModel) {
00136     std::string name = "";
00137     if (insertIntoModel) {
00138         // extracts the name from the fields even before "_laodInstance" and even before construct a new
00139         // ModelElement in such way when constructing the ModelElement, it's done with the correct name and that correct name
00140         // is show in trace
00141         std::map<std::string, std::string>::iterator it = fields->find("name");
00142         if (it != fields->end())
00143             name = (*it).second;
00144         ModelElement* newElement = new ModelElement(model, "ModelElement", name,
00145             insertIntoModel);
00146         try {
00147             newElement->_loadInstance(fields);
00148         } catch (const std::exception& e) {
00149         }
00150         return newElement;
00151     }
00152 }
00153
00154 std::map<std::string, std::string>* ModelElement::SaveInstance(
00155     ModelElement* element) {
00156     std::map<std::string, std::string>* fields; // = new std::list<std::string>();
00157     try {
00158         fields = element->_saveInstance();
00159     } catch (const std::exception& e) {
00160         //element->_model->getTrace()->traceError(e, "Error saving infra " + element->show());
00161     }
00162     return fields;
00163 }
00164
00165 bool ModelElement::Check(ModelElement* element, std::string* errorMessage) {
00166     // element->_model->getTraceManager()->trace(Util::TraceLevel::mostDetailed, "Checking " +
00167     element->_typename + ": " + element->_name); //std::to_string(element->_id));
00168     bool res = false;
00169     Util::IncIndent();
00170     {
00171         try {
00172             res = element->_check(errorMessage);
00173             if (!res) {
00174                 // element->_model->getTraceManager()->trace(Util::TraceLevel::errors, "Error:
00175                 Checking has failed with message '" + *errorMessage + "'");
00176             }
00177         } catch (const std::exception& e) {
00178             // element->_model->getTraceManager()->traceError(e, "Error verifying element " +
00179             element->show());
00180         }
00181     }
00182 }
```

```

00170      }
00171      Util::DecIndent();
00172      return res;
00173  }
00174
00175 void ModelElement::CreateInternalElements(
00176     ModelElement* element) {
00177     try {
00178         element->_createInternalElements();
00179     } catch (const std::exception& e) {
00180         //element->...->_model->getTraceManager()->traceError(e, "Error creating elements of element " +
00181         element->show());
00182     };
00183 }
00184
00185 }
```

7.253 ModelElement.h File Reference

```

#include <string>
#include <list>
#include <vector>
#include <map>
#include "Util.h"
#include "ParserChangesInformation.h"
```

Classes

- class [ModelElement](#)

7.254 ModelElement.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelElement.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 21 de Junho de 2018, 19:40
00012 */
00013
00014 #ifndef MODELEMENT_H
00015 #define MODELEMENT_H
00016
00017 #include <string>
00018 #include <list>
00019 #include <vector>
00020 #include <map>
00021 #include "Util.h"
00022
00023 #include "ParserChangesInformation.h"
00024
00025 class Model;
00026
00031 class ModelElement {
00032 public:
00033     ModelElement(Model* model, std::string elementTypename, std::string
00034         name = "", bool insertIntoModel = true);
00035     //ModelElement(Model* model, std::string elementTypename, std::string name = "", bool insertIntoModel =
00036     true);
00036     //ModelElement(const ModelElement &orig);
00037     virtual ~ModelElement();
00038 public: // get & set
```

```

00039     Util::identification id() const;
00040     void setName(std::string _name);
00041     std::string name() const;
00042     std::string classname() const;
00043 public: // static
00044     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00045         std::string>* fields, bool insertIntoModel); // \todo: return ModelComponent* ?
00046     static std::map<std::string, std::string>* SaveInstance(
00047         ModelElement* element);
00048     static bool Check(ModelElement* element, std::string* errorMessage);
00049     static void CreateInternalElements(ModelElement* element);
00050     static void InitBetweenReplications(ModelElement* element);
00051 public:
00052     virtual std::string show();
00053     std::list<std::string>* childrenElementKeys() const;
00054     ModelElement* childElement(std::string key) const;
00055 protected:
00056     void _setChildElement(std::string key, ModelElement* child);
00057 protected: // must be overridden by derived classes
00058     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00059 protected: // could be overridden by derived classes
00060     virtual bool _check(std::string* errorMessage);
00061     virtual ParserChangesInformation*
00062         _getParserChangesInformation();
00063     virtual void _initBetweenReplications();
00064 private:
00065     void _build(Model* model, std::string thistypename, bool insertIntoModel);
00066 protected:
00067     Util::identification _id;
00068     std::string _name;
00069     std::string _typename;
00070     Model* _parentModel;
00071 protected:
00072     std::map<std::string, ModelElement*>* _childrenElements = new
00073         std::map<std::string, ModelElement*>();
00074 #endif /* MODELELEMENT_H */
00075

```

7.255 ModellInfo.cpp File Reference

```

#include "ModelInfo.h"
#include "Model.h"
#include "Assign.h"

```

7.256 ModellInfo.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelInfo.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 7 de Novembro de 2018, 18:13
00012 */
00013
00014 #include "ModelInfo.h"
00015 #include "Model.h"
00016 #include "Assign.h"
00017
00018 ModelInfo::ModelInfo() {
00019     _name = "Model " + std::to_string(Util::GenerateNewIdOfType<Model>());
00020 }
00021 std::string ModelInfo::show(){
00022     return "analystName=\""+this->_analystName+"\""+
00023         ",description=\""+this->_description+"\""+
00024         ",name=\""+this->_name+"\""+

```

```
00025     ",numberOfReplications='"+std::to_string(this->_numberOfReplications)+  
00026     ",replicationLength='"+std::to_string(this->_replicationLength)+" "+  
00027     "Util::StrTimeUnit(this->_replicationLengthTimeUnit)+  
00028     ",terminatingCondition=\""+this->_terminatingCondition+"\""+  
00029     ",version='"+this->_version+  
00030     ",warmupTime='"+std::to_string(this->_warmUpPeriod)+" "+Util::StrTimeUnit(this->  
00031     _warmUpPeriodTimeUnit);  
00030 }  
00031  
00032 void ModelInfo::setName(std::string _name) {  
00033     this->_name = _name;  
00034     _hasChanged = true;  
00035 }  
00036  
00037 std::string ModelInfo::name() const {  
00038     return _name;  
00039 }  
00040  
00041 void ModelInfo::setAnalystName(std::string _analystName) {  
00042     this->_analystName = _analystName;  
00043     _hasChanged = true;  
00044 }  
00045  
00046 std::string ModelInfo::analystName() const {  
00047     return _analystName;  
00048 }  
00049  
00050 void ModelInfo::setDescription(std::string _description) {  
00051     this->_description = _description;  
00052     _hasChanged = true;  
00053 }  
00054  
00055 std::string ModelInfo::description() const {  
00056     return _description;  
00057 }  
00058  
00059 void ModelInfo::setTitle(std::string _projectTitle) {  
00060     this->_projectTitle = _projectTitle;  
00061     _hasChanged = true;  
00062 }  
00063  
00064 std::string ModelInfo::projectTitle() const {  
00065     return _projectTitle;  
00066 }  
00067  
00068 void ModelInfo::setVersion(std::string _version) {  
00069     this->_version = _version;  
00070     _hasChanged = true;  
00071 }  
00072  
00073 std::string ModelInfo::version() const {  
00074     return _version;  
00075 }  
00076  
00077 void ModelInfo::setNumberOfReplications(unsigned int  
00078     _numberOfReplications) {  
00079     this->_numberOfReplications = _numberOfReplications;  
00080     _hasChanged = true;  
00081 }  
00082 unsigned int ModelInfo::numberOfReplications() const {  
00083     return _numberOfReplications;  
00084 }  
00085  
00086 void ModelInfo::setReplicationLength(double _replicationLength) {  
00087     this->_replicationLength = _replicationLength;  
00088     _hasChanged = true;  
00089 }  
00090  
00091 double ModelInfo::replicationLength() const {  
00092     return _replicationLength;  
00093 }  
00094  
00095 void ModelInfo::setReplicationLengthTimeUnit(  
00096     Util::TimeUnit _replicationLengthTimeUnit) {  
00097     this->_replicationLengthTimeUnit = _replicationLengthTimeUnit;  
00098     _hasChanged = true;  
00099 }  
00100 Util::TimeUnit ModelInfo::replicationLengthTimeUnit()  
00101     const {  
00102         return _replicationLengthTimeUnit;  
00103     }  
00104 void ModelInfo::setWarmUpPeriod(double _warmUpPeriod) {  
00105     this->_warmUpPeriod = _warmUpPeriod;  
00106     _hasChanged = true;
```

```

00107 }
00108
00109 double ModelInfo::warmUpPeriod() const {
00110     return _warmUpPeriod;
00111 }
00112
00113 void ModelInfo::setWarmUpPeriodTimeUnit(
00114     Util::TimeUnit _warmUpPeriodTimeUnit) {
00115     this->_warmUpPeriodTimeUnit = _warmUpPeriodTimeUnit;
00116     _hasChanged = true;
00117 }
00118 Util::TimeUnit ModelInfo::warmUpPeriodTimeUnit() const {
00119     return _warmUpPeriodTimeUnit;
00120 }
00121
00122 void ModelInfo::setTerminatingCondition(std::string _terminatingCondition)
00123 {
00124     this->_terminatingCondition = _terminatingCondition;
00125     _hasChanged = true;
00126 }
00127 std::string ModelInfo::terminatingCondition() const {
00128     return _terminatingCondition;
00129 }
00130
00131
00132 void ModelInfo::loadInstance(std::map<std::string, std::string>* fields) {
00133     this->_analystName = (*fields->find("analystName")).second;
00134     this->_description = (*fields->find("description")).second;
00135     this->_name = (*fields->find("name")).second;
00136     this->_numberOfReplications = std::stoi((*fields->find("numberOfReplications")).second);
00137     this->_projectTitle = (*fields->find("projectTitle")).second;
00138     this->_replicationLength = std::stod((*fields->find("replicationLength")).second);
00139     this->_replicationLengthTimeUnit = static_cast<Util::TimeUnit>(std::stoi((*fields->find("replicationLengthTimeUnit")).second));
00140     this->_terminatingCondition = (*fields->find("terminatingCondition")).second;
00141     this->_version = (*fields->find("version")).second;
00142     this->_warmUpPeriod = std::stod((*fields->find("warmUpTime")).second);
00143     this->_warmUpPeriodTimeUnit = static_cast<Util::TimeUnit>(std::stoi((*fields->find("warmUpTimeTimeUnit")).second));
00144     _hasChanged = false;
00145 }
00146
00147 // \todo::: implement check method (to check things like terminating condition)
00148
00149 std::map<std::string, std::string>* ModelInfo::saveInstance() {
00150     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00151     fields->emplace("typename", "ModelInfo");
00152     fields->emplace("analystName", analystName());
00153     fields->emplace("description", description());
00154     fields->emplace("name", name());
00155     fields->emplace("numberOfReplications", std::to_string(numberOfReplications()));
00156     fields->emplace("projectTitle", projectTitle());
00157     fields->emplace("replicationLength", std::to_string(replicationLength()));
00158     fields->emplace("replicationLengthTimeUnit", std::to_string(static_cast<int>(
00159         replicationLengthTimeUnit())));
00160     fields->emplace("terminatingCondition", terminatingCondition());
00161     fields->emplace("version", version());
00162     fields->emplace("warmUpTime", std::to_string(warmUpPeriod()));
00163     fields->emplace("warmUpTimeTimeUnit", std::to_string(static_cast<int>(
00164         warmUpPeriodTimeUnit())));
00165     _hasChanged = false;
00166 }
00167 bool ModelInfo::hasChanged() const {
00168     return _hasChanged;
00169 }

```

7.257 ModellInfo.h File Reference

```
#include <string>
#include "Util.h"
```

Classes

- class [ModellInfo](#)

7.258 ModelInfo.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: ModelInfo.h
00008 * Author: rafael.luiz.cancian
00009 *
00010 *
00011 * Created on 7 de Novembro de 2018, 18:13
00012 */
00013
00014 #ifndef MODELINFO_H
00015 #define MODELINFO_H
00016
00017 #include <string>
00018 #include "Util.h"
00019
00020 class ModelInfo {
00021 public:
00022     ModelInfo();
00023     virtual ~ModelInfo() = default;
00024 public:
00025     std::string show();
00026     public:// gets and sets
00027     void setName(std::string _name);
00028     std::string name() const;
00029     void setAnalystName(std::string _analystName);
00030     std::string analystName() const;
00031     void setDescription(std::string _description);
00032     std::string description() const;
00033     void setProjectTitle(std::string _projectTitle);
00034     std::string projectTitle() const;
00035     void setVersion(std::string _version);
00036     std::string version() const;
00037     void setNumberOfReplications(unsigned int _numberOfReplications);
00038     unsigned int numberOfReplications() const;
00039     void setReplicationLength(double _replicationLength);
00040     double replicationLength() const;
00041     void setReplicationLengthTimeUnit(Util::TimeUnit
00042         _replicationLengthTimeUnit);
00043     Util::TimeUnit replicationLengthTimeUnit() const;
00044     void setWarmUpPeriod(double _warmUpPeriod);
00045     double warmUpPeriod() const;
00046     void setWarmUpPeriodTimeUnit(Util::TimeUnit _warmUpPeriodTimeUnit)
00047     ;
00048     Util::TimeUnit warmUpPeriodTimeUnit() const;
00049     void setTerminatingCondition(std::string _terminatingCondition);
00050     std::string terminatingCondition() const;
00051 public:
00052     void loadInstance(std::map<std::string, std::string>* fields);
00053     std::map<std::string, std::string>* saveInstance();
00054     bool hasChanged() const;
00055 private:// with public access (get & set)
00056     // model general information
00057     std::string _name;
00058     std::string _analystName = "";
00059     std::string _description = "";
00060     std::string _projectTitle = "";
00061     std::string _version = "1.0";
00062
00063     // replication and warmup duration
00064     unsigned int _numberOfReplications = 1;
00065     double _replicationLength = 60.0; // by default, 60 s
00066     Util::TimeUnit _replicationLengthTimeUnit =
00067         Util::TimeUnit::second;
00068     double _warmUpPeriod = 0.0;
00069     Util::TimeUnit _warmUpPeriodTimeUnit = Util::TimeUnit::second;
00070     std::string _terminatingCondition = "";
00071     bool _hasChanged = false;
00072 };
00073
00074 #endif /* MODELINFO_H */
00075

```

7.259 ModelManager.cpp File Reference

```
#include "ModelManager.h"
#include "List.h"
```

```
#include "Simulator.h"
```

7.260 ModelManager.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelManager.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Maio de 2019, 08:37
00012 */
00013
00014 #include "ModelManager.h"
00015 #include "List.h"
00016 #include "Simulator.h"
00017
00018 ModelManager::ModelManager(Simulator* simulator) {
00019     _simulator = simulator;
00020     _currentModel = nullptr;
00021 }
00022
00023 void ModelManager::insert(Model* model) {
00024     _models->insert(model);
00025     this->_currentModel = model;
00026     _simulator->tracer()->trace(Util::TraceLevel::simulatorResult
00027 , "Model successfully inserted");
00028 }
00029 void ModelManager::remove(Model* model) {
00030     _models->remove(model);
00031     if (_currentModel == model) {
00032         _currentModel = this->front();
00033     }
00034     model->~Model();
00035     _simulator->tracer()->trace(Util::TraceLevel::simulatorResult
00036 , "Model successfully removed");
00037 }
00038 unsigned int ModelManager::size() {
00039     return _models->size();
00040 }
00041
00042 bool ModelManager::saveModel(std::string filename) {
00043     if (_currentModel != nullptr)
00044         return _currentModel->save(filename);
00045     return false;
00046 }
00047
00048 bool ModelManager::loadModel(std::string filename) {
00049     Model* model = new Model(_simulator);
00050     bool res = model->load(filename);
00051     if (res) {
00052         this->insert(model);
00053         _simulator->tracer()->trace(Util::TraceLevel::simulatorResult
00054 , "Model successfully loaded");
00055     } else {
00056         model->~Model();
00057         _simulator->tracer()->trace(Util::TraceLevel::simulatorResult
00058 , "Model could not be loaded");
00059     }
00060 }
00061 void ModelManager::setCurrent(Model* model) {
00062     this->_currentModel = model;
00063 }
00064
00065 Model* ModelManager::current() {
00066     return _currentModel;
00067 }
00068
00069 Model* ModelManager::front() {
00070     return _models->front();
00071 }
00072
00073 Model* ModelManager::next() {
00074     return _models->next();
00075 }
```

7.261 ModelManager.h File Reference

```
#include "Model.h"
#include "TraceManager.h"
```

Classes

- class [ModelManager](#)

7.262 ModelManager.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelManager.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 31 de Maio de 2019, 08:37
00012 */
00013
00014 #ifndef MODELMANAGER_H
00015 #define MODELMANAGER_H
00016
00017 #include "Model.h"
00018 #include "TraceManager.h"
00019
00020 class ModelManager {
00021 public:
00022     ModelManager(Simulator* simulator);
00023     virtual ~ModelManager() = default;
00024 public:
00025     void insert(Model* model);
00026     void remove(Model* model);
00027     void setCurrent(Model* model);
00028     bool saveModel(std::string filename);
00029     bool loadModel(std::string filename);
00030     unsigned int size();
00031 public:
00032     Model* front();
00033     Model* current();
00034     Model* next();
00035     //Model* end();
00036 private:
00037     List<Model*>* _models = new List<Model*>();
00038     Model* _currentModel;
00039 private:
00040     Simulator* _simulator;
00041 };
00042
00043 #endif /* MODELMANAGER_H */
```

7.263 ModelPersistence_if.h File Reference

```
#include <string>
```

Classes

- class [ModelPersistence_if](#)

7.264 ModelPersistence_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelPersistence_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 24 de Agosto de 2018, 19:22
00012 */
00013
00014 #ifndef MODEL_PERSISTENCE_IF_H
00015 #define MODEL_PERSISTENCE_IF_H
00016
00017 #include <string>
00018
00022 class ModelPersistence_if {
00023 public:
00024     // \todo: not a good interface for sure. The Bridge pattern should be a lot better
00025     virtual bool save(std::string filename) = 0;
00026     virtual bool load(std::string filename) = 0;
00027     virtual bool hasChanged() = 0;
00028 private:
00029
00030 };
00031
00032 #endif /* MODEL_PERSISTENCE_IF_H */
00033

```

7.265 ModelPersistenceDefaultImpl1.cpp File Reference

```

#include "ModelPersistenceDefaultImpl1.h"
#include <fstream>
#include <ctime>
#include <regex>
#include <cassert>
#include "ModelComponent.h"
#include "Simulator.h"

```

7.266 ModelPersistenceDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelPersistenceDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:41
00012 */
00013
00014 #include "ModelPersistenceDefaultImpl1.h"
00015 #include <fstream>
00016 #include <ctime>
00017 #include <regex>
00018 #include <cassert>
00019 #include "ModelComponent.h"
00020 #include "Simulator.h"
00021
00022 ModelPersistenceDefaultImpl1::ModelPersistenceDefaultImpl1
00023     (Model* model) {
00024         _model = model;
00025     }
00026 std::map<std::string, std::string>* ModelPersistenceDefaultImpl1::_getSimulatorInfoFieldsToSave() {

```

```

00027     std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00028     fields->emplace("typename", "SimulatorInfo");
00029     fields->emplace("name", _model->parentSimulator()->name());
00030     fields->emplace("version", _model->parentSimulator()->version());
00031     return fields;
00032 }
00033
00034 bool ModelPersistenceDefaultImpl::save(std::string filename) {
00035     _model->tracer()->trace(Util::TraceLevel::toolInternal, "
00036     Saving file \"" + filename + "\"");
00037     Util::IncIndent();
00038     std::list<std::string> *simulInfosToSave, *modelInfosToSave, *modelElementsToSave;
00039     {
00040         //bool res = true;
00041         std::map<std::string, std::string>* fields;
00042         fields = _getSimulatorInfoFieldsToSave();
00043         simulInfosToSave = _adjustFieldsToSave(fields);
00044         // save model own infos
00045         fields = _model->infos()->saveInstance();
00046         modelInfosToSave = _adjustFieldsToSave(fields);
00047         // save infras
00048         modelElementsToSave = new std::list<std::string>();
00049         std::list<std::string>* infraTypenames = _model->elements()->
00050             elementClassnames();
00051         for (std::list<std::string>::iterator itTypenames = infraTypenames->begin(); itTypenames != infraTypenames->end(); itTypenames++) {
00052             if ((*itTypenames) != Util::TypeOf<StatisticsCollector>() && (*itTypenames) != Util::TypeOf<Counter>()) { // STATISTICSCOLLECTR and COUNTERs do NOT need to be saved
00053                 List<ModelElement*>* infras = _model->elements()->
00054                     elementList((*itTypenames));
00055                 _model->tracer()->trace(Util::TraceLevel::toolDetailed, "
00056                 Writing elements of type \"" + (*itTypenames) + "\":");
00057                 Util::IncIndent();
00058                 {
00059                     for (std::list<ModelElement*>::iterator it = infras->list()->begin(); it != infras->
00060                         list()->end(); it++) {
00061                         _model->tracer()->trace(Util::TraceLevel::toolDetailed,
00062                             "Writing " + (*itTypenames) + " \" " + (*it)->name() + "\"");
00063                         fields = (*it)->SaveInstance((*it));
00064                         Util::IncIndent();
00065                         modelElementsToSave->merge(*_adjustFieldsToSave(fields));
00066                         Util::DecIndent();
00067                     }
00068                     Util::DecIndent();
00069                 }
00070             }
00071             // save components
00072             _model->tracer()->trace(Util::TraceLevel::toolDetailed, "
00073             Writing components\":");
00074             //List<ModelComponent*>* components = this->_model->getComponents();
00075             modelComponentsToSave = new std::list<std::string>();
00076             Util::IncIndent();
00077             {
00078                 for (std::list<ModelComponent*>::iterator it = _model->components()->
00079                     begin(); it != _model->components()->end(); it++) {
00080                     fields = (*it)->SaveInstance((*it));
00081                     Util::IncIndent();
00082                     modelComponentsToSave->merge(*_adjustFieldsToSave(fields));
00083                     Util::DecIndent();
00084                 }
00085                 // open file
00086                 std::ofstream savefile;
00087                 savefile.open(filename, std::ofstream::out);
00088                 savefile << "# Genesys simulation model " << std::endl;
00089                 time_t now = time(0);
00090                 char* dt = ctime(&now);
00091                 savefile << "# Last saved on " << dt;
00092                 savefile << "# simulator infos" << std::endl;
00093                 _saveContent(simulInfosToSave, &savefile);
00094                 savefile << "# model infos" << std::endl;
00095                 _saveContent(modelInfosToSave, &savefile);
00096                 savefile << "# model elements" << std::endl;
00097                 _saveContent(modelElementsToSave, &savefile);
00098                 savefile << "# model components" << std::endl;
00099                 _saveContent(modelComponentsToSave, &savefile);
00100             }
00101             Util::DecIndent();

```

```

00102     }
00103     Util::DecIndent();
00104     this->_hasChanged = false;
00105     return true; // \todo: check if save really saved successfully
00106 }
00107
00108 void ModelPersistenceDefaultImpl1::_saveContent(std::list<std::string>* content, std::ofstream* file) {
00109     for (std::list<std::string>::iterator it = content->begin(); it != content->end(); it++) {
00110         *file << (*it) << std::endl << std::endl;
00111     }
00112 }
00113
00114 bool ModelPersistenceDefaultImpl1::_loadFields(std::string line) {
00115     //std::regex regex{R"([=]+)"}; // split on space R"([\s]+)" \todo: HOW SEPARATOR WITH MORE THAN ONE
00116     CHAR
00117     _model->tracer()->trace(Util::TraceLevel::everythingMostDetailed
00118     ,line);
00119     bool res = true;
00120     std::regex regex{R"([;]+)"}; // split on "; ". \todo: How change it by the attribute
00121     _linefieldseparator ??
00122     std::sregex_token_iterator tit{line.begin(), line.end(), regex, -1};
00123     std::list<std::string> lstfields{tit,{}};
00124     // for each field, separate key and value and form a map
00125     try {
00126         std::map<std::string, std::string>* fields = new std::map<std::string, std::string>();
00127         regex = {R"([=]+)"};
00128         std::vector<std::string> veckeyval; // {it,{}};
00129         for (std::list<std::string>::iterator it = lstfields.begin(); it != lstfields.end(); it++) {
00130             //std::cout << (*it) << std::endl;
00131             tit = {(*it).begin(), (*it).end(), regex, -1};
00132             veckeyval = {tit,{}};
00133             trim(veckeyval[0]);
00134             if (veckeyval[0] != "") {
00135                 if (veckeyval.size() > 1) {
00136                     trim(veckeyval[1]);
00137                     fields->emplace(veckeyval[0], veckeyval[1]);
00138                 } else {
00139                     fields->emplace(veckeyval[0], "");
00140                 }
00141             }
00142         }
00143         // now the map<str,str> is ready. Look for the right class to load it
00144         Util::IncIndent();
00145     {
00146         std::string thistypename = (*fields->find("typename")).second;
00147         _model->tracer()->trace(Util::TraceLevel::toolInternal, "
00148     loading " + thistypename + "");
00149         if (thistypename == "SimulatorInfo") {
00150             this->_loadSimulatorInfoFields(fields);
00151         } else if (thistypename == "ModelInfo") {
00152             _model->infos()->loadInstance(fields);
00153         } else {
00154             // this should be a ModelComponent or ModelElement.
00155             //std::string thistypename = (*fields->find("typename")).second;
00156             ModelElement* newTemUselessElement =
00157                 ModelElement::LoadInstance(_model, fields, false);
00158             if (newTemUselessElement != nullptr) {
00159                 newTemUselessElement->~ModelElement();
00160                 Plugin* plugin = this->_model->parentSimulator()->
00161                     plugins()->find(thistypename);
00162                 if (plugin != nullptr) {
00163                     res = plugin->loadAndInsertNew(_model, fields);
00164                     // save fields for components, in order to allow to connect components after all of them have
00165                     been loaded
00166                     if (res && plugin->pluginInfo()->isComponent()) {
00167                         _componentFields->insert(_componentFields->end(), fields);
00168                         //_model->getTraceManager()->trace(Util::TraceLevel::errors, "Inserindo fields do
00169                         componente "+plugin->getPluginInfo()->getPluginTypename());
00170                     }
00171                 } else {
00172                     _model->tracer()->trace(Util::TraceLevel::errorFatal, "
00173                         Error loading file: Could not identity typename \" + thistypename + "\");
00174                     res = false;
00175                 }
00176             } catch (...) {
00177             }
00178         }
00179     }
00180 }
```

```

00179 }
00180
00181 void ModelPersistenceDefaultImpl::loadSimulatorInfoFields(std::map<std::string, std::string>* fields) {
00182 }
00183
00184 bool ModelPersistenceDefaultImpl::load(std::string filename) {
00185     //std::list<Plugin*> plugins = this->_model->getParent()->getPlugins();
00186     //plugins->front()->
00187     //return false;
00188     bool res = true;
00189     _model->tracer()->trace(Util::TraceLevel::toolInternal, "
00190         Loading file \"" + filename + "\"");
00191     Util::IncIndent();
00192     _componentFields->clear();
00193     {
00194         std::ifstream modelFile;
00195         std::string inputLine;
00196         try {
00197             modelFile.open(filename);
00198             while (getline(modelFile, inputLine) && res) {
00199                 //trim(&inputLine);
00200                 if (inputLine.substr(0, 1) != "#" && !inputLine.empty()) {
00201                     //Util::IncIndent();
00202                     res &= _loadFields(inputLine);
00203                     //Util::DecIndent();
00204                 }
00205             }
00206         } catch (...) {
00207             _model->tracer()->trace(Util::TraceLevel::errorFatal, "Error
00208             loading file \"" + filename + "\"");
00209         }
00210         // check if something was loaded
00211         res &= _model->components()->numberOfComponents()>0 & _model->
00212         elements()->numberOfElements()>0;
00213         if (res) {
00214             // connect loaded components
00215             ComponentManager* cm = _model->components();
00216             _model->tracer()->trace(Util::TraceLevel::simulatorDetailed
00217             , "Connecting loaded components");
00218             Util::IncIndent();
00219             for (std::list<std::map<std::string, std::string>*>::iterator it = _componentFields->begin(); it != _componentFields->end(); it++) {
00220                 std::map<std::string, std::string>* fields = (*it);
00221                 // find the component
00222                 ModelComponent* thisComponent = nullptr;
00223                 Util::identification thisId = std::stoi((*fields->find("id")).second);
00224                 for (std::list<ModelComponent*>::iterator itcomp = cm->begin(); itcomp != cm->
00225                     end(); itcomp++) {
00226                     if ((*itcomp)->id() == thisId) {
00227                         thisComponent = (*itcomp);
00228                         break; // end inner for loop
00229                     }
00230                     assert(thisComponent != nullptr);
00231                     // find the next components connected with this one
00232                     unsigned short nextSize = std::stoi((*fields->find("nextSize")).second);
00233                     for (unsigned short i = 0; i < nextSize; i++) {
00234                         Util::identification nextId = std::stoi((*fields->find("nextId" +
00235                             std::to_string(i)).second);
00236                         unsigned short nextInputNumber = 0;
00237                         if ((*fields->find("nextInputNumber" + std::to_string(i)) != fields->end()))
00238                             nextInputNumber = std::stoi((*fields->find("nextInputNumber" + std::to_string(i))).second);
00239                         ModelComponent* nextComponent = nullptr;
00240                         for (std::list<ModelComponent*>::iterator itcomp = cm->begin(); itcomp != cm->
00241                             end(); itcomp++) { // connect the components
00242                             if ((*itcomp)->id() == nextId) { // connect the components
00243                                 nextComponent = (*itcomp);
00244                                 thisComponent->nextComponents()->insert(nextComponent, nextInputNumber)
00245                                 ;
00246                                 _model->tracer()->trace(
00247                                     Util::TraceLevel::toolDetailed, thisComponent->
00248                                     name() + " -> " + nextComponent->name());
00249                                 break;
00250                             }
00251                         }
00252                         Util::DecIndent();
00253                         _model->tracer()->trace(Util::TraceLevel::simulatorInternal
00254                         , "File successfully loaded with " + std::to_string(_model->components() ->

```

```

    numberOfRowsInSection())+" components and "+std::to_string(_model->
00254     elements()->numberOfElements())+" elements";
00255     }
00256     Util::DecIndent();
00257     if (res) {
00258         _hasChanged = false;
00259     }
00260     return res;
00261 }
00262 std::list<std::string>* ModelPersistenceDefaultImpl1::_adjustFieldsToSave(std::map<std::string,
00263     std::string>* fields) {
00264     std::list<std::string>* newList = new std::list<std::string>();
00265     std::string newStr;
00266     for (std::map<std::string, std::string>::iterator it = fields->begin(); it != fields->end(); it++) {
00267         newStr += (*it).first + "=" + (*it).second + this->_linefieldseparator;
00268     }
00269     _model->tracer()->trace(Util::TraceLevel::toolDetailed, newStr
00270 );
00271     newList->push_back(newStr);
00272     return newList;
00273 }
00274 bool ModelPersistenceDefaultImpl1::hasChanged() {
00275     return _hasChanged;
00276 }
```

7.267 ModelPersistenceDefaultImpl1.h File Reference

```
#include "ModelPersistence_if.h"
#include "Model.h"
```

Classes

- class [ModelPersistenceDefaultImpl1](#)

7.268 ModelPersistenceDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelPersistenceDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:41
00012 */
00013
00014 #ifndef MODEL_PERSISTENCE_DEFAULT_IMPL1_H
00015 #define MODEL_PERSISTENCE_DEFAULT_IMPL1_H
00016
00017 #include "ModelPersistence_if.h"
00018 #include "Model.h"
00019
00020 class ModelPersistenceDefaultImpl1 : public
00021     ModelPersistence_if {
00022     public:
00023         ModelPersistenceDefaultImpl1(Model* model);
00024         virtual ~ModelPersistenceDefaultImpl1() = default;
00025     public:
00026         virtual bool save(std::string filename);
00027         virtual bool load(std::string filename);
00028         virtual bool hasChanged();
00029     private:
00030         void _saveContent(std::list<std::string>* content, std::ofstream* file);
00031         bool _loadFields(std::string line);
00032         void _loadSimulatorInfoFields(std::map<std::string, std::string>* fields);
00033         std::list<std::string>* _adjustFieldsToSave(std::map<std::string, std::string>* fields);
00034         std::map<std::string, std::string>* _getSimulatorInfoFieldsToSave();
```

```

00034 private:
00035     std::list<std::map<std::string, std::string>>> _componentFields = new std::list<std::map<std::string,
00036     std::string>>();
00037     Model* _model = nullptr;
00038     bool _hasChanged = false;
00039     std::string _linefieldseparator = "; ";
00040 };
00041
00042 #endif /* MODEL PERSISTENCE DEFAULT IMPL1_H */
00043

```

7.269 ModelSimulation.cpp File Reference

```

#include "ModelSimulation.h"
#include <iostream>
#include "Model.h"
#include "Simulator.h"
#include "SourceModelComponent.h"
#include "StatisticsCollector.h"
#include "Counter.h"
#include "Traits.h"
#include "SimulationControl.h"
#include "ComponentManager.h"

```

7.270 ModelSimulation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ModelSimulation.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 18:04
00012 */
00013
00014 #include "ModelSimulation.h"
00015 #include <iostream>
00016 #include "Model.h"
00017 #include "Simulator.h"
00018 #include "SourceModelComponent.h"
00019 #include "StatisticsCollector.h"
00020 #include "Counter.h"
00021 #include "Traits.h"
00022 #include "SimulationControl.h"
00023 #include "ComponentManager.h"
00024
00025 ModelSimulation::ModelSimulation(Model* model) {
00026     _model = model;
00027     _info = model->infos(); // why??
00028     _statsCountersSimulation->setSortFunc([] (const ModelElement* a, const
00029         ModelElement* b) {
00030         return a->id() < b->id();
00031     });
00032     _simulationReporter = new Traits<SimulationReporter_if>::Implementation
00033     (this, model, this->_statsCountersSimulation);
00034 }
00035
00036 bool ModelSimulation::_isReplicationEndCondition() {
00037     bool finish = _model->futureEvents()->size() == 0;
00038     finish |= _model->parseExpression(_info->terminatingCondition()) !=
0.0;
00039     if (_model->futureEvents()->size() > 0 && !finish) {
00040         // replication length has not been achieve (so far), but next event will happen after that, so it's
00041         // just fine to set now as the replicationLength
00042         finish |= _model->futureEvents()->front()->time() > _info->
00043             replicationLength();
00044     }

```

```

00041     return finish;
00042 }
00043
00047 void ModelSimulation::start() {
00048     if (!_model->check()) {
00049         _model->tracer()->trace(Util::TraceLevel::errorFatal, "Model
00050             check failed. Cannot start simulation.");
00051     }
00052     Util::SetIndent(0); //force indentation
00053     _initSimulation();
00054     _model->onEvents()->NotifySimulationStartHandlers(new
00055         SimulationEvent(0, nullptr));
00056     for (_currentReplicationNumber = 1; _currentReplicationNumber <= _info->
00057         numberOfReplications(); _currentReplicationNumber++) {
00058         Util::SetIndent(1);
00059         _initReplication();
00060         _model->onEvents()->NotifyReplicationStartHandlers(new
00061             SimulationEvent(_currentReplicationNumber, nullptr));
00062
00063         Util::IncIndent();
00064     }
00065     while (!_isReplicationEndCondition()) {
00066         _stepSimulation();
00067         // \todo: Find a better way to separate start, step, pause, stop. Should allow to step without
00068         // start, start, pause and then just step, and in the last step invoke the final part of simulation (wich is
00069         // actually inside start method)
00070         //if (_pauseOnEvent) {
00071         //    std::cout << "[paused] ...press any key to continue..."; // std::cin.get();
00072         //    std::cout << std::endl;
00073         //}
00074     }
00075     Util::SetIndent(1); // force
00076
00077     _model->onEvents()->NotifyReplicationEndHandlers(new
00078         SimulationEvent(_currentReplicationNumber, nullptr));
00079     std::string causeTerminated = "";
00080     if (_model->futureEvents()->empty()) {
00081         causeTerminated = "event queue is empty";
00082     } else if (_stopRequested) {
00083         causeTerminated = "user requested to stop";
00084     } else if (_model->futureEvents()->front()->time() > _info->
00085         replicationLength()) {
00086         causeTerminated = "replication length " + std::to_string(_info->
00087             replicationLength()) + " was achieved";
00088     } else if (_model->parseExpression(_info->
00089         terminatingCondition())) {
00090         causeTerminated = "termination condition was achieved";
00091     } else causeTerminated = "unknown";
00092     std::string message = "Replication " + std::to_string(_currentReplicationNumber) + " of " +
00093         std::to_string(_info->numberOfReplications()) + " has finished with last event at time " +
00094         std::to_string(_simulatedTime) + " because " + causeTerminated;
00095     _model->tracer()->trace(Util::TraceLevel::modelSimulationEvent
00096         , message);
00097     _simulationReporter->showReplicationStatistics();
00098     _actualizeSimulationStatistics();
00099
00100     _simulationReporter->showSimulationStatistics(); //_cStatsSimulation);
00101     Util::DecIndent();
00102
00103     _model->tracer()->trace(Util::TraceLevel::modelSimulationEvent
00104         , "Simulation of model \"" + _info->name() + "\" has finished.\n");
00105     _model->onEvents()->NotifySimulationEndHandlers(new
00106         SimulationEvent(0, nullptr));
00107 }
00108
00109 void ModelSimulation::_actualizeSimulationStatistics() {
00110     //\todo: should not be only CSTAT and COUNTER, but any element that generateReportInformation
00111     const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00112     const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00113
00114     StatisticsCollector *sc, *scSim;
00115     //ModelElement* me;
00116     List<ModelElement*>* cstats = _model->elements()->
00117     elementList(Util::TypeOf<StatisticsCollector>());
00118     for (std::list<ModelElement*>::iterator itMod = cstats->list()->begin(); itMod != cstats->
00119         list()->end(); itMod++) {
00120         sc = dynamic_cast<StatisticsCollector*> ((*itMod));
00121         scSim = nullptr;
00122         for (std::list<ModelElement*>::iterator itSim = _statsCountersSimulation->
00123             list()->begin(); itSim != _statsCountersSimulation->list()->end(); itSim++) {
00124             if ((*itSim)->classname() == UtilTypeOfStatisticsCollector) {
00125                 if ((*itSim)->name() == _cte_stCountSimulNamePrefix + sc->name() && dynamic_cast<
00126                     StatisticsCollector*> ((*itSim)->getParent()) == sc->getParent()) {

```

```

00112         // found
00113         scSim = dynamic_cast<StatisticsCollector*> (*itSim);
00114         break;
00115     }
00116 }
00117 }
00118 //scSim = dynamic_cast<StatisticsCollector*> ((*this->_statsCountersSimulation->find((*it)))); 
00119 if (scSim == nullptr) {
00120     // this is a cstat created during the last replication
00121     //      scSim = new StatisticsCollector(_model->elements(), sc->name(), sc->getParent());
00122     _statsCountersSimulation->insert(scSim);
00123 }
00124 assert(scSim != nullptr);
00125 scSim->getStatistics()->getCollector()->addValue(sc->getStatistics()->average());
00126 }
00127 Counter *cnt; //, *cntSim;
00128 List<ModelElement*>* counters = _model->elements()->
00129 elementList(Util::TypeOf<Counter>());
00130 for (std::list<ModelElement*>::iterator itMod = counters->list()->begin(); itMod != counters->
00131 list()->end(); itMod++) {
00132     cnt = dynamic_cast<Counter*> ((*itMod));
00133     //cntSim = nullptr;
00134     scSim = nullptr;
00135     for (std::list<ModelElement*>::iterator itSim = _statsCountersSimulation->
00136 list()->begin(); itSim != _statsCountersSimulation->list()->end(); itSim++) {
00137         if ((*itSim)->classname() == UtilTypeOf<StatisticsCollector> {
00138             //_model->getTraceManager()->trace(Util::TraceLevel::simulation, (*itSim)->getName() + " == "
00139             "+_cte_stCountSimulNamePrefix + cnt->getName());
00140             if ((*itSim)->name() == _cte_stCountSimulNamePrefix + cnt->name() && dynamic_cast<
00141             StatisticsCollector*> ((*itSim)->getParent()) == cnt->
00142             getParent()) {
00143                 // found
00144                 scSim = dynamic_cast<StatisticsCollector*> (*itSim);
00145                 break;
00146             }
00147         }
00148     }
00149 }
00150 */
00151 }
00152 /*
00153 assert(cntSim != nullptr);
00154 cntSim->incCountValue(cnt->getCountValue());
00155 */
00156 assert(scSim != nullptr);
00157 scSim->getStatistics()->getCollector()->addValue(cnt->getCountValue());
00158 }
00159 }
00160
00161 void ModelSimulation::_showSimulationHeader() {
00162     TraceManager* tm = _model->tracer();
00163     tm->traceReport("\n-----");
00164     // simulator infos
00165     tm->traceReport(_model->parentSimulator()->name());
00166     tm->traceReport(_model->parentSimulator()->
00167 licenceManager()->showLicence());
00168     tm->traceReport(_model->parentSimulator()->
00169 licenceManager()->showLimits());
00170     // model infos
00171     tm->traceReport("Analyst Name: " + _info->analystName());
00172     tm->traceReport("Project Title: " + _info->projectTitle());
00173     tm->traceReport("Number of Replications: " + std::to_string(_info->
00174     numberOfReplications()));
00175     tm->traceReport("Replication Length: " + std::to_string(_info->
00176     replicationLength()) + " " + Util::StrTimeUnit(_info->
00177     replicationLengthTimeUnit()));
00178     //tm->traceReport(Util::TraceLevel::simulation, "");
00179     // model controls and responses
00180     std::string controls;
00181     for (std::list<SimulationControl*>::iterator it = _model->controls()->
00182 list()->begin(); it != _model->controls()->list()->end(); it++) {
00183         controls += (*it)->name() + "(" + (*it)->type() + ", ";
00184     }
00185     tm->traceReport("Simulation controls: " + controls);
00186     std::string responses;
00187     for (std::list<SimulationResponse*>::iterator it = _model->responses()->
00188 list()->begin(); it != _model->responses()->list()->end(); it++) {
00189         responses += (*it)->name() + "(" + (*it)->type() + ", ";
00190     }
00191     tm->traceReport("Simulation responses: " + responses);

```

```

00185 }
00186
00187 void ModelSimulation::_initSimulation() {
00188     _showSimulationHeader();
00189     _model->tracer()->trace(Util::TraceLevel::modelSimulationEvent
00190     , "Simulation of model \""
00191     + _info->name() + "\" is starting.");
00192     // copy all CStats and Counters (used in a replication) to CStats and counters for the whole simulation
00193     // \todo: Should not be CStats and Counters, but any element that generates report importation
00194     this->_statsCountersSimulation->clear();
00195     StatisticsCollector* cstat;
00196     StatisticsCollector* newCStatSimul = new
00197     StatisticsCollector(_model, _cte_stCountSimulNamePrefix + cstat->
00198     name(), cstat->getParent(), false);
00199     // this new CSat should NOT be inserted into the model
00200     // \todo: Counters in replication should be converted into CStats in simulation. Each value counted in
00201     // a replication should be added in a CStat for Stats.
00202     this->_statsCountersSimulation->insert(newCStatSimul);
00203     // copy all Counters (used in a replication) to Counters for the whole simulation
00204     // \todo: Counters in replication should be converted into CStats in simulation. Each value counted in
00205     // a replication should be added in a Counter for Stats.
00206     Counter* counter;
00207     List<ModelElement*>* counters = _model->elements()->
00208     elementList(Util::TypeOf<Counter>());
00209     for (std::list<ModelElement*>::iterator it = counters->list()->begin(); it != counters->
00210     list()->end(); it++) {
00211         counter = dynamic_cast<Counter*> ((*it));
00212         // adding a counter
00213         /*
00214         Counter* newCountSimul = new Counter(_cte_stCountSimulNamePrefix + counter->getName(),
00215         counter->getParent());
00216         this->_statsCountersSimulation->insert(newCountSimul);
00217         */
00218         // addin a cstat (to stat the counts)
00219         StatisticsCollector* newCStatSimul = new
00220         StatisticsCollector(_model, _cte_stCountSimulNamePrefix + counter->
00221         name(), counter->getParent(), false);
00222         this->_statsCountersSimulation->insert(newCStatSimul);
00223     }
00224 }
00225
00226 void ModelSimulation::_initReplication() {
00227     TraceManager* tm = _model->tracer();
00228     tm->trace(Util::TraceLevel::modelSimulationEvent, "");
00229     tm->trace(Util::TraceLevel::modelSimulationEvent, "
00230     Replication " + std::to_string(_currentReplicationNumber) + " of " + std::to_string(_info->
00231     number_of_replications()) + " is starting.");
00232     _model->futureEvents()->clear();
00233     _simulatedTime = 0.0;
00234     _pauseRequested = false;
00235
00236     //if (_currentReplicationNumber > 1) {
00237     // init all components between replications
00238     for (std::list<ModelComponent*>::iterator it = _model->components()->
00239     begin(); it != _model->components()->end(); it++) {
00240         ModelElement::InitBetweenReplications((*it));
00241     }
00242     // init all elements between replications
00243     ModelElement* element;
00244     std::string elementType;
00245     std::string* errorMessage = new std::string();
00246     std::list<std::string>* elementTypes = _model->elements()->
00247     elementClassnames();
00248     for (std::list<std::string>::iterator typeIt = elementTypes->begin(); typeIt != elementTypes->end();
00249     typeIt++) {
00250         elementType = (*typeIt);
00251         List<ModelElement*>* elements = _model->elements()->
00252         elementList(elementType);
00253         for (std::list<ModelElement*>::iterator it = elements->list()->begin(); it != elements->
00254         list()->end(); it++) {
00255             element = (*it);
00256             ModelElement::InitBetweenReplications(element);
00257         }
00258     }
00259 }
00260
00261 // insert first creation events
00262 SourceModelComponent *source;
00263 Entity *newEntity;

```

```

00256     Event *newEvent;
00257     double creationTime;
00258     unsigned int numToCreate;
00259     //std::list<ModelComponent*>* list = _model->getComponents()->list();
00260     for (std::list<ModelComponent*>::iterator it = _model->components()->
00261         begin(); it != _model->components()->end(); it++) {
00262         source = dynamic_cast<SourceModelComponent*> (*it);
00263         if (source != nullptr) {
00264             creationTime = source->firstCreation();
00265             numToCreate = source->entitiesPerCreation();
00266             for (unsigned int i = 1; i <= numToCreate; i++) {
00267                 newEntity = new Entity(_model);
00268                 newEntity->setEntityType(source->entityType());
00269                 newEvent = new Event(creationTime, newEntity, (*it));
00270                 _model->futureEvents()->insert(newEvent);
00271             }
00272             source->setEntitiesCreated(numToCreate);
00273         }
00274     }
00275     if (this->_initializeStatisticsBetweenReplications) {
00276         _initStatistics();
00277     }
00278 }
00279
00280 void ModelSimulation::_initStatistics() {
00281     StatisticsCollector* cstat;
00282     List<ModelElement*>* list = _model->elements()->
00283         elementList(Util::TypeOf<StatisticsCollector>());
00284     for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->
00285         list()->end(); it++) {
00286         cstat = (StatisticsCollector*) (*it);
00287         cstat->getStatistics()->getCollector()->clear();
00288         Counter* counter;
00289         list = _model->elements()->elementList(Util::TypeOf<Counter>());
00290         for (std::list<ModelElement*>::iterator it = list->list()->begin(); it != list->
00291             list()->end(); it++) {
00292             counter = (Counter*) (*it);
00293             counter->clear();
00294         }
00295     }
00296     void ModelSimulation::_checkWarmUpTime(Event* nextEvent) {
00297         double warmupTime = Util::TimeUnitConvert(_model->infos()->
00298             warmUpPeriodTimeUnit(), _model->infos()->
00299             replicationLengthTimeUnit());
00300         warmupTime *= _model->infos()->warmUpPeriod();
00301         if (warmupTime > 0.0 && _model->simulation()->simulatedTime() <= warmupTime &&
00302             nextEvent->time() > warmupTime) // warmupTime. Time to initStats
00303             _model->tracer()->trace(Util::TraceLevel::modelInternal, "
00304             Warmup time reached. Statistics are being reseted.");
00305             _initStatistics();
00306     }
00307
00308     void ModelSimulation::_stepSimulation() {
00309         // process one single event
00310         Event nextEvent;
00311         nextEvent = _model->futureEvents()->front();
00312         if (_model->infos()->warmUpPeriod() > 0.0)
00313             _checkWarmUpTime(nextEvent);
00314         if (nextEvent->time() <= _info->replicationLength()) {
00315             _model->futureEvents()->pop_front();
00316             _model->onEvents()->NotifyReplicationStepHandlers(new
00317                 SimulationEvent(_currentReplicationNumber, nullptr));
00318             _processEvent(nextEvent);
00319         } else {
00320             this->_simulatedTime = _model->infos()->replicationLength();
00321         }
00322
00323     void ModelSimulation::_processEvent(Event* event) {
00324         _model->tracer()->trace(Util::TraceLevel::modelSimulationEvent
00325             , "Processing event=" + event->show() + ")");
00326         _model->tracer()->trace(Util::TraceLevel::modelSimulationInternal
00327             , "Current Entity: " + event->entity()->show());
00328         this->_currentEntity = event->entity();
00329         this->_currentComponent = event->component();
00330         this->_currentInputNumber = event->componentInputNumber();
00331         _simulatedTime = event->time();
00332         _model->onEvents()->NotifyProcessEventHandlers(new
00333             SimulationEvent(_currentReplicationNumber, event));
00334         Util::IncIndent();
00335         try {
00336             //event->getComponent()->Execute(event->getEntity(), event->getComponent()); // Execute is static

```

```
00331     ModelComponent::Execute(event->entity(), event->component(), event->
00332         componentInputNumber());
00333     } catch (std::exception *e) {
00334         _model->tracer()->traceError(*e, "Error on processing event (" + event->
00335             show() + ")");
00336     }
00337 }
00338 void ModelSimulation::pause() {
00339 }
00340
00341 void ModelSimulation::step() {
00342     if (_simulationIsInitiated && _replicationIsInitiated) {
00343         if (!_isReplicationEndCondition()) {
00344             try {
00345                 this->stepSimulation();
00346             } catch (std::exception *e) {
00347                 _model->tracer()->traceError(*e, "Error on simulation step");
00348             }
00349     }
00350 }
00351
00352 }
00353
00354 void ModelSimulation::stop() {
00355 }
00356
00357 void ModelSimulation::restart() {
00358 }
00359
00360 void ModelSimulation::setPauseOnEvent(bool _pauseOnEvent) {
00361     this->_pauseOnEvent = _pauseOnEvent;
00362 }
00363
00364 bool ModelSimulation::isPauseOnEvent() const {
00365     return _pauseOnEvent;
00366 }
00367
00368 void ModelSimulation::setInitializeStatistics(bool
00369     _initializeStatistics) {
00370     this->_initializeStatisticsBetweenReplications = _initializeStatistics;
00371 }
00372 bool ModelSimulation::isInitializeStatistics() const {
00373     return _initializeStatisticsBetweenReplications;
00374 }
00375
00376 void ModelSimulation::setInitializeSystem(bool _initializeSystem) {
00377     this->_initializeSystem = _initializeSystem;
00378 }
00379
00380 bool ModelSimulation::isInitializeSystem() const {
00381     return _initializeSystem;
00382 }
00383
00384 void ModelSimulation::setStepByStep(bool _stepByStep) {
00385     this->_stepByStep = _stepByStep;
00386 }
00387
00388 bool ModelSimulation::isStepByStep() const {
00389     return _stepByStep;
00390 }
00391
00392 void ModelSimulation::setPauseOnReplication(bool _pauseOnReplication)
00393 {
00394     this->_pauseOnReplication = _pauseOnReplication;
00395 }
00396
00397 bool ModelSimulation::isPauseOnReplication() const {
00398     return _pauseOnReplication;
00399 }
00400
00401 double ModelSimulation::simulatedTime() const {
00402     return _simulatedTime;
00403 }
00404
00405 bool ModelSimulation::isRunning() const {
00406     return _running;
00407 }
00408
00409 unsigned int ModelSimulation::currentReplicationNumber() const {
00410     return _currentReplicationNumber;
00411 }
00412
00413 ModelComponent* ModelSimulation::currentComponent() const {
00414     return _currentComponent;
```

```

00414 }
00415
00416 Entity* ModelSimulation::currentEntity() const {
00417     return _currentEntity;
00418 }
00419
00420 SimulationReporter_if* ModelSimulation::reporter() const {
00421     return _simulationReporter;
00422 }
00423
00424 unsigned int ModelSimulation::currentInputNumber() const {
00425     return _currentInputNumber;
00426 }

```

7.271 ModelSimulation.h File Reference

```

#include "Event.h"
#include "Entity.h"
#include "ModelInfo.h"
#include "SimulationReporter_if.h"
#include "Counter.h"

```

Classes

- class [ModelSimulation](#)

7.272 ModelSimulation.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ModelSimulation.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 7 de Novembro de 2018, 18:04
00012 */
00013
00014 #ifndef MODELSIMULATION_H
00015 #define MODELSIMULATION_H
00016
00017 #include "Event.h"
00018 #include "Entity.h"
00019 #include "ModelInfo.h"
00020 #include "SimulationReporter_if.h"
00021 #include "Counter.h"
00022
00023 class Model;
00024
00029 class ModelSimulation {
00030 public:
00031     ModelSimulation(Model* model);
00032     virtual ~ModelSimulation() = default;
00033 public: // simulation control
00034     void start();
00035     void pause();
00036     void step();
00037     void stop();
00038     void restart();
00039 public:
00040     void setPauseOnEvent(bool _pauseOnEvent);
00041     bool isPauseOnEvent() const;
00042     void setStepByStep(bool _stepByStep);
00043     bool isStepByStep() const;
00044     void setInitializeStatistics(bool _initializeStatistics);
00045     bool isInitializeStatistics() const;
00046     void setInitializeSystem(bool _initializeSystem);
00047     bool isInitializeSystem() const;

```

```

00048     void setPauseOnReplication(bool _pauseBetweenReplications);
00049     bool isPauseOnReplication() const;
00050 public: // only gets
00051     double simulatedTime() const;
00052     bool isRunning() const;
00053     unsigned int currentReplicationNumber() const;
00054     ModelComponent* currentComponent() const;
00055     Entity* currentEntity() const;
00056     unsigned int currentInputNumber() const;
00057     SimulationReporter_if* reporter() const;
00058     /*
00059      * PRIVATE
00060      */
00061 private: // simulation control
00062     void _initSimulation();
00063     void _initReplication();
00064     void _initStatistics();
00065     void _checkWarmUpTime(Event* nextEvent);
00066     void _stepSimulation();
00067     void _processEvent(Event* event);
00068 private:
00069     bool _isReplicationEndCondition();
00070     void _actualizeSimulationStatistics();
00071     void _showSimulationHeader();
00072 private:
00073     double _simulatedTime = 0.0;
00074     // list of double double _breakOnTimes;
00075     // list of modules _breakOnModules;
00076     bool _stepByStep = false;
00077     bool _pauseOnReplication = false;
00078     bool _pauseOnEvent = false;
00079     bool _initializeStatisticsBetweenReplications = true;
00080     bool _initializeSystem = true;
00081     bool _running = false;
00082     bool _pauseRequested = false;
00083     bool _stopRequested = false;
00084     bool _simulationIsInitiated = false;
00085     bool _replicationIsInitiated = false;
00086 private:
00087     Entity* _currentEntity;
00088     ModelComponent* _currentComponent;
00089     unsigned int _currentInputNumber;
00090     SimulationReporter_if* _simulationReporter;
00091     unsigned int _currentReplicationNumber;
00092     List<ModelElement*>* _statsCountersSimulation = new
00093         List<ModelElement*>();
00094     //std::list<ModelElement*>* _countersSimulation = new std::list<ModelElement*>();
00095 private:
00096     Model* _model;
00097     ModelInfo* _info;
00098     const std::string _cte_stCountSimulNamePrefix = ""; //Simul.="";
00099 };
00100
00101 #endif /* MODELSIMULATION_H */
00102

```

7.273 Move.cpp File Reference

```
#include "Move.h"
#include "Model.h"
```

7.274 Move.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Move.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 13:17
00012 */

```

```

00013
00014 #include "Move.h"
00015
00016 #include "Model.h"
00017
00018 Move::Move(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Move>(), name) {
00020
00021
00022     std::string Move::show() {
00023         return ModelComponent::show() + "";
00024     }
00025
00026     ModelComponent* Move::LoadInstance(Model* model, std::map<std::string,
00027         std::string>* fields) {
00028         Move* newComponent = new Move(model);
00029         try {
00030             newComponent->_loadInstance(fields);
00031         } catch (const std::exception& e) {
00032         }
00033         return newComponent;
00034     }
00035
00036     void Move::_execute(Entity* entity) {
00037         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038             entity forward");
00039         this->_parentModel->sendEntityToComponent(entity, this->
00040             nextComponents()->frontConnection(), 0.0);
00041     }
00042
00043     bool Move::_loadInstance(std::map<std::string, std::string>* fields) {
00044         if (res) {
00045             //...
00046         }
00047         return res;
00048     }
00049     void Move::_initBetweenReplications() {
00050 }
00051
00052     std::map<std::string, std::string>* Move::_saveInstance() {
00053         std::map<std::string, std::string> fields = ModelComponent::_saveInstance
00054         ();
00055         //...
00056         return fields;
00057     }
00058     bool Move::_check(std::string* errorMessage) {
00059         bool resultAll = true;
00060         //...
00061         return resultAll;
00062     }
00063
00064     PluginInformation* Move::GetPluginInformation(){
00065         PluginInformation* info = new PluginInformation(Util::TypeOf<Move>(),
00066             &Move::LoadInstance);
00067         // ...
00068         return info;
00069     }
00070

```

7.275 Move.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Move](#)

7.276 Move.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Move.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:17
00012 */
00013
00014 #ifndef MOVE_H
00015 #define MOVE_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Move : public ModelComponent {
00020 public: // constructors
00021     Move(Model* model, std::string name = "");
00022     virtual ~Move() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040
00041 #endif /* MOVE_H */
00042

```

7.277 OLD_ODElement.cpp File Reference

```

#include "OLD_ODElement.h"
#include "Model.h"

```

7.278 OLD_ODElement.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 13 de Junho de 2019, 19:12
00012 */
00013
00014 #include "OLD_ODElement.h"
00015 #include "Model.h"
00016
00017 OLD_ODElement::OLD_ODElement(Model* model, std::string name) :
00018     ModelElement(model, Util::TypeOf<OLD_ODElement>(), name) {
00019     //_elems = elems;
00020 }
00021
00022 std::string OLD_ODElement::show() {

```

```

00023     std::string txt = ModelElement::show();
00024     unsigned short i=0;
00025     for (std::list<ODEfunction*>::iterator it = _ODEfunctions->list()->begin(); it != _ODEfunctions->
00026         list()->end(); it++) {
00027         txt += func["+std::to_string(i)+"]="+(*it)->expression+"\\",(func["+std::to_string(i)+"]+
00028         std::to_string((*it)->initialPoint)+"]="+std::to_string((*it)->initialValue)+"";
00029     }
00030
00031     double OLD_ODElement::solve() {
00032         return 0.0; // dummy
00033     }
00034 }
00035
00036
00037 void OLD_ODElement::setStepH(double _h) {
00038     this->stepH = _h;
00039 }
00040
00041 double OLD_ODElement::getStepH() const {
00042     return stepH;
00043 }
00044
00045 void OLD_ODElement::setEndTime(double _endTime) {
00046     this->endTime = _endTime;
00047 }
00048
00049 double OLD_ODElement::getEndTime() const {
00050     return endTime;
00051 }
00052
00053 List<ODEfunction*>* OLD_ODElement::getODEfunctions()
00054     const {
00055         return _ODEfunctions;
00056     }
00057 PluginInformation* OLD_ODElement::GetPluginInformation
00058     () {
00059         PluginInformation* info = new PluginInformation(
00060             Util::TypeOf<OLD_ODElement>(), &OLD_ODElement::LoadInstance);
00061         return info;
00062     }
00063
00064 ModelElement* OLD_ODElement::LoadInstance(
00065     Model* model, std::map<std::string, std::string>* fields) {
00066     OLD_ODElement* newElement = new OLD_ODElement(model);
00067     try {
00068         newElement->_loadInstance(fields);
00069     } catch (const std::exception& e) {
00070     }
00071     return newElement;
00072     bool OLD_ODElement::_loadInstance(std::map<std::string, std::string>* fields)
00073     {
00074         return ModelElement::_loadInstance(fields);
00075     }
00076     std::map<std::string, std::string>* OLD_ODElement::_saveInstance() {
00077         return ModelElement::_saveInstance();
00078     }
00079
00080     bool OLD_ODElement::_check(std::string* errorMessage) {
00081         bool result = true;
00082         unsigned short i=0;
00083         ODEfunction* func;
00084         for (std::list<ODEfunction*>::iterator it = _ODEfunctions->list()->begin(); it != _ODEfunctions->
00085             list()->end(); it++) {
00086             func = (*it);
00087             result &= _parentModel->checkExpression(func->
00088                 expression, "expression["+std::to_string(i++)+"]", errorMessage);
00089         }
00090     }
00091 }
00092 }
```

7.279 OLD_ODElement.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
```

Classes

- class [ODEfunction](#)
- class [OLD_ODEElement](#)

7.280 OLD_ODEElement.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ODE.h
00009  * Author: rlcancian
00010 *
00011 * Created on 13 de Junho de 2019, 19:12
00012 */
00013
00014 #ifndef OLD_ODEELEMENT_H
00015 #define OLD_ODEELEMENT_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020
00021 class ODEfunction {
00022 public:
00023
00024     ODEfunction(std::string expression, double initialPoint, double
00025     initialValue) {
00025         this->expression = expression;
00026         this->initialPoint = initialPoint;
00027         this->initialValue = initialValue;
00028     }
00029     std::string expression;
00030     double initialPoint;
00031     double initialValue;
00032 };
00033
00034 class OLD_ODElement : public ModelElement {
00035 public:
00036     OLD_ODElement(Model* model, std::string name(""));
00037     virtual ~OLD_ODElement() = default;
00038 public:
00039     virtual std::string show();
00040 public:
00041     static PluginInformation* GetPluginInformation();
00042     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>*
00043     fields);
00043 public:
00044     double solve();
00045     void setStepH(double _h);
00046     double getStepH() const;
00047     void setEndTime(double _endTime);
00048     double getEndTime() const;
00049     List<ODEfunction*>* getODEfunctions() const;
00050 protected: // must be overriden by derived classes
00051     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052     virtual std::map<std::string, std::string>* _saveInstance();
00053     virtual bool _check(std::string* errorMessage);
00054 private:
00055
00056 private:
00057     List<ODEfunction*>* _ODEfunctions = new List<ODEfunction*>();
00058     double _stepH = 0.1;
00059     double _endTime;
00060 };
00061
00062 #endif /* OLD_ODEELEMENT_H */
00063

```

7.281 OnEventManager.cpp File Reference

```
#include "OnEventManager.h"
```

7.282 OnEventManager.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    OnEventManager.cpp
00009  * Author:  rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 12:28
00012 */
00013
00014 #include "OnEventManager.h"
00015
00016 OnEventManager::OnEventManager() {
00017 }
00018
00019 void OnEventManager::addOnHandler(List<simulationEventHandler>* list,
00020     simulationEventHandler EventHandler) {
00021     if (list->find(EventHandler) == list->list()->end())
00022         list->insert(EventHandler);
00023 }
00024 void OnEventManager::addOnReplicationStartHandler(
00025     simulationEventHandler EventHandler) {
00026     _addOnHandler(_onReplicationStartHandlers, EventHandler);
00027 }
00028 void OnEventManager::addOnReplicationStepHandler(
00029     simulationEventHandler EventHandler) {
00030     _addOnHandler(_onReplicationStepHandlers, EventHandler);
00031 }
00032 void OnEventManager::addOnProcessEventHandler(
00033     simulationEventHandler EventHandler) {
00034     _addOnHandler(_onEntityMoveHandlers, EventHandler);
00035 }
00036 void OnEventManager::addOnEntityMoveHandler(
00037     simulationEventHandler EventHandler) {
00038     _addOnHandler(_onEntityMoveHandlers, EventHandler);
00039 }
00040 void OnEventManager::addOnReplicationEndHandler(
00041     simulationEventHandler EventHandler) {
00042     _addOnHandler(_onReplicationEndHandlers, EventHandler);
00043 }
00044 void OnEventManager::addOnSimulationStartHandler(
00045     simulationEventHandler EventHandler) {
00046     _addOnHandler(_onSimulationStartHandlers, EventHandler);
00047 }
00048 void OnEventManager::addOnSimulationEndHandler(
00049     simulationEventHandler EventHandler) {
00050     _addOnHandler(_onSimulationEndHandlers, EventHandler);
00051 }
00052 void OnEventManager::_NotifyHandlers(List<simulationEventHandler>* list,
00053     SimulationEvent* se) {
00054     for (std::list<simulationEventHandler>::iterator it = list->list()->begin(); it != list->
00055         list()->end(); it++) {
00056         (*it)(se);
00057     }
00058 }
00059 void OnEventManager::_NotifyHandlerMethods(List<simulationEventHandlerMethod>
00060     * list, SimulationEvent* se) {
00061     for (std::list<simulationEventHandlerMethod>::iterator it = list->list()->begin(); it != list->
00062         list()->end(); it++) {
00063         (*it)(se);
00064     }
00065 }
00066 void OnEventManager::NotifyReplicationStartHandlers(
00067     SimulationEvent* se) {
00068     this->_NotifyHandlers(this->_onReplicationStartHandlers, se);
00069 }
00070 void OnEventManager::NotifyReplicationStepHandlers(
00071     SimulationEvent* se) {
00072     this->_NotifyHandlers(this->_onReplicationStepHandlers, se);

```

```
00071 }
00072
00073 void OnEventManager::NotifyReplicationEndHandlers(
00074     SimulationEvent* se) {
00075     this->_NotifyHandlers(this->_onReplicationEndHandlers, se);
00076 }
00077 void OnEventManager::NotifyEntityMoveHandlers(
00078     SimulationEvent* se) {
00079     this->_NotifyHandlers(this->_onEntityMoveHandlers, se);
00080 }
00081 void OnEventManager::NotifyProcessEventHandlers(
00082     SimulationEvent* se) {
00083     this->_NotifyHandlers(this->_onProcessEventHandlers, se);
00084     this->_NotifyHandlerMethods(this->_onProcessEventHandlerMethods, se);
00085 }
00086 void OnEventManager::NotifySimulationStartHandlers(
00087     SimulationEvent* se) {
00088     this->_NotifyHandlers(this->_onSimulationStartHandlers, se);
00089 }
00090 void OnEventManager::NotifySimulationEndHandlers(
00091     SimulationEvent* se) {
00092     this->_NotifyHandlers(this->_onSimulationEndHandlers, se);
00093 }
```

7.283 OnEventManager.h File Reference

```
#include "List.h"
#include "Event.h"
```

Classes

- class [SimulationEvent](#)
- class [OnEventManager](#)

Typedefs

- [typedef void\(* simulationEventHandler\) \(SimulationEvent *\)](#)
- [typedef std::function< void\(SimulationEvent *\) > simulationEventHandlerMethod](#)

7.283.1 Typedef Documentation

7.283.1.1 [simulationEventHandler](#)

```
typedef void(* simulationEventHandler) (SimulationEvent *)
```

Definition at line 48 of file [OnEventManager.h](#).

7.283.1.2 [simulationEventHandlerMethod](#)

```
typedef std::function<void(SimulationEvent*) > simulationEventHandlerMethod
```

Definition at line 50 of file [OnEventManager.h](#).

7.284 OnEventManager.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: OnEventManager.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 12:28
00012 */
00013
00014 #ifndef ONEVENTMANAGER_H
00015 #define ONEVENTMANAGER_H
00016
00017 #include "List.h"
00018 #include "Event.h"
00019
00020 /* \todo: To implement as item (1) for DS3
00021  * used to get and set values no matter the class (for process analyser)
00022  * should be a wait to invoke a getter or setter no matter the class (a pointer to a member function
00023  * without specifying the class
00024 //typedef double (*memberFunctionGetDoubleVarHandler)(); //template<> ... typedef double
00025 //    (T::*getDoubleVarHandler)() or something like that
00026 //typedef void (*memberFunctionSetDoubleVarHandler)(double);
00027
00028 class SimulationEvent {
00029 public:
00030     SimulationEvent(unsigned int replicationNumber, Event* event) {
00031         _replicationNumber = replicationNumber;
00032         _event = event;
00033     }
00034 public:
00035
00036     unsigned int getReplicationNumber() const {
00037         return _replicationNumber;
00038     }
00039
00040     Event* getEventProcessed() const {
00041         return _event;
00042     }
00043 private:
00044     unsigned int _replicationNumber;
00045     Event* _event;
00046 };
00047
00048 typedef void (*simulationEventHandler)(SimulationEvent*);
00049 // for handlers that are class members (methods)
00050 typedef std::function<void(SimulationEvent*)> simulationEventHandlerMethod;
00051
00052 class OnEventManager {
00053 public:
00054     OnEventManager();
00055     virtual ~OnEventManager() = default;
00056 public: // event listeners (handlers)
00057     void addOnReplicationStartHandler(simulationEventHandler EventHandler);
00058     void addOnReplicationStepHandler(simulationEventHandler EventHandler);
00059     void addOnReplicationEndHandler(simulationEventHandler EventHandler);
00060     void addOnProcessEventHandler(simulationEventHandler EventHandler);
00061     void addOnEntityMoveHandler(simulationEventHandler EventHandler);
00062     void addOnSimulationStartHandler(simulationEventHandler EventHandler);
00063     void addOnSimulationEndHandler(simulationEventHandler EventHandler);
00064     void addOnEntityRemoveHandler(simulationEventHandler EventHandler);
00065
00066     // for handlers that are class members (methods)
00067     template<typename Class> void addOnProcessEventHandler(Class * object, void (Class::*function) (
00068         SimulationEvent*));
00069     // \todo: ...
00070
00071 public:
00072     void NotifyReplicationStartHandlers(SimulationEvent* se);
00073     void NotifyReplicationStepHandlers(SimulationEvent* se);
00074     void NotifyReplicationEndHandlers(SimulationEvent* se);
00075     void NotifyProcessEventHandlers(SimulationEvent* se);
00076     void NotifyEntityMoveHandlers(SimulationEvent* se);
00077     void NotifySimulationStartHandlers(SimulationEvent* se);
00078     void NotifySimulationEndHandlers(SimulationEvent* se);
00079
00080 private:
00081     void _NotifyHandlers(List<simulationEventHandler*>* list,
00082         SimulationEvent* se);
00083     void _NotifyHandlerMethods(List<simulationEventHandlerMethod*>* list,
00084         SimulationEvent* se);
00085     void _addOnHandler(List<simulationEventHandler*>* list,
00086

```

```

    simulationEventHandler EventHandler);
00084 private: // events listener
00085     List<simulationEventHandler>* _onReplicationStartHandlers = new
00086     List<simulationEventHandler>() ;
00087     List<simulationEventHandler>* _onReplicationStepHandlers = new
00088     List<simulationEventHandler>() ;
00089     List<simulationEventHandler>* _onReplicationEndHandlers = new
00090     List<simulationEventHandler>() ;
00091     List<simulationEventHandler>* _onProcessEventHandlers = new
00092     List<simulationEventHandler>() ;
00093     List<simulationEventHandler>* _onEntityMoveHandlers = new
00094     List<simulationEventHandler>() ;
00095     List<simulationEventHandler>* _onSimulationStartHandlers = new
00096     List<simulationEventHandler>() ;
00097     List<simulationEventHandler>* _onSimulationEndHandlers = new
00098     List<simulationEventHandler>() ;
00099     // for handlers that are class members (methods)
00100     List<simulationEventHandlerMethod>* _onProcessEventHandlerMethods =
00101     new List<simulationEventHandlerMethod>();
00102     // \todo: ...
00103 };
00104
00105 // implementation for template methods
00106
00107 template<typename Class> void OnEventManager::addOnProcessEventHandler
00108 (Class * object, void (Class::*function)(SimulationEvent*)) {
00109     simulationEventHandlerMethod handlerMethod = std::bind(function, object,
00110     std::placeholders::_1);
00111     // \todo: if handlerMethod already insert, should not insert it again. Problem to solve <...> for
00112     // function
00113     //if (_onProcessEventHandlerMethods->find(handlerMethod) ==
00114     _onProcessEventHandlerMethods->list()->end())
00115         this->_onProcessEventHandlerMethods->insert(handlerMethod);
00116     // trying unique to solve the issue
00117     //this->_onProcessEventHandlerMethods->list()->unique(); // does not work
00118     // \todo: probably to override == operator for type simulationEventHandlerMethod
00119 }
00120
00121 // ...
00122
00123 #endif /* ONEVENTMANAGER_H */
00124

```

7.285 Parser_if.h File Reference

```
#include <string>
```

Classes

- class [Parser_if](#)

7.286 Parser_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Parser_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 23 de Agosto de 2018, 15:57
00012 */
00013
00014 #ifndef PARSER_IF_H
00015 #define PARSER_IF_H
00016
00017 #include <string>
00018
00019 class Parser_if {
00020 public:

```

```

00021     virtual double parse(const std::string expression) = 0; // may throw exception
00022     virtual double parse(const std::string expression, bool* success, std::string* errorMessage) = 0;
00023     virtual std::string* getErrorMessage() = 0; // to get error message in the case of
00024     thrown exception
00025     // ...
00026 };
00027 #endif /* PARSER_IF_H */
00028

```

7.287 ParserChangesInformation.cpp File Reference

```
#include "ParserChangesInformation.h"
```

7.288 ParserChangesInformation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ParserChangesInformation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 20:42
00012 */
00013
00014 #include "ParserChangesInformation.h"
00015
00016 ParserChangesInformation::ParserChangesInformation() {
00017 }
00018
00019
00020 std::list<ParserChangesInformation::ParserChangesInformation::lexicalelement*>*
00021     ParserChangesInformation::getLexicalElementToRemove()
00022     const {
00023         return _lexicalElementToRemove;
00024     }
00025
00026 std::list<ParserChangesInformation::lexicalelement*>*
00027     ParserChangesInformation::getLexicalElementToAdd() const {
00028         return _lexicalElementToAdd;
00029     }
00030
00031 std::list<ParserChangesInformation::lexinclude*>*
00032     ParserChangesInformation::getLexIncludeToRemove() const {
00033         return _lexIncludeToRemove;
00034     }
00035
00036 std::list<ParserChangesInformation::production*>*
00037     ParserChangesInformation::getProductionToRemove() const {
00038         return _productionToRemove;
00039     }
00040
00041 std::list<ParserChangesInformation::production*>*
00042     ParserChangesInformation::getProductionToAdd() const {
00043         return _productionToAdd;
00044     }
00045
00046 std::list<ParserChangesInformation::typeobj*>*
00047     ParserChangesInformation::getTypeobjToRemove() const {
00048         return _typeobjToRemove;
00049     }
00050

```

```

00051
00052     std::list<ParserChangesInformation::token*>*
00053         ParserChangesInformation::getTokensToRemove() const {
00054             return _tokensToRemove;
00055
00056     std::list<ParserChangesInformation::token*>*
00057         ParserChangesInformation::getTokensToAdd() const {
00058             return _tokensToAdd;
00059

```

7.289 ParserChangesInformation.h File Reference

```
#include <string>
#include <list>
```

Classes

- class [ParserChangesInformation](#)

7.290 ParserChangesInformation.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ParserChangesInformation.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 20:42
00012 */
00013
00014 #ifndef PARSERCHANGESINFORMATION_H
00015 #define PARSERCHANGESINFORMATION_H
00016
00017 #include <string>
00018 #include <list>
00019
00020 class ParserChangesInformation {
00021 public:
00022     typedef std::string token;
00023     typedef std::string typeobj;
00024     typedef std::string lexinclude;
00025     typedef std::pair<std::string, std::string> production;
00026     typedef std::pair<std::string, std::string> lexiclelement;
00027 public:
00028     ParserChangesInformation();
00029     virtual ~ParserChangesInformation() = default;
00030
00031 public: // gets and sets
00032     std::list<ParserChangesInformation::lexiclelement*>*
00033         getLexicalElementToRemove() const;
00034     std::list<ParserChangesInformation::lexiclelement*>* getLexicalElementToAdd() const;
00035     std::list<ParserChangesInformation::lexinclude*>* getLexIncludeToRemove() const;
00036     std::list<ParserChangesInformation::lexinclude*>* getLexIncludeToAdd() const;
00037     std::list<ParserChangesInformation::production*>* getProductionToRemove() const;
00038     std::list<ParserChangesInformation::production*>* getProductionToAdd() const;
00039     std::list<ParserChangesInformation::typeobj*>* getTypeobjToRemove() const;
00040     std::list<ParserChangesInformation::typeobj*>* getTypeobjToAdd() const;
00041     std::list<ParserChangesInformation::token*>* getTokensToRemove() const;
00042     std::list<ParserChangesInformation::token*>* getTokensToAdd() const;
00043     std::list<token*>* _tokensToAdd = new std::list<token*>();
00044     std::list<token*>* _tokensToRemove = new std::list<token*>();
00045     std::list<typeobj*>* _typeobjToAdd = new std::list<typeobj*>();
00046     std::list<typeobj*>* _typeobjToRemove = new std::list<typeobj*>();
00047     std::list<production*>* _productionToAdd = new std::list<production*>();

```

```

00048     std::list<production*>* _productionToRemove = new std::list<production*>();
00049     std::list<lexinclude*>* _lexIncludeToAdd = new std::list<lexinclude*>();
00050     std::list<lexinclude*>* _lexIncludeToRemove = new std::list<lexinclude*>();
00051     std::list<lexicalelement*>* _lexicalElementToAdd = new std::list<lexicalelement*>();
00052     std::list<lexicalelement*>* _lexicalElementToRemove = new std::list<lexicalelement*>();
00053
00054 };
00055
00056 #endif /* PARSERCHANGESINFORMATION_H */
00057

```

7.291 ParserDefaultImpl1.cpp File Reference

```
#include "ParserDefaultImpl1.h"
```

7.292 ParserDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ParserDefaultImpl1.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 01:25
00012 */
00013
00014 #include "ParserDefaultImpl1.h"
00015
00016 ParserDefaultImpl1::ParserDefaultImpl1(
    Model* model) {
00017     _model = model;
00018 }
00019
00020
00021 double ParserDefaultImpl1::parse(const std::string expression) { // may throw
    exception
00022     double result = std::atof(expression.c_str()); // change by a real parser
00023     return result;
00024 }
00025
00026 std::string* ParserDefaultImpl1::getErrorMessage() {
00027     std::string* errorMsg = new std::string();
00028     return errorMsg; /* @ \todo: */
00029 }
00030
00031 double ParserDefaultImpl1::parse(const std::string expression, bool* success,
    std::string* errorMessage) {
00032     try {
00033         double result = this->parse(expression);
00034         std::string temp(""); /* \todo: CHECK SCOPE OF VARIABLE */
00035         errorMessage = &temp;
00036         *success = true;
00037         return result;
00038     } catch (...) {
00039         std::string temp("Error parsing...");
00040         errorMessage = &temp;
00041         *success = false;
00042         return 0.0;
00043     }
00044 }
00045

```

7.293 ParserDefaultImpl1.h File Reference

```
#include "Parser_if.h"
```

Classes

- class [ParserDefaultImpl1](#)

7.294 ParserDefaultImpl1.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: ParserDefaultImpl1.h
00008 * Author: rafael.luiz.cancian
00009 *
00010 *
00011 * Created on 2 de Agosto de 2018, 01:25
00012 */
00013
00014 #ifndef PARSERDEFAULTIMPL1_H
00015 #define PARSERDEFAULTIMPL1_H
00016
00017 #include "Parser_if.h"
00018
00019 class Model;
00020
00021 class ParserDefaultImpl1 : public Parser_if {
00022 public:
00023     ParserDefaultImpl1(Model* model);
00024     virtual ~ParserDefaultImpl1() = default;
00025 private:
00026     Model* _model;
00027 };
00028
00029 #endif /* PARSERDEFAULTIMPL1_H */
00030
00031
00032
00033
00034

```

7.295 ParserManager.cpp File Reference

```
#include "ParserManager.h"
```

7.296 ParserManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ParserManager.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 20:34
00012 */
00013
00014 #include "ParserManager.h"
00015
00016 ParserManager::ParserManager() {
00017 }
00018
00019

```

7.297 ParserManager.h File Reference

```
#include "ParserChangesInformation.h"
```

Classes

- class [ParserManager](#)
- struct [ParserManager::NewParser](#)
- struct [ParserManager::GenerateNewParserResult](#)

7.298 ParserManager.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ParserManager.h
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 20:34
00012 */
00013
00014 #ifndef PARSEMANAGER_H
00015 #define PARSEMANAGER_H
00016
00017 #include "ParserChangesInformation.h"
00018
00019 class ParserManager {
00020 public:
00021     struct NewParser {
00022         std::string bisonFilename;
00023         std::string flexFilename;
00024         std::string compiledParserFilename;
00025     };
00026     struct GenerateNewParserResult {
00027         bool result;
00028         std::string bisonMessages;
00029         std::string lexMessages;
00030         std::string compilationMessages;
00031         NewParser newParser;
00032     };
00033 };
00034 public:
00035     ParserManager();
00036     virtual ~ParserManager() = default;
00037 public:
00038     ParserManager::GenerateNewParserResult
00039         generateNewParser(ParserChangesInformation* changes);
00040     bool connectNewParser(ParserManager::NewParser newParser);
00041 };
00042
00043 #endif /* PARSEMANAGER_H */
```

7.299 PickStation.cpp File Reference

```
#include "PickStation.h"
#include "Model.h"
```

7.300 PickStation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: PickStation.cpp
00008 * Author: rlcancian
00009 *
00010 *
00011 * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #include "PickStation.h"
00015
00016 #include "Model.h"
00017
00018 PickStation::PickStation(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<PickStation>(), name) {
00019 }
00020
00021
00022 std::string PickStation::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* PickStation::LoadInstance(
    Model* model, std::map<std::string, std::string>* fields) {
00027     PickStation* newComponent = new PickStation(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031         }
00032     return newComponent;
00033 }
00034
00035
00036 void PickStation::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00039 }
00040
00041 bool PickStation::_loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelComponent::_loadInstance(fields);
00043     if (res) {
00044         //...
00045     }
00046     return res;
00047 }
00048
00049 void PickStation::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* PickStation::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
();
00054     //...
00055     return fields;
00056 }
00057
00058 bool PickStation::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* PickStation::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(
        Util::TypeOf<PickStation>(), &PickStation::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069

```

7.301 PickStation.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [PickStation](#)

7.302 PickStation.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    PickStation.h
00009  * Author:  rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #ifndef PICKSTATION_H
00015 #define PICKSTATION_H
00016
00017 #include "ModelComponent.h"
00018
00062 class PickStation : public ModelComponent {
00063 public: // constructors
00064     PickStation(Model* model, std::string name="");
00065     virtual ~PickStation() = default;
00066 public: // virtual
00067     virtual std::string show();
00068 public: // static
00069     static PluginInformation* GetPluginInformation();
00070     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00071                                         std::string>* fields);
00071 protected: // virtual
00072     virtual void _execute(Entity* entity);
00073     virtual void _initBetweenReplications();
00074     virtual bool _loadInstance(std::map<std::string, std::string>* _saveInstance());
00075     virtual std::map<std::string, std::string>* _check(std::string* errorMessage);
00076 private: // methods
00077 private: // attributes 1:1
00079 private: // attributes 1:n
00080 };
00081
00082
00083 #endif /* PICKSTATION_H */
00084

```

7.303 PickUp.cpp File Reference

```
#include "PickUp.h"
#include "Model.h"
```

7.304 PickUp.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    PickUp.cpp
00009  * Author:  rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #include "PickUp.h"
00015

```

```

00016 #include "Model.h"
00017
00018 PickUp::PickUp(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<PickUp>(), name) {
00019 }
00020
00021
00022 std::string PickUp::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* PickUp::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00027     PickUp* newComponent = new PickUp(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031
00032     }
00033     return newComponent;
00034 }
00035
00036 void PickUp::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00039 }
00040
00041 bool PickUp::_loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelComponent::_loadInstance(fields);
00043     if (res) {
00044         //...
00045     }
00046     return res;
00047 }
00048
00049 void PickUp::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* PickUp::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
();
00054     //...
00055     return fields;
00056 }
00057
00058 bool PickUp::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* PickUp::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<PickUp>(
), &PickUp::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069
00070

```

7.305 PickUp.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [PickUp](#)

7.306 PickUp.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    PickUp.h
00009  * Author:  rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #ifndef PICKUP_H
00015 #define PICKUP_H
00016
00017 #include "ModelComponent.h"
00018
00019 class PickUp : public ModelComponent {
00020 public: // constructors
00021     PickUp(Model* model, std::string name="");
00022     virtual ~PickUp() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059 #endif /* PICKUP_H */
00060

```

7.307 Plugin.cpp File Reference

```

#include "Plugin.h"
#include "Model.h"
#include "SourceModelComponent.h"
#include "SinkModelComponent.h"
#include "Assign.h"
#include <assert.h>

```

7.308 Plugin.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Plugin.cpp
00009  * Author:  rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:58
00012 */
00013
00014 #include "Plugin.h"
00015 #include "Model.h"
00016 #include "SourceModelComponent.h"
00017 #include "SinkModelComponent.h"

```

```

00018 #include "Assign.h"
00019 #include <assert.h>
00020
00021 Plugin::Plugin(StaticGetPluginInformation getInformation) {
00022     this->_StatMethodGetInformation = getInformation;
00023     try {
00024         PluginInformation* infos = _StatMethodGetInformation();
00025         this->_pluginInfo = infos;
00026         this->_isValidPlugin = true;
00027     } catch (...) {
00028         this->_isValidPlugin = false;
00029     }
00030 }
00031
00032 PluginInformation* Plugin::pluginInfo() const {
00033     return _pluginInfo;
00034 }
00035
00036 bool Plugin::isValidPlugin() const {
00037     return _isValidPlugin;
00038 }
00039
00040 //Plugin::Plugin(std::string pluginTypename, bool source, bool drain) {
00041 //    __fullfilename = fullfilename;
00042 //    __source = source;
00043 //    __drain = drain;
00044 //}
00045
00046 ModelElement* Plugin::loadNew(Model* model, std::map<std::string,
00047     std::string>* fields) {
00048     if (this->_pluginInfo->isComponent()) {
00049         return _loadNewComponent(model, fields);
00050     } else {
00051         return _loadNewElement(model, fields);
00052     }
00053 }
00054 bool Plugin::loadAndInsertNew(Model* model, std::map<std::string, std::string>
00055     * fields) {
00056     if (this->_pluginInfo->isComponent()) {
00057         ModelComponent* newComp = _loadNewComponent(model, fields);
00058         if (newComp != nullptr) {
00059             //model->getTraceManager()->trace(newComp->show());
00060             return true; //model->components()->insert(newComp);
00061         } else {
00062             ModelElement* newElem = _loadNewElement(model, fields);
00063             if (newElem != nullptr) {
00064                 //model->getTraceManager()->trace(newElem->show());
00065                 return true; //model->elements()->insert(this->_pluginInfo->pluginTypename(), newElem);
00066             }
00067         }
00068         return false;
00069     }
00070 }
00071 ModelComponent* Plugin::_loadNewComponent(Model* model, std::map<std::string,
00072     std::string>* fields) {
00073     //return this->_pluginInfo->loader(model, fields);
00074     StaticLoaderComponentInstance loader = this->_pluginInfo->
00075         componentLoader();
00076     ModelComponent* newElementOrComponent = loader(model, fields);
00077     return newElementOrComponent;
00078 }
00079 ModelElement* Plugin::_loadNewElement(Model* model, std::map<std::string, std::string>
00080     * fields) {
00081     StaticLoaderElementInstance loader = this->_pluginInfo->
00082         elementLoader();
00083     ModelElement* newElementOrComponent = loader(model, fields);
00084     return newElementOrComponent;
00085 }
00086 }
```

7.309 Plugin.h File Reference

```
#include "Util.h"
#include <string>
#include <functional>
#include <list>
#include "PluginInformation.h"
```

Classes

- class [Plugin](#)

7.310 Plugin.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Plugin.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:58
00012 */
00013
00014 #ifndef PLUGIN_H
00015 #define PLUGIN_H
00016
00017 #include "Util.h"
00018 #include <string>
00019 #include <functional>
00020 #include <list>
00021
00022 #include "PluginInformation.h"
00023
00024 class Plugin {
00025 public:
00026     Plugin(std::string filename_so_dll);
00027     Plugin(StaticGetPluginInformation getInformation); // temporary. Just
00028     while compiled together
00029     virtual ~Plugin() = default;
00030 public:
00031     bool isIsValidPlugin() const;
00032     PluginInformation* pluginInfo() const;
00033 public:
00034     ModelElement* loadNew(Model* model, std::map<std::string, std::string>* fields)
00035     ,
00036     bool loadAndInsertNew(Model* model, std::map<std::string, std::string>* fields);
00037 private:
00038     ModelComponent* _loadNewComponent(Model* model, std::map<std::string, std::string>* fields);
00039     ModelElement* _loadNewElement(Model* model, std::map<std::string, std::string>* fields
00040 );
00041 private: // read only
00042     bool _isValidPlugin;
00043     PluginInformation* _pluginInfo;
00044 private:
00045     StaticGetPluginInformation _StatMethodGetInformation;
00046 };
00047 */
00048
00049 #endif /* PLUGIN_H */
00050

```

7.311 PluginConnector_if.h File Reference

```
#include <string>
#include "Plugin.h"
```

Classes

- class [PluginConnector_if](#)

7.312 PluginConnector_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: PluginConnector_if.h
00009  * Author: rlcancian
00010 *
00011  * Created on 9 de Setembro de 2019, 19:17
00012 */
00013
00014 #ifndef PLUGINCONNECTOR_IF_H
00015 #define PLUGINCONNECTOR_IF_H
00016
00017 #include<string>
00018 #include "Plugin.h"
00019
00020 class PluginConnector_if {
00021 public:
00022     virtual Plugin* check(const std::string dynamicLibraryFilename) = 0;
00023     virtual Plugin* connect(const std::string dynamicLibraryFilename) = 0;
00024     virtual bool disconnect(const std::string dynamicLibraryFilename) = 0;
00025     virtual bool disconnect(Plugin* plugin) = 0;
00026 };
00027
00028 #endif /* PLUGINCONNECTOR_IF_H */
00029

```

7.313 PluginConnectorDummyImpl1.cpp File Reference

```

#include "PluginConnectorDummyImpl1.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "Record.h"
#include "Decide.h"
#include "Dummy.h"
#include "Route.h"
#include "Enter.h"
#include "Leave.h"
#include "Write.h"
#include "LSODE.h"
#include "MarkovChain.h"
#include "EntityType.h"
#include "Attribute.h"
#include "Variable.h"
#include "ProbDistrib.h"
#include "EntityGroup.h"
#include "Station.h"
#include "Formula.h"
#include "Set.h"
#include "OLD_ODElement.h"
#include "Util.h"

```

7.314 PluginConnectorDummyImpl1.cpp

```
00001 /*
```

```

00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: PluginConnectorDummyImpl1.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 19:24
00012 */
00013
00014 #include "PluginConnectorDummyImpl1.h"
00015
00016 // Model Components
00017 #include "Create.h"
00018 #include "Delay.h"
00019 #include "Dispose.h"
00020 #include "Seize.h"
00021 #include "Release.h"
00022 #include "Assign.h"
00023 #include "Record.h"
00024 #include "Decide.h"
00025 #include "Dummy.h"
00026 #include "Route.h"
00027 #include "Enter.h"
00028 #include "Leave.h"
00029 #include "Write.h"
00030 #include "LSODE.h"
00031 #include "MarkovChain.h"
00032
00033 // Model elements
00034 #include "EntityType.h"
00035 #include "Attribute.h"
00036 #include "Variable.h"
00037 #include "ProbDistrib.h"
00038 #include "EntityGroup.h"
00039 #include "Station.h"
00040 #include "Formula.h"
00041 #include "Set.h"
00042 #include "OLD_ODElement.h"
00043
00044 #include "Util.h"
00045
00046 PluginConnectorDummyImpl1::PluginConnectorDummyImpl1()
{
00047 }
00048
00049
00050 Plugin* PluginConnectorDummyImpl1::check(const std::string
    dynamicLibraryFilename) {
00051     return nullptr;
00052 }
00053
00054 Plugin* PluginConnectorDummyImpl1::connect(const std::string
    dynamicLibraryFilename) {
00055     std::string fn = getFileName(dynamicLibraryFilename);
00056     StaticGetPluginInformation GetInfo = nullptr;
00057     Plugin* pluginResult = nullptr;
00058     // model elements
00059     if (fn == "attribute.so")
00060         GetInfo = &Attribute::GetPluginInformation;
00061     else if (fn == "assign.so")
00062         GetInfo = &Assign::GetPluginInformation;
00063     else if (fn == "counter.so")
00064         GetInfo = &Counter::GetPluginInformation;
00065     else if (fn == "entitygroup.so")
00066         GetInfo = &EntityGroup::GetPluginInformation;
00067     else if (fn == "entitytype.so")
00068         GetInfo = &EntityType::GetPluginInformation;
00069     else if (fn == "formula.so")
00070         GetInfo = &Formula::GetPluginInformation;
00071     else if (fn == "ode.so")
00072         GetInfo = &OLD_ODElement::GetPluginInformation;
00073     else if (fn == "queue.so")
00074         GetInfo = &Queue::GetPluginInformation;
00075     else if (fn == "resource.so")
00076         GetInfo = &Resource::GetPluginInformation;
00077     else if (fn == "set.so")
00078         GetInfo = &Set::GetPluginInformation;
00079     else if (fn == "statisticscollector.so")
00080         GetInfo = &StatisticsCollector::GetPluginInformation;
00081     else if (fn == "station.so")
00082         GetInfo = &Station::GetPluginInformation;
00083     else if (fn == "variable.so")
00084         GetInfo = &Variable::GetPluginInformation;
00085     // model components

```

```

00086     else if (fn == "create.so")
00087         GetInfo = &Create::GetPluginInformation;
00088     else if (fn == "write.so")
00089         GetInfo = &Write::GetPluginInformation;
00090     else if (fn == "decide.so")
00091         GetInfo = &Decide::GetPluginInformation;
00092     else if (fn == "delay.so")
00093         GetInfo = &Delay::GetPluginInformation;
00094     else if (fn == "dispose.so")
00095         GetInfo = &Dispose::GetPluginInformation;
00096     else if (fn == "dummy.so")
00097         GetInfo = &Dummy::GetPluginInformation;
00098     else if (fn == "record.so")
00099         GetInfo = &Record::GetPluginInformation;
00100    else if (fn == "release.so")
00101    GetInfo = &Release::GetPluginInformation;
00102    else if (fn == "seize.so")
00103    GetInfo = &Seize::GetPluginInformation;
00104    else if (fn == "route.so")
00105    GetInfo = &Route::GetPluginInformation;
00106    else if (fn == "enter.so")
00107    GetInfo = &Enter::GetPluginInformation;
00108    else if (fn == "leave.so")
00109    GetInfo = &Leave::GetPluginInformation;
00110    else if (fn == "lsode.so")
00111    GetInfo = &LSODE::GetPluginInformation;
00112    else if (fn == "markovchain.so")
00113    GetInfo = &MarkovChain::GetPluginInformation;
00114 //else if (fn=="")
00115
00116     if (GetInfo != nullptr) {
00117         pluginResult = new Plugin(GetInfo);
00118     }
00119     return pluginResult;
00120 }
00121
00122 bool PluginConnectorDummyImpl1::disconnect(const std::string
00123     dynamicLibraryFilename) {
00124     return true;
00125 }
00126 bool PluginConnectorDummyImpl1::disconnect(
00127     Plugin* plugin) {
00128     return true;
00129 }
```

7.315 PluginConnectorDummyImpl1.h File Reference

```
#include "PluginConnector_if.h"
```

Classes

- class [PluginConnectorDummyImpl1](#)

7.316 PluginConnectorDummyImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: PluginConnectorDummyImpl1.h
00009  * Author: rlcancian
00010 *
00011  * Created on 9 de Setembro de 2019, 19:24
00012 */
00013
00014 #ifndef PLUGINCONNECTORDUMMYIMPL1_H
00015 #define PLUGINCONNECTORDUMMYIMPL1_H
00016
00017 #include "PluginConnector_if.h"
```

```

00018
00019 class PluginConnectorDummyImpl : public
00020     PluginConnector_if {
00021     public:
00022         PluginConnectorDummyImpl();
00023         virtual ~PluginConnectorDummyImpl() = default;
00024     public:
00025         virtual Plugin* check(const std::string dynamicLibraryFilename);
00026         virtual Plugin* connect(const std::string dynamicLibraryFilename);
00027         virtual bool disconnect(const std::string dynamicLibraryFilename);
00028         virtual bool disconnect(Plugin* plugin);
00029     private:
00030 };
00031
00032 #endif /* PLUGINCONNECTORDUMMYIMPL_H */
00033

```

7.317 PluginInformation.cpp File Reference

```
#include "PluginInformation.h"
```

7.318 PluginInformation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PluginInformation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 20:02
00012 */
00013
00014 #include "PluginInformation.h"
00015
00016
00017     PluginInformation::PluginInformation(std::string pluginTypename,
00018     StaticLoaderComponentInstance componentloader) {
00019     this->_componentloader = componentloader;
00020     this->elementloader = nullptr;
00021     this->_isComponent = true;
00022     this->_pluginTypename = pluginTypename;
00023 }
00024     PluginInformation::PluginInformation(std::string
00025     pluginTypename, StaticLoaderElementInstance
00026     elementloader) {
00027     this->_componentloader = nullptr;
00028     this->elementloader = elementloader;
00029     this->_isComponent = false;
00030     this->_pluginTypename = pluginTypename;
00031
00032     StaticLoaderElementInstance
00033     PluginInformation::elementloader() const {
00034         return _elementloader;
00035     }
00036
00037     StaticLoaderComponentInstance
00038     PluginInformation::componentLoader() const {
00039         return _componentloader;
00040     }
00041
00042     bool PluginInformation::isGenerateReport() const {
00043         return _generateReport;
00044     }
00045
00046     bool PluginInformation::isComponent() const {
00047         return _isComponent;
00048     }
00049
00050     bool PluginInformation::isSendTransfer() const {

```

```
00048     return _sendTransfer;
00049 }
00050
00051     bool PluginInformation::isReceiveTransfer() const {
00052         return _receiveTransfer;
00053     }
00054
00055     bool PluginInformation::isSink() const {
00056         return _isSink;
00057     }
00058
00059     bool PluginInformation::isSource() const {
00060         return _isSource;
00061     }
00062
00063     std::string PluginInformation::observation() const {
00064         return _observation;
00065     }
00066
00067     std::string PluginInformation::version() const {
00068         return _version;
00069     }
00070
00071     std::string PluginInformation::date() const {
00072         return _date;
00073     }
00074
00075     std::string PluginInformation::author() const {
00076         return _author;
00077     }
00078
00079     std::string PluginInformation::pluginTypename() const {
00080         return _pluginTypename;
00081     }
00082
00083     void PluginInformation::insertDynamicLibFileDependence
00084     (std::string filename) {
00085         _dynamicLibFilenameDependencies->insert(_dynamicLibFilenameDependencies->end(), filename);
00086     }
00087
00088     void PluginInformation::setDynamicLibFilenameDependencies
00089     (std::list<std::string>* dynamicLibFilenameDependencies) {
00090         this->_dynamicLibFilenameDependencies = dynamicLibFilenameDependencies
00091     ;
00092     }
00093
00094     std::list<std::string>* PluginInformation::dynamicLibFilenameDependencies
00095     () const {
00096         return _dynamicLibFilenameDependencies;
00097     }
00098
00099     void PluginInformation::setGenerateReport(bool generateReport) {
00100         this->_generateReport = generateReport;
00101     }
00102
00103     void PluginInformation::setSendTransfer(bool sendTransfer) {
00104         this->_sendTransfer = sendTransfer;
00105     }
00106
00107     void PluginInformation::setReceiveTransfer(bool receiveTransfer) {
00108         this->_receiveTransfer = receiveTransfer;
00109     }
00110
00111     void PluginInformation::setSink(bool Sink) {
00112         _isSink = Sink;
00113     }
00114
00115     void PluginInformation::setSource(bool Source) {
00116         _isSource = Source;
00117     }
00118
00119     void PluginInformation::setObservation(std::string
00120     observation) {
00121         this->_observation = observation;
00122     }
00123
00124     void PluginInformation::setVersion(std::string
00125     version) {
00126         this->_version = version;
00127     }
00128
00129     void PluginInformation:: setDate(std::string date) {
00130         this->_date = date;
00131     }
00132
00133     void PluginInformation::setAuthor(std::string
00134     author) {
```

```

00128     this->_author = author;
00129 }
00130
00131     void PluginInformation::setMaximumOutputs(unsigned short
00132         _maximumOutputs) {
00133     this->_maximumOutputs = _maximumOutputs;
00134 }
00135     unsigned short PluginInformation::maximumOutputs() const {
00136     return _maximumOutputs;
00137 }
00138
00139     void PluginInformation::setMinimumOutputs(unsigned short
00140         _minimumOutputs) {
00141     this->_minimumOutputs = _minimumOutputs;
00142 }
00143     unsigned short PluginInformation::minimumOutputs() const {
00144     return _minimumOutputs;
00145 }
00146
00147     void PluginInformation::setMaximumInputs(unsigned short
00148         _maximumInputs) {
00149     this->_maximumInputs = _maximumInputs;
00150 }
00151     unsigned short PluginInformation::maximumInputs() const {
00152     return _maximumInputs;
00153 }
00154
00155     void PluginInformation::setMinimumInputs(unsigned short
00156         _minimumInputs) {
00157     this->_minimumInputs = _minimumInputs;
00158 }
00159     unsigned short PluginInformation::minimumInputs() const {
00160     return _minimumInputs;
00161 }
00162
00163

```

7.319 PluginInformation.h File Reference

```
#include <map>
#include <list>
```

Classes

- class [PluginInformation](#)

Typedefs

- [typedef ModelComponent *\(* StaticLoaderComponentInstance\) \(Model *, std::map< std::string, std::string > *\)](#)
- [typedef ModelElement *\(* StaticLoaderElementInstance\) \(Model *, std::map< std::string, std::string > *\)](#)
- [typedef PluginInformation *\(* StaticGetPluginInformation\) \(\)](#)

7.319.1 Typedef Documentation

7.319.1.1 StaticGetPluginInformation

```
typedef PluginInformation*(* StaticGetPluginInformation) ()
```

Definition at line 29 of file [PluginInformation.h](#).

7.319.1.2 StaticLoaderComponentInstance

```
typedef ModelComponent*(* StaticLoaderComponentInstance) (Model *, std::map< std::string, std::string > *)
```

Definition at line 26 of file [PluginInformation.h](#).

7.319.1.3 StaticLoaderElementInstance

```
typedef ModelElement*(* StaticLoaderElementInstance) (Model *, std::map< std::string, std::string > *)
```

Definition at line 27 of file [PluginInformation.h](#).

7.320 PluginInformation.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    PluginInformation.h
00009  * Author:  rlcancian
00010 *
00011  * Created on 9 de Setembro de 2019, 20:02
00012 */
00013
00014 #ifndef PLUGININFORMATION_H
00015 #define PLUGININFORMATION_H
00016
00017 #include <map>
00018 #include <list>
00019
00020 class ModelElement;
00021 class ModelComponent;
00022 class Model;
00023 class ElementManager;
00024
00025 // \todo: the following 2 types were different but now on they are the same and should be merged
00026 typedef ModelComponent* (*StaticLoaderComponentInstance)(Model*, std::map<std::string,
00027 std::string>*);
00028 typedef ModelElement* (*StaticLoaderElementInstance)(Model*, std::map<std::string,
00029 std::string>*);
00030
00031
00032 class PluginInformation {
00033 public:
00034     PluginInformation(std::string pluginTypename,
00035                         StaticLoaderComponentInstance componentloader);
00035     PluginInformation(std::string pluginTypename,
00036                         StaticLoaderElementInstance elementloader);
00036 public:
00037     // gets
00038     StaticLoaderElementInstance elementloader() const;
00039     StaticLoaderComponentInstance componentLoader() const;
```

```

00040     bool isGenerateReport() const;
00041     bool isComponent() const;
00042     bool isSendTransfer() const;
00043     bool isReceiveTransfer() const;
00044     bool isSink() const;
00045     bool isSource() const;
00046     std::string observation() const;
00047     std::string version() const;
00048     std::string date() const;
00049     std::string author() const;
00050     std::string pluginTypename() const;
00051 // sets
00052     void insertDynamicLibFileDependence(std::string filename);
00053     void setDynamicLibFilenameDependencies(std::list<std::string>*
00054         dynamicLibFilenameDependencies);
00055     std::list<std::string>* dynamicLibFilenameDependencies() const;
00056     void setGenerateReport(bool generateReport);
00057     void setSendTransfer(bool sendTransfer);
00058     void setReceiveTransfer(bool receiveTransfer);
00059     void setSink(bool Sink);
00060     void setSource(bool Source);
00061     void setObservation(std::string observation);
00062     void setVersion(std::string version);
00063     void setDate(std::string date);
00064     void setMaximumOutputs(unsigned short _maximumOutputs);
00065     unsigned short maximumOutputs() const;
00066     void setMinimumOutputs(unsigned short _minimumOutputs);
00067     unsigned short minimumOutputs() const;
00068     void setMaximumInputs(unsigned short _maximumInputs);
00069     unsigned short maximumInputs() const;
00070     void setMinimumInputs(unsigned short _minimumInputs);
00071     unsigned short minimumInputs() const;
00072 public:
00073 private:
00074     std::string _author = "prof. Dr. Ing. Rafael Luiz Cancian";
00075     std::string _date = "01/07/2018";
00076     std::string _version = "0.9.1";
00077     std::string _observation = "First implementation not fully completed nor tested. Use with caution.";
00078     bool _isSource = false;
00079     bool _isSink = false;
00080     bool _receiveTransfer = false;
00081     bool _sendTransfer = false;
00082     bool _generateReport = false;
00083     std::list<std::string>* _dynamicLibFilenameDependencies = new std::list<std::string>();
00084 // set from constructor
00085     std::string _pluginTypename;
00086     bool _isComponent = false;
00087     unsigned short _minimumInputs = 1;
00088     unsigned short _maximumInputs = 1;
00089     unsigned short _minimumOutputs = 1;
00090     unsigned short _maximumOutputs = 1;
00091     StaticLoaderComponentInstance _componentloader;
00092     StaticLoaderElementInstance _elementloader;
00093 };
00094
00095
00096 #endif /* PLUGININFORMATION_H */
00097

```

7.321 PluginManager.cpp File Reference

```

#include "PluginManager.h"
#include "Simulator.h"
#include "Traits.h"

```

7.322 PluginManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*

```

```
00008 * File: PluginManager.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Maio de 2019, 17:49
00012 */
00013
00014 #include "PluginManager.h"
00015 #include "Simulator.h"
00016 #include "Traits.h"
00017
00018 PluginManager::PluginManager(Simulator* simulator) {
00019     _simulator = simulator;
00020     this->_pluginConnector = new Traits<PluginConnector_if>::Implementation
00021     ();
00022
00023
00024 //bool PluginManager::check(Plugin* plugin){
00025 //    return true;
00026 //}
00027
00028 bool PluginManager::_insert(Plugin* plugin) {
00029     PluginInformation* plugInfo = plugin->pluginInfo();
00030     if (plugInfo->isValidPlugin() && plugInfo != nullptr) {
00031         std::string msg = "Inserting ";
00032         if (plugInfo->isComponent())
00033             msg += "component";
00034         else
00035             msg += "element";
00036         msg += " plugin \"" + plugin->pluginTypename() + "\"";
00037         _simulator->tracer()->trace(Util::TraceLevel::simulatorDetailed
00038         , msg);
00039         // insert all dependencies before to insert this plugin
00040         bool allDependenciesInserted = true;
00041         if (plugInfo->dynamicLibFilenameDependencies()->size() > 0) {
00042             Util::IncIndent();
00043             {
00044                 _simulator->tracer()->trace(
00045                     Util::TraceLevel::simulatorDetailed, "Inserting dependencies...");
00046                 for (std::list<std::string>::iterator it = plugInfo->
00047                     dynamicLibFilenameDependencies()->begin(); it != plugInfo->
00048                     dynamicLibFilenameDependencies()->end(); it++) {
00049                     allDependenciesInserted &= (this->insert(*it) != nullptr);
00050                 }
00051             }
00052             Util::DecIndent();
00053         }
00054         if (!allDependenciesInserted) {
00055             _simulator->tracer()->trace(Util::TraceLevel::errorRecover
00056             , "Plugin dependencies could not be inserted; therefore, plugin will not be inserted");
00057             return false;
00058         }
00059         if (this->find(plugInfo->pluginTypename()) != nullptr) { // plugin already exists
00060             _simulator->tracer()->trace(
00061                 Util::TraceLevel::simulatorDetailed, "Plugin already exists and was not
00062                 inserted again");
00063             return false;
00064         }
00065     }
00066 }
00067
00068 bool PluginManager::check(std::string dynamicLibraryFilename) {
00069     Plugin* plugin;
00070     try {
00071         plugin = _pluginConnector->check(dynamicLibraryFilename);
00072     } catch (...) {
00073         return false;
00074     }
00075     return (plugin != nullptr);
00076 }
00077
00078 Plugin* PluginManager::insert(std::string dynamicLibraryFilename) {
00079     Plugin* plugin;
00080     try {
00081         plugin = _pluginConnector->connect(dynamicLibraryFilename);
00082         if (plugin != nullptr)
00083             _insert(plugin);
00084     } else {
```

```

00085     _simulator->tracer()->trace(Util::TraceLevel::errorRecover
00086     , "Plugin from file \"" + dynamicLibraryFilename + "\" could not be loaded.");
00087     } catch (...) {
00088     }
00089     return nullptr;
00090     }
00091     return plugin;
00092 }
00093
00094 bool PluginManager::remove(std::string dynamicLibraryFilename) {
00095
00096     Plugin* pi = this->find(dynamicLibraryFilename);
00097     remove(pi);
00098 }
00099
00100 bool PluginManager::remove(Plugin* plugin) {
00101     if (plugin != nullptr) {
00102         _plugins->remove(plugin);
00103         try {
00104             _pluginConnector->disconnect(plugin);
00105         } catch (...) {
00106             return false;
00107         }
00108         _simulator->tracer()->trace(Util::TraceLevel::simulatorResult
00109         , "Plugin successfully removed");
00110         return true;
00111     }
00112     _simulator->tracer()->trace(Util::TraceLevel::simulatorResult
00113         , "Plugin could not be removed");
00114     return false;
00115 }
00116 Plugin* PluginManager::find(std::string pluginTypeName) {
00117     for (std::list<Plugin*>::iterator it = this->_plugins->list()->begin(); it != _plugins->
00118         list()->end(); it++) {
00119         if ((*it)->pluginInfo()->pluginTypename() == pluginTypeName) {
00120             return (*it);
00121         }
00122     }
00123     return nullptr;
00124 }
00125 Plugin* PluginManager::front() {
00126
00127     return this->_plugins->front();
00128 }
00129
00130 Plugin* PluginManager::next() {
00131
00132     return _plugins->next();
00133 }
00134
00135 Plugin* PluginManager::last() {
00136     return this->_plugins->last();
00137 }

```

7.323 PluginManager.h File Reference

```
#include "List.h"
#include "Plugin.h"
#include "PluginConnector_if.h"
```

Classes

- class [PluginManager](#)

7.324 PluginManager.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
```

```

00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: PluginManager.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Maio de 2019, 17:49
00012 */
00013
00014 #ifndef PLUGINMANAGER_H
00015 #define PLUGINMANAGER_H
00016
00017 #include "List.h"
00018 #include "Plugin.h"
00019 #include "PluginConnector_if.h"
00020
00021 class Simulator;
00022
00023 class PluginManager {
00024 public:
00025     PluginManager(Simulator* simulator);
00026     virtual ~PluginManager() = default;
00027 public:
00028     bool check(const std::string dynamicLibraryFilename);
00029     Plugin* insert(const std::string dynamicLibraryFilename);
00030     bool remove(const std::string dynamicLibraryFilename);
00031     bool remove(Plugin* plugin);
00032     Plugin* find(std::string pluginTypeName);
00033 public:
00034     Plugin* front();
00035     Plugin* next();
00036     Plugin* last();
00037 private:
00038     bool _insert(Plugin* plugin);
00039 private:
00040     List<Plugin*>* _plugins = new List<Plugin*>();
00041     Simulator* _simulator;
00042     PluginConnector_if* _pluginConnector;
00043 };
00044
00045 #endif /* PLUGINMANAGER_H */
00046

```

7.325 ProbDistrib.cpp File Reference

```

#include "ProbDistrib.h"
#include <assert.h>
#include <cmath>
#include <complex>
#include <random>
#include <boost/math/distributions.hpp>

```

7.326 ProbDistrib.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ProbDistrib.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 23 de Agosto de 2018, 17:25
00012 */
00013
00014 #include "ProbDistrib.h"
00015 #include <assert.h>
00016 #include <cmath>
00017 #include <complex>
00018 #include <random>

```

```

00019
00020 #include <boost/math/distributions.hpp>
00021
00022 using namespace boost::math;
00023
00024 double ProbDistrib::uniform(double x, double min, double max) {
00025     if (x >= min && x <= max)
00026         return 1.0 / (max - min);
00027     else
00028         return 0.0;
00029 }
00030
00031 double ProbDistrib::exponential(double x, double mean) {
00032     assert(x >= 0);
00033     return mean * exp(-mean * x);
00034 }
00035
00036 double ProbDistrib::erlang(double x, double shape, double scale) { // 
00037     return ProbDistrib::gamma(x, shape, scale); // \todo: NOT EXACTLY THIS... REDO
00038 }
00039
00040 double ProbDistrib::normal(double x, double mean, double stddev) {
00041     double p1 = 1 / (stddev * std::sqrt(2 * M_PI));
00042     double rdf = (x - mean) / stddev;
00043     double p2 = std::exp(-0.5 * rdf * rdf);
00044     return p1*p2;
00045 }
00046
00047 double ProbDistrib::gamma(double x, double shape, double scale) {
00048     gamma_distribution<> my_gamma(shape, scale);
00049     return pdf(my_gamma, x);
00050 }
00051
00052 double ProbDistrib::beta(double x, double alpha, double beta) {
00053     beta_distribution<> my_beta(alpha, beta);
00054     return pdf(my_beta, x);
00055 }
00056
00057 double ProbDistrib::weibull(double x, double shape, double scale) {
00058     weibull_distribution<> my_weibull(shape, scale);
00059     return pdf(my_weibull, x);
00060 }
00061
00062 double ProbDistrib::logNormal(double x, double mean, double stddev) {
00063     lognormal_distribution<> my_lognormal(mean, stddev);
00064     return pdf(my_lognormal, x);
00065 }
00066
00067 double ProbDistrib::triangular(double x, double min, double mode, double max) {
00068     triangular_distribution<> my_triangular(min, mode, max);
00069     return pdf(my_triangular, x);
00070 }
00071
00072 double ProbDistrib::tStudent(double x, double mean, double stddev, unsigned int
    degreeFreedom) {
00073     students_t_distribution<> my_students_t(degreeFreedom);
00074     return pdf(my_students_t, x)*mean + stddev; //t-student SEEMS to be based on NORM(0,1)
00075 }
00076
00077 double ProbDistrib::fFisher(double x, double k, double m) {
00078     fisher_f_distribution<> my_fisher_f(k, m);
00079     return pdf(my_fisher_f, x);
00080 }
00081
00082 double ProbDistrib::chi2(double x, double m) {
00083     chi_squared_distribution<> my_chi_squared(m);
00084     return pdf(my_chi_squared, x);
00085 }
00086
00087 double ProbDistrib::poisson(double x, double mean) {
00088     poisson_distribution<> my_poisson(mean);
00089     return pdf(my_poisson, x);
00090 }
00091
00092 double ProbDistrib::inverseNormal(double cumulativeProbability, double mean,
    double stddev) {
00093     normal_distribution<> my_normal(mean, stddev);
00094     return quantile(my_normal, cumulativeProbability);
00095 }
00096
00097 double ProbDistrib::inverseTStudent(double cumulativeProbability, double mean,
    double stddev, double degreeFreedom) {
00098     students_t_distribution<> my_students_t(degreeFreedom);
00099     return quantile(my_students_t, cumulativeProbability);
00100 }
00101
00102 double ProbDistrib::inverseFFisher(double cumulativeProbability, double k,

```

```

00103     double m) {
00104         fisher_f_distribution<> my_fisher_f(k, m);
00105         return quantile(my_fisher_f, cumulativeProbability);
00106     }
00107     double ProbDistrib::inverseChi2(double cumulativeProbability, double m) {
00108         chi_squared_distribution<> my_chi_squared(m);
00109         return quantile(my_chi_squared, cumulativeProbability);
00110     }

```

7.327 ProbDistrib.h File Reference

```
#include <boost/math/distributions.hpp>
#include <boost/math/distributions/normal.hpp>
```

Classes

- class [ProbDistrib](#)

7.328 ProbDistrib.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007  * File:  ProbDistrib_if.h
00008  * Author: rafael.luiz.cancian
00009  *
00010  *
00011  * Created on 2 de Agosto de 2018, 00:32
00012 */
00013
00014 #ifndef PROBDISTRIB_IF_H
00015 #define PROBDISTRIB_IF_H
00016
00017 #include <boost/math/distributions.hpp>
00018 #include <boost/math/distributions/normal.hpp>
00019
00020 class ProbDistrib {
00021 public:
00022     static double beta(double x, double alpha, double beta);
00023     static double chi2(double x, double m);
00024     static double erlang(double x, double shape, double scale); // int M
00025     static double exponential(double x, double mean);
00026     static double ffisher(double x, double k, double m);
00027     static double gamma(double x, double shape, double scale);
00028     static double logNormal(double x, double mean, double stddev);
00029     static double normal(double x, double mean, double stddev);
00030     static double poisson(double x, double mean);
00031     static double triangular(double x, double min, double mode, double max);
00032     static double tStudent(double x, double mean, double stddev, unsigned int degreeFreedom);
00033     static double uniform(double x, double min, double max);
00034     static double weibull(double x, double shape, double scale);
00035     // inverse
00036     static double inverseChi2(double cumulativeProbability, double m);
00037     static double inverseFFisher(double cumulativeProbability, double k, double m);
00038     static double inverseNormal(double cumulativeProbability, double mean, double stddev);
00039     static double inverseTStudent(double cumulativeProbability, double mean, double stddev,
00040                                   double degreeFreedom);
00041 };
00042
00043 #endif /* PROBDISTRIB_IF_H */
00044

```

7.329 ProcessAnalyser_if.h File Reference

```
#include "List.h"
#include "SimulationScenario.h"
#include "SimulationControl.h"
#include "SimulationResponse.h"
#include "TraceManager.h"
```

Classes

- class [ProcessAnalyser_if](#)

7.330 ProcessAnalyser_if.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ProcessAnalyser_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 14:26
00012 */
00013
00014 #ifndef PROCESSANALYSER_IF_H
00015 #define PROCESSANALYSER_IF_H
00016
00017 #include "List.h"
00018 #include "SimulationScenario.h"
00019 #include "SimulationControl.h"
00020 #include "SimulationResponse.h"
00021 #include "TraceManager.h"
00022
00023 class ProcessAnalyser_if {
00024 public:
00025     virtual List<SimulationScenario*>* getScenarios() const = 0;
00026     virtual List<SimulationControl*>* getControls() const = 0;
00027     virtual List<SimulationResponse*>* getResponses() const = 0;
00028     virtual List<SimulationControl*>*
00029         extractControlsFromModel(std::string modelName) const = 0;
00030     virtual List<SimulationResponse*>*
00031         extractResponsesFromModel(std::string modelName) const = 0;
00032     virtual void startSimulationOfScenario(
00033         SimulationScenario* scenario) = 0;
00034     virtual void startSimulation() = 0;
00035     virtual void stopSimulation() = 0;
00036     virtual void addTraceSimulationHandler(
00037         traceSimulationProcessListener
00038         traceSimulationProcessListener) = 0;
00039 };
00040 #endif /* PROCESSANALYSER_IF_H */
```

7.331 ProcessAnalyserDefaultImpl1.cpp File Reference

```
#include "ProcessAnalyserDefaultImpl1.h"
```

7.332 ProcessAnalyserDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ProcessAnalyserDefaultImpl1.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Maio de 2019, 20:45
00012 */
00013
00014 #include "ProcessAnalyserDefaultImpl1.h"
00015
00016 ProcessAnalyserDefaultImpl1::ProcessAnalyserDefaultImpl1
00017 {
00018
00019
00020 List<SimulationScenario*>*
00021     ProcessAnalyserDefaultImpl1::getScenarios() const {
00022 }
00023 List<SimulationControl*>*
00024     ProcessAnalyserDefaultImpl1::getControls() const {
00025         return _controls;
00026 }
00027 List<SimulationResponse*>*
00028     ProcessAnalyserDefaultImpl1::getResponses() const {
00029 }
00030 List<SimulationControl*>*
00031     ProcessAnalyserDefaultImpl1::extractControlsFromModel(
00032         std::string modelName) const {
00033 }
00034 List<SimulationResponse*>*
00035     ProcessAnalyserDefaultImpl1::extractResponsesFromModel(
00036         std::string modelName) const {
00037 }
00038 void ProcessAnalyserDefaultImpl1::startSimulationOfScenario
00039     (SimulationScenario* scenario) {
00040 }
00041 void ProcessAnalyserDefaultImpl1::startSimulation() {
00042 }
00043 void ProcessAnalyserDefaultImpl1::stopSimulation() {
00044 }
00045 void ProcessAnalyserDefaultImpl1::addTraceSimulationHandler
00046     (traceSimulationProcessListener
        traceSimulationProcessListener) {

```

7.333 ProcessAnalyserDefaultImpl1.h File Reference

```

#include "ProcessAnalyser_if.h"
#include "SimulationScenario.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"

```

Classes

- class [ProcessAnalyserDefaultImpl1](#)

7.334 ProcessAnalyserDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  ProcessAnalyserDefaultImpl1.h
00009  * Author: rlcancian
00010 *
00011  * Created on 20 de Maio de 2019, 20:45
00012 */
00013
00014 #ifndef PROCESSANALYSERDEFAULTIMPL1_H
00015 #define PROCESSANALYSERDEFAULTIMPL1_H
00016
00017 #include "ProcessAnalyser_if.h"
00018 #include "SimulationScenario.h"
00019 #include "SimulationResponse.h"
00020 #include "SimulationControl.h"
00021
00022 class ProcessAnalyserDefaultImpl1 : public
00023     ProcessAnalyser_if {
00024     public:
00025         ProcessAnalyserDefaultImpl1();
00026         virtual ~ProcessAnalyserDefaultImpl1() = default;
00027     protected:
00028         virtual List<SimulationScenario*>* getScenarios() const;
00029         virtual List<SimulationControl*>* getControls() const;
00030         virtual List<SimulationResponse*>* getResponses() const;
00031         virtual List<SimulationControl*>*
00032             extractControlsFromModel(std::string modelName) const;
00033         virtual List<SimulationResponse*>*
00034             extractResponsesFromModel(std::string modelName) const;
00035         virtual void startSimulationOfScenario(
00036             SimulationScenario* scenario);
00037         virtual void startSimulation();
00038         virtual void stopSimulation();
00039         virtual void addTraceSimulationHandler(
00040             traceSimulationProcessListener
00041             traceSimulationProcessListener);
00042     private:
00043         List<SimulationControl*>* _controls = new
00044             List<SimulationControl*>();
00045     };
00046 #endif /* PROCESSANALYSERDEFAULTIMPL1_H */
00047

```

7.335 Queue.cpp File Reference

```

#include "Queue.h"
#include "Model.h"
#include "Attribute.h"

```

7.336 Queue.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Queue.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 21 de Agosto de 2018, 17:12
00012 */
00013
00014 #include "Queue.h"
00015 #include "Model.h"

```

```
00016 #include "Attribute.h"
00017
00018 Queue::Queue(Model* model, std::string name) : ModelElement(model,
00019     Util::TypeOf<Queue>(), name) {
00020     _initCStats();
00021 }
00022 void Queue::_initCStats() {
00023     _cstatNumberInQueue = new StatisticsCollector(
00024         _parentModel, _name + "." + "Number_In_Queue", this); /* \todo: ++ WHY THIS INSERT
00025         "DISPOSE" AND "10ENTITYTYPE" STATCOLL ?? */
00026     _cstatTimeInQueue = new StatisticsCollector(_parentModel,
00027         _name + "." + "Time_In_Queue", this);
00028     _childrenElements->insert({"NumberInQueue", _cstatNumberInQueue});
00029     _childrenElements->insert({"TimeInQueue", _cstatTimeInQueue});
00030 }
00031
00032 }
00033
00034 std::string Queue::show() {
00035     return ModelElement::show() +
00036         ",waiting=" + this->_list->show();
00037 }
00038
00039 void Queue::insertElement(Waiting* element) {
00040     _list->insert(element);
00041     this->_cstatNumberInQueue->getStatistics()->getCollector()->
00042         addValue(_list->size());
00043
00044 void Queue::removeElement(Waiting* element) {
00045     double tnow = _parentModel->simulation()->
00046         simulatedTime();
00047     _list->remove(element);
00048     this->_cstatNumberInQueue->getStatistics()->getCollector()->
00049         addValue(_list->size());
00050     double timeInQueue = tnow - element->getTimeStartedWaiting();
00051     this->_cstatTimeInQueue->getStatistics()->getCollector()->
00052         addValue(timeInQueue);
00053 }
00054
00055
00056 unsigned int Queue::size() {
00057     return _list->size();
00058 }
00059
00060 Waiting* Queue::first() {
00061     return _list->front();
00062 }
00063
00064 Waiting* Queue::getAtRank(unsigned int rank) {
00065     return _list->getAtRank(rank);
00066 }
00067
00068 void Queue::setAttributeName(std::string attributeName) {
00069     this->_attributeName = attributeName;
00070 }
00071
00072 std::string Queue::getAttributeName() const {
00073     return _attributeName;
00074 }
00075
00076 void Queue::setOrderRule(OrderRule _orderRule) {
00077     this->_orderRule = _orderRule;
00078 }
00079
00080 OrderRule Queue::getOrderRule() const {
00081     return _orderRule;
00082 }
00083
00084 double Queue::sumAttributesFromWaiting(
00085     Util::identification attributeID) {
00086     double sum = 0.0;
00087     for (std::list<Waiting*>::iterator it = _list->list()->begin(); it != _list->
00088         list()->end(); it++) {
00089         sum += (*it)->getEntity()->attributeValue(attributeID);
00090     }
00091     return sum;
00092 }
00093
00094 double Queue::getAttributeFromWaitingRank(unsigned int rank,
```

```

00093     Util::identification attributeID) {
00094     Waiting* wait = _list->getAtRank(rank);
00095     if (wait != nullptr) {
00096         return wait->getEntity()->attributeValue(attributeID);
00097     }
00098     return 0.0;
00099 }
00100 PluginInformation* Queue::GetPluginInformation() {
00101     PluginInformation* info = new PluginInformation(Util::TypeOf<Queue>()
00102     , &Queue::LoadInstance);
00103     return info;
00104 }
00105 ModelElement* Queue::LoadInstance(Model* model, std::map<std::string,
00106     std::string>* fields) {
00107     Queue* newElement = new Queue(model);
00108     try {
00109         newElement->_loadInstance(fields);
00110     } catch (const std::exception& e) {
00111     }
00112     return newElement;
00113 }
00114
00115 bool Queue::_loadInstance(std::map<std::string, std::string>* fields) {
00116     bool res = ModelElement::_loadInstance(fields);
00117     if (res) {
00118         try {
00119             this->_attributeName = (*fields->find("attributeName")).second;
00120             this->_orderRule = static_cast<OrderRule> (std::stoi((*fields->find("orderRule")).second));
00121         } catch (...) {
00122         }
00123     }
00124     return res;
00125 }
00126
00127 std::map<std::string, std::string>* Queue::_saveInstance() {
00128     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00129     //Util::TypeOf<Queue>());
00130     fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00131     fields->emplace("attributeName", this->_attributeName);
00132 }
00133
00134 bool Queue::_check(std::string* errorMessage) {
00135     return _parentModel->elements()->check(Util::TypeOf<Attribute>(),
00136         _attributeName, "AttributeName", false, errorMessage);
00137 }
00138 void Queue::_createInternalElements() {
00139     //__initCStats();
00140 }
00141
00142 ParserChangesInformation*
00143     Queue::_getParserChangesInformation() {
00144     ParserChangesInformation* changes = new
00145         ParserChangesInformation();
00146     //changes->getProductionToAdd()->insert(...);
00147     //changes->getTokensToAdd()->insert(...);
00148     return changes;
00149 }

```

7.337 Queue.h File Reference

```
#include "ModelElement.h"
#include "List.h"
#include "Entity.h"
#include "ElementManager.h"
#include "StatisticsCollector.h"
#include "Plugin.h"
#include "ModelComponent.h"
```

Classes

- class Waiting

- class Queue

7.338 Queue.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Queue.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 17:12
00012 */
00013
00014 #ifndef QUEUE_H
00015 #define QUEUE_H
00016
00017 #include "ModelElement.h"
00018 #include "List.h"
00019 #include "Entity.h"
00020 #include "ElementManager.h"
00021 #include "StatisticsCollector.h"
00022 #include "Plugin.h"
00023 #include "ModelComponent.h"
00024
00025 class Waiting {
00026 public:
00027
00028     Waiting(Entity* entity, ModelComponent* component, double timeStartedWaiting
00029     ) {
00030         _entity = entity;
00031         _component = component;
00032         _timeStartedWaiting = timeStartedWaiting;
00033     }
00034
00035     virtual ~Waiting()=default;
00036 public:
00037     virtual std::string show() {
00038         return //ModelElement::show()+
00039             ",entity=" + std::to_string(_entity->id()) +
00040             ",component=\"" + _component->name() + "\""+
00041             ",timeStatedWaiting=" + std::to_string(_timeStartedWaiting);
00042     }
00043 public:
00044     double getTimeStartedWaiting() const {
00045         return _timeStartedWaiting;
00046     }
00047
00048     ModelComponent* getComponent() const {
00049         return _component;
00050     }
00051
00052     Entity* getEntity() const {
00053         return _entity;
00054     }
00055
00056 private:
00057     Entity* _entity;
00058     ModelComponent* _component;
00059     double _timeStartedWaiting;
00060 };
00061
00091 class Queue : public ModelElement {
00092 public:
00093
00094     enum class OrderRule : int {
00095         FIFO = 1, LIFO = 2, HIGHESTVALUE = 3, SMALLESTVALUE = 4
00096     };
00097
00098 public:
00099     Queue(Model* model, std::string name="");
00100     virtual ~Queue();
00101 public:
00102     virtual std::string show();
00103 public:
00104     static PluginInformation* GetPluginInformation();
00105     static ModelElement* LoadInstance(Model* model, std::map<std::string, std::string>*
00106                                         fields);
00106 public:

```

```

00107     void insertElement(Waiting* element);
00108     void removeElement(Waiting* element);
00109     unsigned int size();
00110     Waiting* first();
00111     Waiting* getAtRank(unsigned int rank);
00112     void setAttributeName(std::string _attributeName);
00113     std::string getAttributeName() const;
00114     void setOrderRule(OrderRule _orderRule);
00115     Queue::OrderRule getOrderRule() const;
00116 public: // to implement SIMAN functions
00117     double sumAttributesFromWaiting(Util::identification attributeID); // use to
        implement SIMAN SAQUE function
00118     double getAttributeFromWaitingRank(unsigned int rank, Util::identification
        attributeID);
00119 public:
00120     void initBetweenReplications();
00121 protected:
00122     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00123     virtual std::map<std::string, std::string>* _saveInstance();
00124     virtual bool _check(std::string* errorMessage);
00125     virtual void _createInternalElements();
00126     virtual ParserChangesInformation* _getParserChangesInformation();
00127
00128 private:
00129     void _initCStats();
00130 private: //1::n
00131     List<Waiting*>* _list = new List<Waiting*>();
00132 private: //1::1
00133     OrderRule _orderRule = OrderRule::FIFO;
00134     std::string _attributeName = "";
00135 private: // inner children elements
00136     StatisticsCollector* _cstatNumberInQueue;
00137     StatisticsCollector* _cstatTimeInQueue;
00138 };
00139
00140 #endif /* QUEUE_H */
00141

```

7.339 README.md File Reference

7.340 README.md

```

00001 # ReGenESyS
00002 ### Reborn Generic and Expansible System Simulator
00003 ##### Genesys is a result of teaching and research activities of Professor Dr. Ing Rafael Luiz
        Cancian. It began in early 2002 as a way to teach students the basics and simulation techniques of systems
        implemented by other comercial simulation tools, such as Arena. In Genesys development he replicated all the SIMAN
        language, used by Arena software, and Genesys has become a clone of that tool, including its graphical
        interface. Genesys allowed the inclusion of new simulation components through dynamic link libraries and also the
        parallel execution of simulation models in a distributed environment. The development of Genesys continued
        until 2007, when the professor stopped teaching systems simulation classes. Ten years later the professor
        starts again to teach systems simulation classes and to carry out scientific research in the area. So in 2018
        Genesys is reborn as ReGenesys, with new language and programming techniques, and even more ambitious goals.
00004
00005 ##### Developed by [rlcancian](https://github.com/rlcancian)

```

7.341 Record.cpp File Reference

```

#include "Record.h"
#include "Model.h"
#include <fstream>
#include <cstdio>
#include <iostream>

```

7.342 Record.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Record.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 9 de Agosto de 2018, 13:52
00012 */
00013
00014 #include "Record.h"
00015 #include "Model.h"
00016 #include <fstream>
00017 #include <cstdio>
00018 #include <iostream>
00019
00020 Record::Record(Model* model, std::string name) :
00021     ModelComponent(model, Util::TypeOf<Record>(), name) {
00022     _cstatExpression = new StatisticsCollector(_parentModel, _expressionName
00023 , this);
00024     _parentModel->elements()->insert (_cstatExpression);
00025 }
00026
00027 Record::~Record() {
00028     _parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(),
00029     _cstatExpression);
00030
00031     std::string Record::show() {
00032         return ModelComponent::show() +
00033             ",expressionName=\"" + this->_expressionName + "\" " +
00034             ",expression=\"" + _expression + "\" " +
00035             "filename=\"" + _filename + "\"";
00036     }
00037
00038     void Record::setExpressionName(std::string expressionName) {
00039         this->_expressionName = expressionName;
00040         this->_cstatExpression->setName(expressionName);
00041     }
00042
00043     std::string Record::getExpressionName() const {
00044         return _expressionName;
00045     }
00046
00047     StatisticsCollector* Record::getCstatExpression() const {
00048         return _cstatExpression;
00049     }
00050
00051     void Record::setFilename(std::string filename) {
00052         this->_filename = filename;
00053     }
00054
00055     std::string Record::getFilename() const {
00056         return _filename;
00057     }
00058
00059     void Record::setExpression(std::string expression) {
00060         this->_expression = expression;
00061     }
00062
00063
00064     void Record::_execute(Entity* entity) {
00065         double value = _parentModel->parseExpression(_expression);
00066         _cstatExpression->getStatistics()->getCollector()->
00067             addValue(value);
00068         std::ofstream file;
00069         file.open(_filename, std::ofstream::out | std::ofstream::app);
00070         file << value << std::endl;
00071         file.close(); // \todo: open and close for every data is not a good idea. Should open when replication
00072         // starts and close when it finishes.
00073         _parentModel->tracer()->traceSimulation(
00074             _parentModel->simulation()->simulatedTime(), entity, this, "Recording
00075             value " + std::to_string(value));
00076         _parentModel->sendEntityToComponent(entity, this->
00077             nextComponents()->frontConnection(), 0.0);
00078     }
00079
00080     std::map<std::string, std::string>* Record::_saveInstance() {
```

```

00077     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00078     ( //Util::TypeOf<Record>());
00079     fields->emplace("expression0" , this->_expression);
00080     fields->emplace("expressionName0" , this->_expressionName);
00081     fields->emplace("fileName0" , this->_filename);
00082     return fields;
00083
00084 bool Record::_loadInstance(std::map<std::string, std::string>* fields) {
00085     bool res = ModelComponent::_loadInstance(fields);
00086     if (res) {
00087         this->_expression = ((*fields->find("expression0"))).second;
00088         this->_expressionName = ((*fields->find("expressionName0"))).second;
00089         this->_filename = ((*fields->find("fileName0"))).second;
00090     }
00091     return res;
00092 }
00093
00094 void Record::_initBetweenReplications() {
00095 }
00096
00097 bool Record::_check(std::string* errorMessage) {
00098     // when cheking the model (before simulating it), remove the file if exists
00099     std::remove(_filename.c_str());
00100     return _parentModel->checkExpression(_expression, "expression", errorMessage
00101 );
00102
00103 PluginInformation* Record::GetPluginInformation(){
00104     PluginInformation* info = new PluginInformation(Util::TypeOf<Record>(
00105     ), &Record::LoadInstance); return info;
00106 }
00107
00108 ModelComponent* Record::LoadInstance(Model* model,
00109     std::map<std::string, std::string>* fields) {
00110     Record* newComponent = new Record(model);
00111     try {
00112         newComponent->_loadInstance(fields);
00113     } catch (const std::exception& e) {
00114     }
00115     return newComponent;
00116 }
00117 }
00118

```

7.343 Record.h File Reference

```
#include "ModelComponent.h"
#include <string>
```

Classes

- class Record

7.344 Record.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Record.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 9 de Agosto de 2018, 13:52
00012 */
00013
00014 #ifndef RECORD_H

```

```

00015 #define RECORD_H
00016
00017 #include "ModelComponent.h"
00018 #include <string>
00019
00020 class Record : public ModelComponent {
00021 public:
00022     Record(Model* model, std::string name = "");
00023     virtual ~Record();
00024 public:
00025     void setFilename(std::string filename);
00026     std::string getFilename() const;
00027     void setExpression(std::string expression);
00028     std::string getExpression() const;
00029     void setExpressionName(std::string expressionName);
00030     std::string getExpressionName() const;
00031     StatisticsCollector* getCstatExpression() const;
00032 public:
00033     virtual std::string show();
00034 public:
00035     static PluginInformation* GetPluginInformation();
00036     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00037                                         std::string*>* fields);
00038 protected:
00039     virtual void _execute(Entity* entity);
00040     virtual bool _loadInstance(std::map<std::string, std::string*>* fields);
00041     virtual void _initBetweenReplications();
00042     virtual std::map<std::string, std::string*>* _saveInstance();
00043     virtual bool _check(std::string* errorMessage);
00044 private:
00045     std::string _expression = "";
00046     std::string _expressionName = "";
00047     std::string _filename = "";
00048 private:
00049     StatisticsCollector* _cstatExpression; /* \todo: Create an internal class to
00050                                              aggregate ExpressionStatisticsCollector, and change Record to get a list of it, so Record can record a set of
00051                                              expressions into a set of files */
00052 };
00053
00054 #endif /* RECORD_H */
00055

```

7.345 Release.cpp File Reference

```

#include "Release.h"
#include "Model.h"
#include "Resource.h"
#include "Attribute.h"
#include <assert.h>

```

7.346 Release.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Release.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #include "Release.h"
00015 #include "Model.h"
00016 #include "Resource.h"
00017 #include "Attribute.h"
00018 #include <assert.h>
00019
00020 Release::Release(Model* model, std::string name) :
00021     ModelComponent(model, Util::TypeOf<Release>(), name) {
00022 }

```

```

00022
00023
00024 std::string Release::show() {
00025     return ModelComponent::show() +
00026         ",resourceType=" + std::to_string(static_cast<int>(this->_resourceType)) +
00027         ",resource=\"" + this->_resource->name() + "\" +
00028         ",quantity=" + this->_quantity;
00029 }
00030
00031 void Release::setPriority(unsigned short _priority) {
00032     this->_priority = _priority;
00033 }
00034
00035 unsigned short Release::priority() const {
00036     return _priority;
00037 }
00038
00039 void Release::setResourceType(Resource::ResourceType
00040     _resourceType) {
00041     this->_resourceType = _resourceType;
00042
00043 Resource::ResourceType Release::resourceType() const {
00044     return _resourceType;
00045 }
00046
00047 void Release::setQuantity(std::string _quantity) {
00048     this->_quantity = _quantity;
00049 }
00050
00051 std::string Release::quantity() const {
00052     return _quantity;
00053 }
00054
00055 void Release::setRule(Resource::ResourceRule _rule) {
00056     this->_rule = _rule;
00057 }
00058
00059 Resource::ResourceRule Release::rule() const {
00060     return _rule;
00061 }
00062
00063 void Release::setSaveAttribute(std::string _saveAttribute) {
00064     this->_saveAttribute = _saveAttribute;
00065 }
00066
00067 std::string Release::saveAttribute() const {
00068     return _saveAttribute;
00069 }
00070
00071 void Release::setResource(Resource* _resource) {
00072     this->_resource = _resource;
00073 }
00074
00075 Resource* Release::resource() const {
00076     return _resource;
00077 }
00078
00079 void Release::_execute(Entity* entity) {
00080     Resource* resource = nullptr;
00081     if (this->_resourceType == Resource::ResourceType::SET) {
00082         /* \todo: +: not implemented yet */
00083     } else {
00084         resource = this->_resource;
00085     }
00086     unsigned int quantity = _parentModel->parseExpression(this->
00087         _quantity);
00088     assert(_resource->getNumberBusy() >= quantity);
00089     _parentModel->tracer()->traceSimulation(
00090         _parentModel->simulation()->simulatedTime(), entity, this, "Entity frees
00091         " + std::to_string(quantity) + " units of resource \""
00092         + resource->name() + "\" seized on time "
00093         + std::to_string(_resource->getLastTimeSeized()));
00094     _resource->release(quantity, _parentModel->simulation()->
00095         simulatedTime()); //releases and sets the 'LastTimeSeized' property
00096     _parentModel->sendEntityToComponent(entity, this->
00097         nextComponents()->frontConnection(), 0.0);
00098 }
00099
00100 void Release::_initBetweenReplications() {
00101     this->_resource->initBetweenReplications();
00102 }
00103
00104 bool Release::_loadInstance(std::map<std::string, std::string>* fields) {
00105     bool res = ModelComponent::_loadInstance(fields);
00106     if (res) {
00107         this->_priority = std::stoi((*(fields->find("priority"))).second);
00108         this->_quantity = ((*(fields->find("quantity"))).second);
00109     }
00110 }
```

```
00102     this->_resourceType = static_cast<Resource::ResourceType> (std::stoi((*(fields->find("resourceType"))).second));
00103     this->_rule = static_cast<Resource::ResourceRule> (std::stoi((*(fields->find("rule"))).second));
00104     this->_saveAttribute = ((*(fields->find("saveAttribute"))).second);
00105     //Util::identitifcation resourceId = std::stoi((*(fields->find("resourceId"))).second);
00106     //Resource* res = dynamic_cast<Resource*> (_model->elements()->element(Util::TypeOf<Resource>(),
00107     resourceId));
00108     std::string resourceName = ((*(fields->find("resourceName"))).second);
00109     Resource* res = dynamic_cast<Resource*> (_parentModel->
00110     elements()->element(Util::TypeOf<Resource>(), resourceName));
00111     this->_resource = res;
00112 }
00113
00114 std::map<std::string, std::string>* Release::_saveInstance() {
00115     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00116     (); //Util::TypeOf<Release>());
00117     fields->emplace("priority", std::to_string(this->_priority));
00118     fields->emplace("quantity", this->_quantity);
00119     fields->emplace("resourceType", std::to_string(static_cast<int> (this->_resourceType)));
00120     fields->emplace("resourceId", std::to_string(this->_resource->id()));
00121     fields->emplace("resourceName", (this->_resource->name()));
00122     fields->emplace("rule", std::to_string(static_cast<int> (this->_rule)));
00123     fields->emplace("saveAttribute", this->_saveAttribute);
00124
00125 }
00126
00127 bool Release::_check(std::string* errorMessage) {
00128     bool resultAll = true;
00129     resultAll &= _parentModel->checkExpression(_quantity, "quantity",
00130     errorMessage);
00131     resultAll &= _parentModel->elements()->check(Util::TypeOf<Resource>(),
00132     _resource, "resource", errorMessage);
00133     resultAll &= _parentModel->elements()->check(Util::TypeOf<Attribute>(),
00134     _saveAttribute, "SaveAttribute", false, errorMessage);
00135
00136     return resultAll;
00137 }
00138
00139 void Release::setResourceName(std::string resourceName) throw () {
00140     ModelElement* resource = _parentModel->
00141     elements()->element(Util::TypeOf<Resource>(), resourceName);
00142     if (resource != nullptr) {
00143         this->_resource = dynamic_cast<Resource*> (resource);
00144     } else {
00145         throw std::invalid_argument("Resource does not exist");
00146     }
00147 }
00148 std::string Release::resourceName() const {
00149     return _resource->name();
00150 }
00151
00152 PluginInformation* Release::GetPluginInformation() {
00153     PluginInformation* info = new PluginInformation(Util::TypeOf<Release>
00154     (), &Release::LoadInstance);
00155     info->insertDynamicLibFileDependence("resource.so");
00156     return info;
00157 }
00158
00159 ModelComponent* Release::LoadInstance(Model* model,
00160     std::map<std::string, std::string>* fields) {
00161     Release* newComponent = new Release(model);
00162     try {
00163         newComponent->_loadInstance(fields);
00164     } catch (const std::exception& e) {
00165     }
00166     return newComponent;
00167 }
```

7.347 Release.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Resource.h"
#include "Plugin.h"
```

Classes

- class [Release](#)

7.348 Release.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Release.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #ifndef RELEASE_H
00015 #define RELEASE_H
00016
00017 #include <string>
00018
00019 #include "ModelComponent.h"
00020 #include "Resource.h"
00021 #include "Plugin.h"
00022
00063 class Release : public ModelComponent {
00064 public:
00065     Release(Model* model, std::string name = "");
00066     virtual ~Release() = default;
00067 public:
00068     virtual std::string show();
00069 public:
00070     static PluginInformation* GetPluginInformation();
00071     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00072                                         std::string>* fields);
00072 public: // get & set
00073     void setPriority(unsigned short _priority);
00074     unsigned short priority() const;
00075     void setResourceType(Resource::ResourceType _resourceType);
00076     Resource::ResourceType resourceType() const;
00077     void setQuantity(std::string _quantity);
00078     std::string quantity() const;
00079     void setRule(Resource::ResourceRule _rule);
00080     Resource::ResourceRule rule() const;
00081     void setSaveAttribute(std::string _saveAttribute);
00082     std::string saveAttribute() const;
00083     void setResource(Resource* _resource);
00084     Resource* resource() const;
00085     // indirect access to and Resource*
00086     void set resourceName(std::string resourceName) throw();
00087     std::string resourceName() const;
00088
00089 protected:
00090     virtual void _execute(Entity* entity);
00091     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00092     virtual void _initBetweenReplications();
00093     virtual std::map<std::string, std::string>* _saveInstance();
00094     virtual bool _check(std::string* errorMessage);
00095 private:
00096     // unsigned int _allocationType = 0; // uint ? enum?
00097     unsigned short _priority = 0;
00098     Resource::ResourceType _resourceType =
00099         Resource::ResourceType::RESOURCE;
00100     std::string _quantity = "1";
00101     Resource::ResourceRule _rule =
00102         Resource::ResourceRule::SMALLESTBUSY;
00103     std::string _saveAttribute = "";
00102
00103 private: // no g&s
00104     Resource* _resource;
00105 };
00106
00107 #endif /* RELEASE_H */
00108

```

7.349 Remove.cpp File Reference

```
#include "Remove.h"
#include "Model.h"
```

7.350 Remove.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 *
00008 * File: Remove.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Remove.h"
00015 #include "Model.h"
00016
00017 Remove::Remove(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<Remove>(), name) {
00018 }
00019
00020
00021 std::string Remove::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Remove::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00026     Remove* newComponent = new Remove(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031     return newComponent;
00032 }
00033 }
00034
00035 void Remove::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00037     this->_parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Remove::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Remove::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Remove::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Remove::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Remove::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Remove>(
), &Remove::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069
```

7.351 Remove.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class Remove

7.352 Remove.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Remove.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef REMOVE_H
00015 #define REMOVE_H
00016
00017 #include "ModelComponent.h"
00018
00037 class Remove : public ModelComponent {
00038 public: // constructors
00039     Remove(Model* model, std::string name="");
00040     virtual ~Remove() = default;
00041 public: // virtual
00042     virtual std::string show();
00043 public: // static
00044     static PluginInformation* GetPluginInformation();
00045     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00046                                         std::string>* fields);
00046 protected: // virtual
00047     virtual void _execute(Entity* entity);
00048     virtual void _initBetweenReplications();
00049     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00050     virtual std::map<std::string, std::string>* _saveInstance();
00051     virtual bool _check(std::string* errorMessage);
00052 private: // methods
00053 private: // attributes 1:1
00054 private: // attributes 1:n
00055 };
00056
00057
00058 #endif /* REMOVE_H */
00059
```

7.353 Request.cpp File Reference

```
#include "Request.h"
#include "Model.h"
```

7.354 Request.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006 /*
00007 * File: Request.cpp
00008 * Author: rlcancian
00009 *
00010 *
00011 * Created on 11 de Setembro de 2019, 13:17
00012 */
00013
00014 #include "Request.h"
00015 #include "Model.h"
00016
00017 Request::Request(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<Request>(), name) {
00018 }
00019
00020
00021 std::string Request::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Request::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00026     Request* newComponent = new Request(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
00034
00035 void Request::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00037     this->parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00038 }
00039
00040 bool Request::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelComponent::_loadInstance(fields);
00042     if (res) {
00043         //...
00044     }
00045     return res;
00046 }
00047
00048 void Request::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Request::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
();
00053     //...
00054     return fields;
00055 }
00056
00057 bool Request::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Request::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Request>
(),
        &Request::LoadInstance);
00065     // ...
00066     return info;
00067 }
00068
00069

```

7.355 Request.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class Request

7.356 Request.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Request.h
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 13:17
00012 */
00013
00014 #ifndef REQUEST_H
00015 #define REQUEST_H
00016
00017 #include "ModelComponent.h"
00018
00055 class Request : public ModelComponent {
00056 public: // constructors
00057     Request(Model* model, std::string name = "");
00058     virtual ~Request() = default;
00059 public: // virtual
00060     virtual std::string show();
00061 public: // static
00062     static PluginInformation* GetPluginInformation();
00063     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00064                                         std::string>* fields);
00064 protected: // virtual
00065     virtual void _execute(Entity* entity);
00066     virtual void _initBetweenReplications();
00067     virtual bool _loadInstance(std::map<std::string, std::string>* _saveInstance());
00068     virtual std::map<std::string, std::string>* _saveInstance();
00069     virtual bool _check(std::string* errorMessage);
00070 private: // methods
00071 private: // attributes 1:1
00072 private: // attributes 1:n
00073 };
00074
00075
00076 #endif /* REQUEST_H */
00077

```

7.357 RequirementTester.cpp File Reference

```
#include "RequirementTester.h"
```

7.358 RequirementTester.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: RequirementTester.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 4 de Novembro de 2019, 14:13
00012 */
00013
00014 #include "RequirementTester.h"
00015
00016 RequirementTester::RequirementTester() {

```

```

00017 }
00018
00019 RequirementTester::RequirementTester(const
00020     RequirementTester& orig) {
00021
00022 RequirementTester::~RequirementTester() {
00023 }
00024

```

7.359 RequirementTester.h File Reference

Classes

- class [RequirementTester](#)

7.360 RequirementTester.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: RequirementTester.h
00009  * Author: rlcancian
00010 *
00011 * Created on 4 de Novembro de 2019, 14:13
00012 */
00013
00014 #ifndef REQUIREMENTTESTER_H
00015 #define REQUIREMENTTESTER_H
00016
00017 class RequirementTester {
00018 public:
00019     RequirementTester();
00020     RequirementTester(const RequirementTester& orig);
00021     virtual ~RequirementTester();
00022 private:
00023 };
00024
00025
00026 #endif /* REQUIREMENTTESTER_H */
00027

```

7.361 Resource.cpp File Reference

```

#include "Resource.h"
#include "Counter.h"
#include "Model.h"

```

7.362 Resource.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Resource.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:52

```

```

00012  */
00013
00014 #include "Resource.h"
00015 #include "Counter.h"
00016 #include "Model.h"
00017
00018 Resource::Resource(Model* model, std::string name) :
00019     ModelElement(model, Util::TypeOf<Resource>(), name) {
00020     _initCStats();
00021 }
00022 void Resource::_initCStats() {
00023     _cstatTimeSeized = new StatisticsCollector(_parentModel,
00024         _name+"."+"Time_Seized", this);
00025     _numSeizes = new Counter(_parentModel, _name+"."+"Seizes", this);
00026     _numReleases = new Counter(_parentModel, _name+"."+"Releases", this);
00027     _childrenElements->insert({"TimeSeized",_cstatTimeSeized});
00028     _childrenElements->insert({"NumSeizes",_numSeizes});
00029     _childrenElements->insert({"NumReleases",_numReleases});
00030 }
00031 Resource::~Resource() {
00032     // _parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatTimeSeized);
00033     // _cstatTimeSeized->~StatisticsCollector();
00034 }
00035 std::string Resource::show() {
00036     return ModelElement::show() +
00037         ",capacity=" + std::to_string(_capacity) +
00038         ",costBusyByour=" + std::to_string(_costBusyHour) +
00039         ",costIdleByour=" + std::to_string(_costIdleHour) +
00040         ",costPerUse=" + std::to_string(_costPerUse) +
00041         ",state=" + std::to_string(static_cast<int> (_resourceState));
00042 }
00043
00044 void Resource::seize(unsigned int quantity, double tnow) {
00045     _numberBusy += quantity;
00046     _numSeizes->incCountValue(quantity);
00047     _lastTimeSeized = tnow;
00048     _resourceState = Resource::ResourceState::BUSY;
00049 }
00050
00051 void Resource::release(unsigned int quantity, double tnow) {
00052     if (_numberBusy >= quantity) {
00053         _numberBusy -= quantity;
00054     } else {
00055         _numberBusy = 0;
00056     }
00057     if (_numberBusy==0) {
00058         _resourceState = Resource::ResourceState::IDLE;
00059     }
00060     _numReleases->incCountValue(quantity);
00061     double timeSeized = tnow - _lastTimeSeized;
00062     // Collect statistics about time seized
00063     this->_cstatTimeSeized->getStatistics()->getCollector()->
00064     addValue(timeSeized);
00065     //
00066     _lastTimeSeized = timeSeized;
00067     _notifyEventHandlers();
00068 }
00069 void Resource::initBetweenReplications() {
00070     this->_lastTimeSeized = 0.0;
00071     this->_numberBusy = 0;
00072     this->_numSeizes->clear();
00073     this->_numReleases->clear();
00074 }
00075
00076 void Resource::setResourceState(ResourceState _resourceState) {
00077     this->_resourceState = _resourceState;
00078 }
00079
00080 Resource::ResourceState Resource::getResourceState() const
00081 {
00082     return _resourceState;
00083 }
00084 void Resource::setCapacity(unsigned int _capacity) {
00085     this->_capacity = _capacity;
00086 }
00087
00088 unsigned int Resource::getCapacity() const {
00089     return _capacity;
00090 }
00091
00092 void Resource::setCostBusyHour(double _costBusyHour) {
00093     this->_costBusyHour = _costBusyHour;
00094 }

```

```
00095
00096     double Resource::getCostBusyHour() const {
00097         return _costBusyHour;
00098     }
00099
00100    void Resource::setCostIdleHour(double _costIdleHour) {
00101        this->_costIdleHour = _costIdleHour;
00102    }
00103
00104    double Resource::getCostIdleHour() const {
00105        return _costIdleHour;
00106    }
00107
00108    void Resource::setCostPerUse(double _costPerUse) {
00109        this->_costPerUse = _costPerUse;
00110    }
00111
00112    double Resource::getCostPerUse() const {
00113        return _costPerUse;
00114    }
00115
00116    unsigned int Resource::getNumberBusy() const {
00117        return _numberBusy;
00118    }
00119
00120    void Resource::addResourceEventHandler(
00121        ResourceEventHandler eventHandler) {
00122        this->_resourceEventHandlers->insert(eventHandler); // \todo: priority should be registered as
00123        well, so handlers are invoked ordered by priority
00124
00125    double Resource::getLastTimeSeized() const {
00126        return _lastTimeSeized;
00127    }
00128
00129    void Resource::_notifyEventHandlers() {
00130        for (std::list<ResourceEventHandler>::iterator it = this->_resourceEventHandlers->
00131            list()->begin(); it != _resourceEventHandlers->list()->end(); it++) {
00132            (*it)(this);
00133        }
00134
00135    PluginInformation* Resource::GetPluginInformation() {
00136        PluginInformation* info = new PluginInformation(
00137            Util::TypeOf<Resource>(), &Resource::LoadInstance); return info;
00138    }
00139
00140    ModelElement* Resource::LoadInstance(Model* model,
00141        std::map<std::string, std::string>* fields) {
00142        Resource* newElement = new Resource(model);
00143        try {
00144            newElement->_loadInstance(fields);
00145        } catch (const std::exception& e) {
00146        }
00147        return newElement;
00148    }
00149
00150    bool Resource::_loadInstance(std::map<std::string, std::string>* fields) {
00151        bool res = ModelElement::_loadInstance(fields);
00152        if (res) {
00153            this->_capacity = std::stoi( (*fields->find("capacity")) .second );
00154            this->_costBusyHour = std::stod((*(fields->find("costBusyHour"))).second);
00155            this->_costIdleHour= std::stod((*(fields->find("costIdleHour"))).second);
00156            this->_costPerUse = std::stod((*(fields->find("costPerUse"))).second);
00157        }
00158        return res;
00159    }
00160
00161    std::map<std::string, std::string>* Resource::_saveInstance() {
00162        std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00163        //Util::TypeOf<Resource>());
00164        fields->emplace("capacity", std::to_string(this->_capacity));
00165        fields->emplace("costBusyHour", std::to_string(this->_costBusyHour));
00166        fields->emplace("costIdleHour", std::to_string(this->_costIdleHour));
00167        fields->emplace("costPerUse", std::to_string(this->_costPerUse));
00168        return fields;
00169    }
00170
00171    bool Resource::_check(std::string* errorMessage) {
00172        return true;
00173    }
00174
00175    void Resource::_createInternalElements() {
00176        //_initCStats();
00177    }
```

7.363 Resource.h File Reference

```
#include "ModelElement.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
#include "Counter.h"
#include "Plugin.h"
```

Classes

- class [Resource](#)

7.364 Resource.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Resource.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:52
00012 */
00013
00014 #ifndef RESOURCE_H
00015 #define RESOURCE_H
00016
00017 #include "ModelElement.h"
00018 #include "StatisticsCollector.h"
00019 #include "ElementManager.h"
00020 #include "Counter.h"
00021 #include "Plugin.h"
00022
00023 class Resource : public ModelElement {
00024 public:
00025     typedef std::function<void(Resource*) > ResourceEventHandler;
00026
00027     template<typename Class>
00028     static ResourceEventHandler SetResourceEventHandler(void (Class::*function)(Resource*), Class * object) {
00029         return std::bind(function, object, std::placeholders::_1);
00030     }
00031
00032     enum class ResourceType : int {
00033         SET = 1, RESOURCE = 2
00034     };
00035
00036     enum class ResourceRule : int {
00037         RANDOM = 1, CICLICAL = 2, ESPECIFIC = 3, SMALLESTBUSY = 4, LARGESTREMAININGCAPACITY = 5
00038     };
00039
00040     enum class ResourceState : int {
00041         IDLE = 1, BUSY = 2, FAILED = 3, INACTIVE = 4, OTHER = 5
00042     };
00043
00044     public:
00045         //Resource(Model* model);
00046         Resource(Model* model, std::string name(""));
00047         virtual ~Resource();
00048     public:
00049         virtual std::string show();
00050     public:
00051         static PluginInformation* GetPluginInformation();
00052         static ModelElement* LoadInstance(Model* model, std::map<std::string,
00053             std::string>* fields);
00054     public:
00055         void seize(unsigned int quantity, double tnow);
00056         void release(unsigned int quantity, double tnow);
00057         void initBetweenReplications();
00058     public: // g&s
00059         void setResourceState(ResourceState _resourceState);
```

```

00115     Resource::ResourceState getResourceState() const;
00116     void setCapacity(unsigned int _capacity);
00117     unsigned int getCapacity() const;
00118     void setCostBusyHour(double _costBusyHour);
00119     double getCostBusyHour() const;
00120     void setCostIdleHour(double _costIdleHour);
00121     double getCostIdleHour() const;
00122     void setCostPerUse(double _costPerUse);
00123     double getCostPerUse() const;
00124 public: // gets
00125     unsigned int getNumberBusy() const;
00126 public:
00127     void addResourceEventHandler(ResourceEventHandler eventHandler);
00128     double getLastTimeSeized() const;
00129 protected:
00130     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00131     virtual std::map<std::string, std::string>* _saveInstance();
00132     virtual bool _check(std::string* errorMessage);
00133     virtual void _createInternalElements();
00134
00135 private:
00136     void _initCStats();
00137     void _notifyEventHandlers();
00138 //private:
00139 //    ElementManager* _elems;
00140 private:
00141     unsigned int _capacity = 1;
00142     double _costBusyHour = 1.0;
00143     double _costIdleHour = 1.0;
00144     double _costPerUse = 1.0;
00145     ResourceState _resourceState = ResourceState::IDLE;
00146 private: // only gets
00147     unsigned int _numberBusy = 0;
00148     //unsigned int _numberOut = 0;
00149     double _lastTimeSeized = 0.0; // \todo: It won't work for resources with capacity>1, when not all
         capacity is seized and them some more are seized. Seized time of first units will be lost. I don't have a
         solution so far
00150 private: // not gets nor sets
00151     //unsigned int _seizes = 0;
00152     //double _whenSeized; // same as last? check
00153 private: // inner children elements
00154     StatisticsCollector* _cstatTimeSeized;
00155     Counter* _numSeizes;
00156     Counter* _numReleases;
00157 private: //1::n
00158     List<ResourceEventHandler>* _resourceEventHandlers = new
        List<ResourceEventHandler>();
00159     //aFailures: TStringList;
00160     //std::list<Failure*>* _failures;
00161 };
00162
00163 #endif /* RESOURCE_H */
00164

```

7.365 Route.cpp File Reference

```

#include "Route.h"
#include "Model.h"
#include "Attribute.h"
#include "Simulator.h"

```

7.366 Route.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Route.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */

```

```

00013
00014 #include "Route.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017 #include "Simulator.h"
00018
00019 Route::Route(Model* model, std::string name) : ModelComponent(model,
00020     Util::TypeOf<Route>(), name) {
00021
00022
00023 std::string Route::show() {
00024     std::string msg= ModelComponent::show() +
00025         ",destinationType='"+std::to_string(static_cast<int>(this->_routeDestinationType))+
00026         ",timeExpression='"+this->_routeTimeExpression+" "+Util::StrTimeUnit(this->
00027             _routeTimeTimeUnit);
00028     if (_station != nullptr)
00029         msg += ",station="+this->_station->name();
00030     return msg;
00031 }
00032 ModelComponent* Route::LoadInstance(Model* model,
00033     std::map<std::string, std::string>* fields) {
00034     Route* newComponent = new Route(model);
00035     try {
00036         newComponent->_loadInstance(fields);
00037     } catch (const std::exception& e) {
00038     }
00039     return newComponent;
00040 }
00041
00042 void Route::setStation(Station* _station) {
00043     this->_station = _station;
00044 }
00045
00046 Station* Route::getStation() const {
00047     return _station;
00048 }
00049
00050 void Route::setRouteTimeExpression(std::string _routeTimeExpression) {
00051     this->_routeTimeExpression = _routeTimeExpression;
00052 }
00053
00054 std::string Route::getRouteTimeExpression() const {
00055     return _routeTimeExpression;
00056 }
00057
00058 void Route::setRouteTimeTimeUnit(Util::TimeUnit _routeTimeTimeUnit
00059 ) {
00060     this->_routeTimeTimeUnit = _routeTimeTimeUnit;
00061 }
00062 Util::TimeUnit Route::getRouteTimeTimeUnit() const {
00063     return _routeTimeTimeUnit;
00064 }
00065
00066 void Route::setRouteDestinationType(
00067     DestinationType _routeDestinationType) {
00068     this->_routeDestinationType = _routeDestinationType;
00069 }
00070 Route::DestinationType Route::getRouteDestinationType()
00071     const {
00072         return _routeDestinationType;
00073 }
00074 void Route::_execute(Entity* entity) {
00075     // adds the route time to the TransferTime statistics / attribute related to the Entity
00076     double routeTime = _parentModel->parseExpression(_routeTimeExpression) *
00077         Util::TimeUnitConvert(_routeTimeTimeUnit, _parentModel->
00078             infos()->replicationLengthTimeUnit());
00079     entity->entityType()->statisticsCollector("Transfer Time")-
00080         getStatistics()->getCollector()->addValue(routeTime);
00081     entity->attributeValue("Entity.TransferTime", entity->
00082         attributeValue("Entity.TransferTime") + routeTime);
00083     if (routeTime > 0.0) {
00084         // calculates when this Entity will reach the end of this route and schedule this Event
00085         double routeEndTime = _parentModel->simulation()->
00086             simulatedTime() + routeTime;
00087         Event* newEvent = new Event(routeEndTime, entity, _station->
00088             getEnterIntoStationComponent());
00089         _parentModel->futureEvents()->insert(newEvent);
00090         _parentModel->tracer()->trace("End of route of entity " + std::to_string(entity
00091             ->entityNumber()) + " to the component '" + _station->
00092             getEnterIntoStationComponent()->name() + "' was scheduled to time " +
00093             std::to_string(routeEndTime));

```

```
00085     } else {
00086         // send without delay
00087         _parentModel->sendEntityToComponent(entity, _station->
00088             getEnterIntoStationComponent(), 0.0);
00089     }
00090 }
00091 bool Route::_loadInstance(std::map<std::string, std::string>* fields) {
00092     bool res = ModelComponent::_loadInstance(fields);
00093     if (res) {
00094         this->_routeTimeExpression = (*fields->find("routeTimeExpression")).second;
00095         this->_routeTimeTimeUnit = static_cast<Util::TimeUnit> (std::stoi((*fields->find("routeTimeTimeUnit")).second));
00096         this->_routeDestinationType = static_cast<Route::DestinationType> (std::stoi((*fields->find("routeDestinationType")).second));
00097         std::string stationName = ((*fields->find("stationName"))).second;
00098         Station* station = dynamic_cast<Station*> (_parentModel->
00099             elements()->element(Util::TypeOf<Station>(), stationName));
00100         this->_station = station;
00101     }
00102     return res;
00103 }
00104 void Route::_initBetweenReplications() {
00105 }
00106
00107 std::map<std::string, std::string>* Route::_saveInstance() {
00108     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00109     ();
00110     fields->emplace("stationId", std::to_string(this->_station->id()));
00111     fields->emplace("stationName", (this->_station->name()));
00112     fields->emplace("routeTimeExpression", this->_routeTimeExpression);
00113     fields->emplace("routeTimeTimeUnit", std::to_string(static_cast<int> (this->_routeTimeTimeUnit)));
00114     fields->emplace("routeDestinationType", std::to_string(static_cast<int> (this->_routeDestinationType)));
00115 }
00116
00117 bool Route::_check(std::string* errorMessage) {
00118     //include attributes needed
00119     ElementManager* elements = _parentModel->elements();
00120     std::vector<std::string> neededNames = {"Entity.TransferTime", "Entity.Station"};
00121     std::string neededName;
00122     for (unsigned int i = 0; i < neededNames.size(); i++) {
00123         neededName = neededNames[i];
00124         if (elements->element(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00125             Attribute* attr1 = new Attribute(_parentModel, neededName);
00126             elements->insert(attr1);
00127         }
00128     }
00129     // include StatisticsCollector needed in EntityType
00130     std::list<ModelElement*>* enttypes = elements->elementList(Util::TypeOf<EntityType>())->
00131     list();
00132     for (std::list<ModelElement*>::iterator it= enttypes->begin(); it!= enttypes->end(); it++) {
00133         static_cast<EntityType*>((it))->statisticsCollector("Transfer Time"); // force create this
00134         CStat before simulation starts
00135         }
00136         bool resultAll = true;
00137         resultAll &= _parentModel->checkExpression(_routeTimeExpression, "Route time
00138         expression", errorMessage);
00139         resultAll &= _parentModel->elements()->check(Util::TypeOf<Station>(), _station
00140         , "Station", errorMessage);
00141         if (resultAll) {
00142             resultAll &= _station->getEnterIntoStationComponent() != nullptr;
00143             if (!resultAll) {
00144                 errorMessage->append("Station has no component to enter into it");
00145             }
00146         }
00147 PluginInformation* Route::GetPluginInformation() {
00148     PluginInformation* info = new PluginInformation(Util::TypeOf<Route>()
00149     , &Route::LoadInstance);
00150     info->setSendTransfer(true);
00151     info->insertDynamicLibFileDependence("station.so");
00152     return info;
00153 }
00154 }
```

7.367 Route.h File Reference

```
#include "ModelComponent.h"
#include "Station.h"
```

Classes

- class [Route](#)

7.368 Route.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Route.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:15
00012 */
00013
00014 #ifndef ROUTE_H
00015 #define ROUTE_H
00016
00017 #include "ModelComponent.h"
00018 #include "Station.h"
00019
00020 class Route : public ModelComponent {
00021 public:
00022     enum class DestinationType : int {
00023         Station = 0, BySequence = 1
00024     };
00025 public:
00026     Route(Model* model, std::string name = "");
00027     virtual ~Route() = default;
00028 public:
00029     virtual std::string show();
00030 public:
00031     static PluginInformation* GetPluginInformation();
00032     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00033                                         std::string>* fields);
00034 public:
00035     void setStation(Station* _station);
00036     Station* getStation() const;
00037     void setRouteTimeExpression(std::string _routeTimeExpression);
00038     std::string getRouteTimeExpression() const;
00039     void setRouteTimeTimeUnit(Util::TimeUnit _routeTimeTimeUnit);
00040     Util::TimeUnit getRouteTimeTimeUnit() const;
00041     void setRouteDestinationType(DestinationType
00042         _routeDestinationType);
00043     Route::DestinationType getRouteDestinationType() const;
00044 public:
00045 protected:
00046     virtual void _execute(Entity* entity);
00047     virtual void _initBetweenReplications();
00048     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00049     virtual std::map<std::string, std::string>* _saveInstance();
00050     virtual bool _check(std::string* errorMessage);
00051 private:
00052     std::string _routeTimeExpression = "0.0";
00053     Util::TimeUnit _routeTimeTimeUnit = Util::TimeUnit::second;
00054     Route::DestinationType _routeDestinationType =
00055         DestinationType::Station;
00056 private: // association
00057     Station* _station;
00058 };
00059 #endif /* ROUTE_H */
```

7.369 Sampler_if.h File Reference

Classes

- class [Sampler_if](#)
- struct [Sampler_if::RNG_Parameters](#)

7.370 Sampler_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Sampler_if.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 13:20
00012 */
00013
00014 #ifndef Sampler_IF_H
00015 #define Sampler_IF_H
00016
00020 class Sampler_if {
00021 public:
00022
00026     struct RNG_Parameters {
00027         virtual ~RNG_Parameters() = default;
00028     };
00029 public: // probability distributions
00030     virtual double random() = 0;
00031     virtual double sampleBeta(double alpha, double beta, double infLimit, double supLimit) = 0;
00032     virtual double sampleDiscrete(double acumProb, double value, ...) = 0;
00033     virtual double sampleErlang(double mean, int M) = 0;
00034     virtual double sampleExponential(double mean) = 0;
00035     virtual double sampleGamma(double mean, double alpha) = 0;
00036     virtual double sampleLogNormal(double mean, double stddev) = 0;
00037     virtual double sampleNormal(double mean, double stddev) = 0;
00038     virtual double sampleTriangular(double min, double mode, double max) = 0;
00039     virtual double sampleUniform(double min, double max) = 0;
00040     virtual double sampleWeibull(double alpha, double scale) = 0;
00041 public:
00042     virtual void setRNGparameters(RNG_Parameters* param) = 0;
00043     virtual RNG_Parameters* getRNGparameters() const = 0;
00044 };
00045
00046 #endif /* Sampler_IF_H */
00047

```

7.371 SamplerBoostImpl.cpp File Reference

```
#include "SamplerBoostImpl.h"
```

7.372 SamplerBoostImpl.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SamplerBoostImpl.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 21 de Outubro de 2019, 17:24
00012 */

```

```

00013
00014 #include "SamplerBoostImpl.h"
00015
00016 using namespace boost::random;
00017
00018 SamplerBoostImpl::SamplerBoostImpl() {
00019 }
00020
00021 double SamplerBoostImpl::random() {
00022     uniform_int_distribution<> dist(0.0, 1.0);
00023     return dist(_gen);
00024 }
00025
00026 double SamplerBoostImpl::sampleBeta(double alpha, double beta, double infLimit,
00027     double supLimit) {
00028     return 0.0; //dummy
00029 }
00030 double SamplerBoostImpl::sampleDiscrete(double acumProb, double value, ...)
00031 {
00032     return 0.0; //dummy
00033 }
00034 double SamplerBoostImpl::sampleErlang(double mean, int M) {
00035     return 0.0; //dummy
00036 }
00037
00038 double SamplerBoostImpl::sampleExponential(double mean) {
00039     return 0.0; //dummy
00040 }
00041
00042 double SamplerBoostImpl::sampleGamma(double mean, double alpha) {
00043     return 0.0; //dummy
00044 }
00045
00046 double SamplerBoostImpl::sampleLogNormal(double mean, double stddev) {
00047     return 0.0; //dummy
00048 }
00049
00050 double SamplerBoostImpl::sampleNormal(double mean, double stddev) {
00051     return 0.0; //dummy
00052 }
00053
00054 double SamplerBoostImpl::sampleTriangular(double min, double mode, double
00055     max) {
00056     return 0.0; //dummy
00057 }
00058 double SamplerBoostImpl::sampleUniform(double min, double max) {
00059     uniform_int_distribution<> dist(min, max);
00060     return dist(_gen);
00061 }
00062
00063 double SamplerBoostImpl::sampleWeibull(double alpha, double scale) {
00064     return 0.0; //dummy
00065 }
00066
00067 void SamplerBoostImpl::setRNGparameters(
00068     Sampler_if::RNG_Parameters* param) {
00069 }
00070
00071 Sampler_if::RNG_Parameters*
00072     SamplerBoostImpl::getRNGparameters() const {
00073 }
```

7.373 SamplerBoostImpl.h File Reference

```
#include "Sampler_if.h"
#include <boost/random.hpp>
```

Classes

- class [SamplerBoostImpl](#)
- struct [SamplerBoostImpl::BoostImplRNG_Parameters](#)

7.374 SamplerBoostImpl.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SamplerBoostImpl.h
00009 * Author: rlcancian
00010 *
00011 * Created on 21 de Outubro de 2019, 17:24
00012 */
00013
00014 #ifndef SAMPLERBOOSTIMPL_H
00015 #define SAMPLERBOOSTIMPL_H
00016
00017 #include "Sampler_if.h"
00018 #include <boost/random.hpp>
00019
00020 class SamplerBoostImpl : public Sampler_if {
00021 public:
00022     struct BoostImplRNG_Parameters : public
00023         RNG_Parameters {
00024         double param;
00025     };
00026
00027 public:
00028     SamplerBoostImpl();
00029     virtual ~SamplerBoostImpl() = default;
00030 public: // probability distributions
00031     virtual double random();
00032     virtual double sampleBeta(double alpha, double beta, double infLimit, double supLimit);
00033     virtual double sampleDiscrete(double acumProb, double value, ...);
00034     virtual double sampleErlang(double mean, int M);
00035     virtual double sampleExponential(double mean);
00036     virtual double sampleGamma(double mean, double alpha);
00037     virtual double sampleLogNormal(double mean, double stddev);
00038     virtual double sampleNormal(double mean, double stddev);
00039     virtual double sampleTriangular(double min, double mode, double max);
00040     virtual double sampleUniform(double min, double max);
00041     virtual double sampleWeibull(double alpha, double scale);
00042 public:
00043     void reset();
00044 public:
00045     virtual void setRNGparameters(Sampler_if::RNG_Parameters*
00046         param);
00046     virtual RNG_Parameters* getRNGparameters() const;
00047 private:
00048     boost::random::mt19937 _gen;
00049 };
00050
00051 #endif /* SAMPLERBOOSTIMPL_H */
00052

```

7.375 SamplerDefaultImpl1.cpp File Reference

```

#include <cmath>
#include <complex>
#include <cassert>
#include "SamplerDefaultImpl1.h"

```

7.376 SamplerDefaultImpl1.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SamplerDefaultImpl1.cpp

```

```

00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Agosto de 2018, 01:10
00012 * 22/10/2019 old genesys code reinserted
00013 */
00014
00015 #include <cmath>
00016 #include <complex>
00017 #include <cassert>
00018
00019 #include "SamplerDefaultImpl1.h"
00020
00021 SamplerDefaultImpl1::SamplerDefaultImpl1() {
00022     reset();
00023 }
00024
00025 void SamplerDefaultImpl1::reset() {
00026     _xi = static_cast<DefaultImpl1RNG_Parameters*> (_param)->seed;
00027     //normalflag = true;
00028 }
00029
00030
00031 double SamplerDefaultImpl1::random() {
00032     double module = (double) static_cast<DefaultImpl1RNG_Parameters*> (_param)->module;
00033     _xi *= static_cast<DefaultImpl1RNG_Parameters*> (_param)->multiplier;
00034     _xi -= std::trunc((double) _xi / module) * module;
00035     return (double) _xi / (double) static_cast<DefaultImpl1RNG_Parameters*> (_param)->module;
00036 }
00037
00038 double SamplerDefaultImpl1::sampleUniform(double min, double max) {
00039     return min + (max - min) * random();
00040 }
00041
00042 double SamplerDefaultImpl1::sampleExponential(double mean) {
00043     return mean * (-std::log(random()));
00044 }
00045
00046 double SamplerDefaultImpl1::sampleErlang(double mean, int M) {
00047     unsigned int i;
00048     double P;
00049     assert((mean >= 0.0) && (M > 0));
00050     P = 1;
00051     for (i = 1; i <= M; i++) {
00052         P *= random();
00053     }
00054     return (mean / M) * (-log(P));
00055 }
00056
00057 double SamplerDefaultImpl1::sampleNormal(double mean, double stddev) {
00058     double z = std::sqrt(-2 * std::log(random())) * std::cos(2 * M_PI *
00059     random());
00060     return mean + stddev*z;
00061
00062 double SamplerDefaultImpl1::_gammaJong(double alpha) {
00063     double R;
00064     double R1, R2, X, Y;
00065     do {
00066         do {
00067             R1 = random();
00068             R2 = random();
00069         } while (!(R1 > 1e-30) and (R2 > 1e-30));
00070         if (log(R2) / alpha < -1e3)
00071             X = 0;
00072         else
00073             X = exp(log(R2) / alpha);
00074         if ((log(R1) / (1 - alpha) < -1e3))
00075             Y = 0;
00076         else
00077             Y = exp(log(R1) / (1 - alpha));
00078     } while (!(X + Y <= 1));
00079     do {
00080         R = random();
00081     } while (!(R > 1e-20));
00082     return -log(R) * X / (Y + X);
00083 }
00084
00085 double SamplerDefaultImpl1::sampleGamma(double mean, double alpha) {
00086     int i;
00087     double P;
00088     int IntAlpha;
00089     double OstAlpha;
00090     assert(!((mean <= 0.0) || (alpha <= 0.0)));
00091     if (alpha < 1.0)
00092         return (mean / alpha) * _gammaJong(alpha);
00093     else {
00094         if (alpha == 1.0)

```

```

00095     return mean * (-log(random()));
00096     else {
00097         IntAlpha = round(alpha);
00098         OstAlpha = alpha - IntAlpha;
00099         do {
00100             P = 1;
00101             for (i = 1; i <= IntAlpha; i++)
00102                 P *= random();
00103             } while (!(P > 0));
00104             if (OstAlpha > 0)
00105                 return (mean / alpha) * ((-log(P)) + _gammaJ0nk(OstAlpha));
00106             else
00107                 return (mean / alpha) * (-log(P));
00108     };
00109 };
00110 }
00111
00112 double SamplerDefaultImpl1::sampleBeta(double alpha, double beta, double
infLimit, double supLimit) {
00113     double X, Y1, Y2;
00114     assert(!((alpha <= 0.0) || (beta <= 0.0) || (infLimit > supLimit) || (infLimit < 0) || (supLimit < 0)));
00115     do {
00116         Y1 = sampleGamma(alpha, alpha);
00117         Y2 = sampleGamma(beta, beta);
00118         X = Y1 / (Y1 + Y2);
00119         } while (!(X >= 0) && (X <= 1.0)));
00120     return infLimit + (supLimit - infLimit) * X;
00121 }
00122
00123 double SamplerDefaultImpl1::sampleWeibull(double alpha, double scale) {
00124     assert(!((alpha <= 0.0) || (scale <= 0.0)));
00125     return exp(log(scale * (-log(random()))) / alpha);
00126 }
00127
00128 double SamplerDefaultImpl1::sampleLogNormal(double mean, double stddev)
{
00129     double meanNorm, DispNorm;
00130     assert(!((mean <= 0.0) || (stddev <= 0.0)));
00131     DispNorm = log((stddev * stddev) / (mean * mean) + 1.0);
00132     meanNorm = log(mean) - 0.5 * DispNorm;
00133     return exp(sampleNormal(meanNorm, sqrt(DispNorm)));
00134 }
00135
00136 double SamplerDefaultImpl1::sampleTriangular(double min, double mode,
double max) {
00137     double Part1, Part2, Full, R;
00138     assert(!((min > mode) || (max < mode) || (min > max)));
00139     Part1 = mode - min;
00140     Part2 = max - mode;
00141     Full = max - min;
00142     R = random();
00143     if (R <= Part1 / Full)
00144         return min + sqrt(Part1 * Full * R);
00145     else
00146         return max - sqrt(Part2 * Full * (1.0 - R));
00147 }
00148
00149 double SamplerDefaultImpl1::sampleDiscrete(double acumProb, double value
, ...) {
00150 }
00151
00152 void SamplerDefaultImpl1::setRNGparameters(
    Sampler_if::RNG_Parameters * param) {
00153
00154     _param = param; // there is a better solution for this...
00155 }
00156
00157 Sampler_if::RNG_Parameters *
    SamplerDefaultImpl1::getRNGparameters() const {
00158     return _param;
00159 }

```

7.377 SamplerDefaultImpl1.h File Reference

```
#include "Sampler_if.h"
```

Classes

- class SamplerDefaultImpl1

- struct SamplerDefaultImpl1::DefaultImpl1RNG_Parameters

7.378 SamplerDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 /*
00008  * File: SamplerDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 2 de Agosto de 2018, 01:10
00012 */
00013
00014 #ifndef SAMPLERDEFAULTIMPL1_H
00015 #define SAMPLERDEFAULTIMPL1_H
00016
00017 #include "Sampler_if.h"
00018
00019 class SamplerDefaultImpl1 : public Sampler_if {
00020 public:
00021
00022     struct DefaultImpl1RNG_Parameters : public
00023         RNG_Parameters {
00024         unsigned int seed = 666;
00025         unsigned int module = 2147483647;
00026         unsigned int multiplier = 950706376;
00027         ~DefaultImpl1RNG_Parameters() = default;
00028     };
00029 public:
00030     SamplerDefaultImpl1();
00031     virtual ~SamplerDefaultImpl1() = default;
00032 public: // probability distributions
00033     virtual double random();
00034     virtual double sampleBeta(double alpha, double beta, double infLimit, double supLimit);
00035     virtual double sampleDiscrete(double acumProb, double value, ...);
00036     virtual double sampleErlang(double mean, int M);
00037     virtual double sampleExponential(double mean);
00038     virtual double sampleGamma(double mean, double alpha);
00039     virtual double sampleLogNormal(double mean, double stddev);
00040     virtual double sampleNormal(double mean, double stddev);
00041     virtual double sampleTriangular(double min, double mode, double max);
00042     virtual double sampleUniform(double min, double max);
00043     virtual double sampleWeibull(double alpha, double scale);
00044 public:
00045     void reset();
00046 public:
00047     virtual void setRNGparameters(RNG_Parameters* param);
00048     virtual RNG_Parameters* getRNGparameters() const;
00049 private:
00050     double _gammaJ0nk(double alpha);
00051     RNG_Parameters* _param = new DefaultImpl1RNG_Parameters();
00052     unsigned int _xi;
00053     //bool _normalflag;
00054 };
00055
00056 #endif /* SAMPLERDEFAULTIMPL1_H */
00057

```

7.379 ScenarioExperiment_if.h File Reference

Classes

- class ScenarioExperiment_if

7.380 ScenarioExperiment_if.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ScenarioExperiment_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 14:52
00012 */
00013
00014 #ifndef SCENARIOEXPERIMENT_IF_H
00015 #define SCENARIOEXPERIMENT_IF_H
00016
00017 class ScenarioExperiment_if {
00018 };
00019
00020 #endif /* SCENARIOEXPERIMENT_IF_H */
00021
```

7.381 Schedule.cpp File Reference

```
#include "Schedule.h"
```

7.382 Schedule.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Schedule.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:16
00012 */
00013
00014 #include "Schedule.h"
00015
00016 Schedule::Schedule(Model* model) {
00017 }
00018
00019
```

7.383 Schedule.h File Reference

Classes

- class [Schedule](#)

7.384 Schedule.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Schedule.h
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:16
00012 */
00013
00014 #ifndef SCHEDULE_H
00015 #define SCHEDULE_H
00016
00017 class Model;
00018
00067 class Schedule {
00068 public:
00069     Schedule(Model* model);
00070     virtual ~Schedule() = default;
00071 private:
00072 };
00073 };
00074
00075 #endif /* SCHEDULE_H */
00076

```

7.385 Search.cpp File Reference

```
#include "Search.h"
#include "Model.h"
```

7.386 Search.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Search.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Search.h"
00015 #include "Model.h"
00016
00017 Search::Search(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<Search>(), name) {
00018 }
00019
00020
00021 std::string Search::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Search::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00026     Search* newComponent = new Search(model);
00027     try {
00028         newComponent->_loadInstance(fields);
00029     } catch (const std::exception& e) {
00030     }
00031 }
00032     return newComponent;
00033 }
00034
00035 void Search::_execute(Entity* entity) {

```

```

00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00037     entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
00039     nextComponents()->frontConnection(), 0.0);
00040 }
00041
00042 bool Search::_loadInstance(std::map<std::string, std::string>* fields) {
00043     bool res = ModelComponent::_loadInstance(fields);
00044     if (res) {
00045         //...
00046     }
00047     return res;
00048 }
00049
00050 void Search::_initBetweenReplications() {
00051     std::map<std::string, std::string>* Search::_saveInstance() {
00052         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053         ();
00054         //...
00055         return fields;
00056     }
00057     bool Search::_check(std::string* errorMessage) {
00058         bool resultAll = true;
00059         //...
00060         return resultAll;
00061     }
00062
00063 PluginInformation* Search::GetPluginInformation(){
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Search>(
00065         ), &Search::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069

```

7.387 Search.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Search](#)

7.388 Search.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Search.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef SEARCH_H
00015 #define SEARCH_H
00016
00017 #include "ModelComponent.h"
00018
00055 class Search : public ModelComponent {
00056 public: // constructors
00057     Search(Model* model, std::string name(""));
00058     virtual ~Search() = default;
00059 public: // virtual
00060     virtual std::string show();

```

```

00061 public: // static
00062     static PluginInformation* GetPluginInformation();
00063     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00064                                         std::string>* fields);
00064 protected: // virtual
00065     virtual void _execute(Entity* entity);
00066     virtual void _initBetweenReplications();
00067     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00068     virtual std::map<std::string, std::string>* _saveInstance();
00069     virtual bool _check(std::string* errorMessage);
00070 private: // methods
00071 private: // attributes 1:1
00072 private: // attributes 1:n
00073 };
00074
00075
00076 #endif /* SEARCH_H */
00077

```

7.389 SecondExampleOfSimulation.cpp File Reference

```

#include "SecondExampleOfSimulation.h"
#include "SinkModelComponent.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "EntityType.h"

```

7.390 SecondExampleOfSimulation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SecondExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 3 de Setembro de 2019, 18:59
00012 */
00013
00014 #include "SecondExampleOfSimulation.h"
00015 #include "SinkModelComponent.h"
00016
00017 // you have to included need libs
00018
00019 // GEnSys Simulator
00020 #include "Simulator.h"
00021
00022 // Model Components
00023 #include "Create.h"
00024 #include "Delay.h"
00025 #include "Dispose.h"
00026
00027 // Model elements
00028 #include "EntityType.h"
00029
00030 SecondExampleOfSimulation::SecondExampleOfSimulation()
00031 {
00032
00033 int SecondExampleOfSimulation::main(int argc, char** argv) {
00034     Simulator* simulator = new Simulator();
00035     // set the trace level of simulation to "blockArrival" level, which is an intermediate level of tracing
00036     simulator->tracer()->setTraceLevel(
00037         Util::TraceLevel::componentArrival);
00038     // Handle traces and simulation events to output them
00039     this->setDefaultTraceHandlers(simulator->tracer());
00040     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00041     this->insertFakePluginsByHand(simulator);
00042
00043
00044

```

```

00045     bool wantToCreateNewModelAndSaveInsteadOfJustLoad = true;
00046     Model model;
00047     if (wantToCreateNewModelAndSaveInsteadOfJustLoad) {
00048         // creates an empty model
00049         model = new Model(simulator);
00050         // build the simulation model
00051         // set general info about the model
00052         ModelInfo* infos = model->infos();
00053         infos->setAnalystName("Your name");
00054         infos->setProjectTitle("The title of the project");
00055         infos->setDescription("This simulation model tests one of the most basic models possible.
00056     ");
00057         infos->setReplicationLength(30);
00058         infos->setReplicationLengthTimeUnit(
00059             Util::TimeUnit::minute); // each replication will last 30 minutes (simulated time)
00060         infos->setNumberOfReplications(3); // replicates the simulation 3 times
00061         // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00062         EntityType* entityType1 = new EntityType(model, "Type_of_Representative_Entity");
00063         // model->insert(entityType1);
00064         // create a ModelComponent of type Create, used to insert entities into the model
00065         Create* create1 = new Create(model);
00066         create1->setEntityType(entityType1);
00067         create1->setTimeBetweenCreationsExpression("Expo(2)");
00068         create1->setTimeUnit(Util::TimeUnit::minute);
00069         create1->setEntitiesPerCreation(1);
00070         // model->insert(create1);
00071         // create a ModelComponent of type Delay, used to represent a time delay
00072         Delay* delay1 = new Delay(model);
00073         delay1->setDelayExpression("NORM(1,0.2)");
00074         delay1->setDelayTimeUnit(Util::TimeUnit::minute);
00075         // model->insert(delay1);
00076         // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00077         Dispose* dispose1 = new Dispose(model);
00078         // model->insert(dispose1);
00079         // connect model components to create a "workflow"
00080         create1->nextComponents()->insert(delay1);
00081         delay1->nextComponents()->insert(dispose1);
00082         // insert the model into the simulator
00083         simulator->models()->insert(model);
00084         // if the model is ok then save the model into a text file
00085         if (model->check())
00086             model->save("./temp/secondExampleOfSimulation.txt");
00087         } else {
00088             simulator->models()->loadModel("./temp/secondExampleOfSimulation.txt");
00089             model = simulator->models()->current();
00090         }
00091         // execute the simulation
00092         model->simulation()->start();
00093     }
00094 }
```

7.391 SecondExampleOfSimulation.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [SecondExampleOfSimulation](#)

7.392 SecondExampleOfSimulation.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SecondExampleOfSimulation.h
00009  * Author: rlcancian
00010 */

```

```

00011 * Created on 3 de Setembro de 2019, 18:59
00012 */
00013
00014 #ifndef SECONDEXAMPLEOFSIMULATION_H
00015 #define SECONDEXAMPLEOFSIMULATION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class SecondExampleOfSimulation: public
00020     BaseConsoleGenesysApplication {
00021     public:
00022         SecondExampleOfSimulation();
00023     public:
00024         virtual int main(int argc, char** argv);
00025     };
00026 #endif /* SECONDEXAMPLEOFSIMULATION_H */
00027

```

7.393 Seize.cpp File Reference

```

#include "Seize.h"
#include "Resource.h"
#include "Attribute.h"

```

7.394 Seize.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Seize.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #include "Seize.h"
00015 #include "Resource.h"
00016 #include "Attribute.h"
00017
00018 Seize::Seize(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Seize>(), name) {
00020     std::string Seize::show() {
00021         return ModelComponent::show() +
00022             ",resourceType=" + std::to_string(static_cast<int> (this->_resourceType)) +
00023             ",resource=\"" + this->_resource->name() + "\""
00024             ",quantity=" + this->_quantity;
00025     }
00026
00027 void Seize::setLastMemberSeized(unsigned int _lastMemberSeized) {
00028     this->_lastMemberSeized = _lastMemberSeized;
00029 }
00030
00031 unsigned int Seize::getLastMemberSeized() const {
00032     return _lastMemberSeized;
00033 }
00034
00035 void Seize::setSaveAttribute(std::string _saveAttribute) {
00036     this->_saveAttribute = _saveAttribute;
00037 }
00038
00039 std::string Seize::getSaveAttribute() const {
00040     return _saveAttribute;
00041 }
00042
00043 void Seize::setRule(Resource::ResourceRule _rule) {
00044     this->_rule = _rule;
00045 }
00046
00047 Resource::ResourceRule Seize::getRule() const {

```

```
00048     return _rule;
00049 }
00050
00051 void Seize::setQuantity(std::string _quantity) {
00052     this->_quantity = _quantity;
00053 }
00054
00055 std::string Seize::getQuantity() const {
00056     return _quantity;
00057 }
00058
00059 void Seize::setResourceType(Resource::ResourceType
00060     _resourceType) {
00061     this->_resourceType = _resourceType;
00062 }
00063 Resource::ResourceType Seize::getResourceType() const {
00064     return _resourceType;
00065 }
00066
00067 void Seize::setPriority(unsigned short _priority) {
00068     this->_priority = _priority;
00069 }
00070
00071 unsigned short Seize::getPriority() const {
00072     return _priority;
00073 }
00074
00075 void Seize::setAllocationType(unsigned int _allocationType) {
00076     this->_allocationType = _allocationType;
00077 }
00078
00079 unsigned int Seize::getAllocationType() const {
00080     return _allocationType;
00081 }
00082
00083 void Seize::setQueueName(std::string queueName) throw () {
00084     Queue* queue = dynamic_cast<Queue*>(_parentModel->
00085         elements()->element(Util::TypeOf<Queue>(), queueName));
00086     if (queue != nullptr) {
00087         _queue = queue;
00088     } else {
00089         throw std::invalid_argument("Queue does not exist");
00090     }
00091 }
00092 void Seize::_handlerForResourceEvent(Resource* resource) {
00093     Waiting* first = _queue->first();
00094     if (first != nullptr) { // there are entities waiting in the queue
00095         unsigned int quantity = _parentModel->parseExpression(this->_quantity);
00096         if ((resource->getCapacity() - resource->getNumberBusy()) >= quantity) { // enough quantity to seize
00097             double tnow = _parentModel->simulation()->
00098                 simulatedTime();
00099             resource->seize(quantity, tnow);
00100             _parentModel->futureEvents()->insert(new
00101                 Event(tnow, first->getEntity(), this->nextComponents()->
00102                     frontConnection()));
00103             _queue->removeElement(first);
00104             _parentModel->tracer()->traceSimulation(tnow, first->
00105                 getEntity(), this, "Waiting entity " + std::to_string(first->getEntity()->
00106                     entityNumber()) + " now seizes " + std::to_string(quantity) + " elements of resource \\" +
00107                     resource->name() + "\\");
00108     }
00109 }
00110
00111 void Seize::setResourceName(std::string resourceName) throw () {
00112     Resource* resource = dynamic_cast<Resource*>(_parentModel->
00113         elements()->element(Util::TypeOf<Resource>(), resourceName));
00114     if (resource != nullptr) {
00115         _resource = resource;
00116     } else {
00117         throw std::invalid_argument("Resource does not exist");
00118     }
00119
00120 std::string Seize::getResourceName() const {
00121     return _resource->name();
00122 }
00123
00124 void Seize::setResource(Resource* resource) {
```

```

00125     this->_resource = resource;
00126     _resource->addResourceEventHandler(Resource::SetResourceEventHandler<Seize>(&
00127         Seize:::_handlerForResourceEvent, this));
00128 }
00129 Resource* Seize::getResource() const {
00130     return _resource;
00131 }
00132
00133 void Seize::setQueue(Queue* queue) {
00134     this->_queue = queue;
00135 }
00136
00137 Queue* Seize::getQueue() const {
00138     return _queue;
00139 }
00140 void Seize::_execute(Entity* entity) {
00141     /* \todo: +: not implemented yet */
00142     Resource* resource = nullptr;
00143     if (this->_resourceType == Resource::ResourceType::SET) {
00144         /* \todo: +: not implemented yet */
00145     } else {
00146         resource = this->_resource;
00147     }
00148     unsigned int quantity = _parentModel->parseExpression(this->_quantity);
00149     if (resource->getCapacity() - resource->getNumberBusy() < quantity) { // not
00150         enough free quantity to allocate. Entity goes to the queue
00151         WaitingResource* waitingRec = new WaitingResource(entity, this,
00152             _parentModel->simulation()->simulatedTime(), quantity);
00153         this->_queue->insertElement(waitingRec); // ->list()->insert(waitingRec);
00154         _parentModel->tracer()->traceSimulation(
00155             _parentModel->simulation()->simulatedTime(), entity, this, "Entity
00156             starts to wait for resource in queue '" + _queue->name() + "' with " + std::to_string(_queue->
00157             size()) + " elements");
00158     } else { // alocate the resource
00159         _parentModel->tracer()->traceSimulation(
00160             _parentModel->simulation()->simulatedTime(), entity, this, "Entity
00161             seizes " + std::to_string(quantity) + " elements of resource '" + resource->name() + "' (capacity:" +
00162             std::to_string(resource->getCapacity()) + ", numberbusy:" + std::to_string(resource->
00163             getNumberBusy()) + ")");
00164         resource->seize(quantity, _parentModel->simulation()->
00165             simulatedTime());
00166         _parentModel->sendEntityToComponent(entity, this->
00167             nextComponents()->frontConnection(), 0.0);
00168     }
00169 }
00170
00171 void Seize::_initBetweenReplications() {
00172     this->_lastMemberSeized = 0;
00173     this->_queue->initBetweenReplications();
00174     this->_resource->initBetweenReplications();
00175 }
00176
00177 bool Seize::_loadInstance(std::map<std::string, std::string>* fields) {
00178     bool res = ModelComponent::_loadInstance(fields);
00179     if (res) {
00180         this->_allocationType = std::stoi((*(fields->find("allocationType"))).second);
00181         this->_priority = std::stoi((*(fields->find("priority"))).second);
00182         this->_quantity = ((*(fields->find("quantity"))).second);
00183         this->_resourceType = static_cast<Resource::ResourceType> (std::stoi((*(fields->
00184             find("resourceType"))).second));
00185         this->_rule = static_cast<Resource::ResourceRule> (std::stoi((*(fields->find("rule"))).second));
00186         this->_saveAttribute = ((*(fields->find("saveAttribute"))).second);
00187         //Util::identitifcation queueId = std::stoi((*(fields->find("queueId"))).second);
00188         //Queue* queue = dynamic_cast<Queue*> (_model->elements()->element(Util::TypeOf<Queue>(), queueId));
00189         std::string queueName = ((*(fields->find("queueName"))).second);
00190         Queue* queue = dynamic_cast<Queue*> (_parentModel->
00191             elements()->element(Util::TypeOf<Queue>(), queueName));
00192         this->_queue = queue;
00193         //Util::identitifcation resourceId = std::stoi((*(fields->find("resourceId"))).second);
00194         //Resource* resource = dynamic_cast<Resource*> (_model->elements()->element(Util::TypeOf<Resource>(),
00195             resourceId));
00196         std::string resourceName = ((*(fields->find("resourceName"))).second);
00197         Resource* resource = dynamic_cast<Resource*> (_parentModel->
00198             elements()->element(Util::TypeOf<Resource>(), resourceName));
00199         this->_resource = resource;
00200         _resource->addResourceEventHandler(Resource::SetResourceEventHandler<Seize>(&
00201             Seize:::_handlerForResourceEvent, this));
00202     }
00203     return res;
00204 }
00205
00206 std::map<std::string, std::string>* Seize::_saveInstance() {
00207     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance

```

```

(); //Util::TypeOf<Seize>());
0014     fields->emplace("allocationType", std::to_string(this->_allocationType));
0015     fields->emplace("priority=", std::to_string(this->_priority));
0016     fields->emplace("quantity", this->_quantity);
0017     fields->emplace("queueId", std::to_string(this->_queue->id()));
0018     fields->emplace("queueName", (this->_queue->name()));
0019     fields->emplace("resourceType", std::to_string(static_cast<int> (this->_resourceType)));
0020     fields->emplace("resourceId", std::to_string(this->_resource->id()));
0021     fields->emplace("resourceName", (this->_resource->name()));
0022     fields->emplace("rule", std::to_string(static_cast<int> (this->_rule)));
0023     fields->emplace("saveAttribute", this->_saveAttribute);
0024     return fields;
0025 }
0026
0027 bool Seize::_check(std::string* errorMessage) {
0028     bool resultAll = true;
0029     resultAll &= _parentModel->checkExpression(_quantity, "quantity",
0030         errorMessage);
0031     resultAll &= _parentModel->elements()->check(Util::TypeOf<Resource>(),
0032         _resource, "Resource", errorMessage);
0033     resultAll &= _parentModel->elements()->check(Util::TypeOf<Queue>(), _queue, "Queue",
0034         errorMessage);
0035     resultAll &= _parentModel->elements()->check(Util::TypeOf<Attribute>(),
0036         _saveAttribute, "SaveAttribute", false, errorMessage);
0037     return resultAll;
0038 }
0039
0040 PluginInformation* Seize::GetPluginInformation() {
0041     PluginInformation* info = new PluginInformation(Util::TypeOf<Seize>()
0042         , &Seize::LoadInstance);
0043     info->insertDynamicLibFileDependence("queue.so");
0044     info->insertDynamicLibFileDependence("resource.so");
0045     return info;
0046 }
0047
0048 ModelComponent* Seize::LoadInstance(Model* model,
0049     std::map<std::string, std::string>* fields) {
0050     Seize* newComponent = new Seize(model);
0051     try {
0052         newComponent->_loadInstance(fields);
0053     } catch (const std::exception& e) {
0054     }
0055     return newComponent;
0056 }
0057
0058 }
```

7.395 Seize.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Model.h"
#include "Resource.h"
#include "Queue.h"
#include "Plugin.h"
```

Classes

- class WaitingResource
- class Seize

7.396 Seize.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
```

```

00006
00007 /*
00008 * File: Seize.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Agosto de 2018, 16:17
00012 */
00013
00014 #ifndef SEIZE_H
00015 #define SEIZE_H
00016
00017 #include <string>
00018 #include "ModelComponent.h"
00019 #include "Model.h"
00020 #include "Resource.h"
00021 #include "Queue.h"
00022 #include "Plugin.h"
00023
00024 class WaitingResource : public Waiting {
00025 public:
00026
00027     WaitingResource(Entity* entity, ModelComponent* component, double
00028         timeStartedWaiting, unsigned int quantity) : Waiting(entity, component, timeStartedWaiting) {
00029         _quantity = quantity;
00030     }
00031     WaitingResource(const WaitingResource& orig) :
00032         Waiting(orig)
00033     {
00034         virtual ~WaitingResource() {
00035     }
00036     public:
00037
00038         virtual std::string show() {
00039             return Waiting::show() +
00040                 ",quantity=" + std::to_string(this->_quantity);
00041         }
00042     public:
00043
00044         unsigned int getQuantity() const {
00045             return _quantity;
00046         }
00047     private:
00048         unsigned int _quantity;
00049     };
00050
00125 class Seize : public ModelComponent {
00126 public:
00127     Seize(Model* model, std::string name="");
00128     virtual ~Seize() = default;
00129 public:
00130     virtual std::string show();
00131 public:
00132     static PluginInformation* GetPluginInformation();
00133     static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>
00134     * fields);
00134 public: // get & set
00135     void setLastMemberSeized(unsigned int _lastMemberSeized);
00136     unsigned int getLastMemberSeized() const;
00137     void setSaveAttribute(std::string _saveAttribute);
00138     std::string getSaveAttribute() const;
00139     void setRule(Resource::ResourceRule _rule);
00140     Resource::ResourceRule getRule() const;
00141     void setQuantity(std::string _quantity);
00142     std::string getQuantity() const;
00143     void setResourceType(Resource::ResourceType _resourceType);
00144     Resource::ResourceType getResourceType() const;
00145     void setPriority(unsigned short _priority);
00146     unsigned short getPriority() const;
00147     void setAllocationType(unsigned int _allocationType);
00148     unsigned int getAllocationType() const;
00149     // indirect access to Queue* and Resource*
00150     void setResourceName(std::string _resourceName) throw();
00151     std::string getResourceName() const;
00152     void setQueueName(std::string queueName) throw();
00153     std::string getQueueName() const;
00154     void setResource(Resource* resource);
00155     Resource* getResource() const;
00156     void setQueue(Queue* queue);
00157     Queue* getQueue() const;
00158 protected:
00159     virtual void _execute(Entity* entity);
00160     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00161     virtual void _initBetweenReplications();
00162     virtual std::map<std::string, std::string>* _saveInstance();
00163     virtual bool _check(std::string* errorMessage);

```

```

00164 private:
00165     void _handlerForResourceEvent(Resource* resource);
00166 private:
00167     unsigned int _allocationType = 0; // uint ? enum?
00168     unsigned short _priority = 0;
00169     Resource::ResourceType _resourceType =
00170         Resource::ResourceType::RESOURCE;
00171     std::string _quantity = "1";
00172     Resource::ResourceRule _rule =
00173         Resource::ResourceRule::SMALLESTBUSY;
00174     std::string _saveAttribute = "";
00175     //std::string _resourceName = "Resource 1"; // trying to access resource and queue indirectly
00176     //std::string _queueName;
00177 private: // not gets or sets
00178     Queue* _queue; // usually has a queue, but not always (it could be a hold) /* \todo: Evaluate if
00179     is better to associate queue to seize or to the resource */
00180     Resource* _resource; // usually has a resource, but not always (it could be a set)
00181     //Set* _set;
00182     unsigned int _lastMemberSeized = 0;
00183 };
00184 #endif /* SEIZE_H */
00185

```

7.397 Separate.cpp File Reference

```
#include "Separate.h"
#include "Model.h"
```

7.398 Separate.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Separate.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #include "Separate.h"
00015
00016 #include "Model.h"
00017
00018 Separate::Separate(Model* model, std::string name) :
00019     ModelComponent(model, Util::TypeOf<Separate>(),name) {
00020
00021
00022     std::string Separate::show() {
00023         return ModelComponent::show() + "";
00024     }
00025
00026     ModelComponent* Separate::LoadInstance(Model* model,
00027         std::map<std::string, std::string>* fields) {
00028         Separate* newComponent = new Separate(model);
00029         try {
00030             newComponent->_loadInstance(fields);
00031         } catch (const std::exception& e) {
00032
00033             return newComponent;
00034         }
00035
00036     void Separate::_execute(Entity* entity) {
00037         //Entity* cloned = new Entity(entity);
00038         this->_parentModel->sendEntityToComponent(entity,
00039             nextComponents()->frontConnection(), 0.0);
00040         //this->_parentModel->sendEntityToComponent(cloned, nextComponents()->getConnectionAtRank(1), 0.0);
00041
00042     bool Separate::_loadInstance(std::map<std::string, std::string>* fields) {

```

```

00043     bool res = ModelComponent::_loadInstance(fields);
00044     if (res) {
00045         //...
00046     }
00047     return res;
00048 }
00049
00050 void Separate::_initBetweenReplications() {
00051 }
00052
00053 std::map<std::string, std::string>* Separate::_saveInstance() {
00054     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00055     ();
00056     //...
00057     return fields;
00058 }
00059 bool Separate::_check(std::string* errorMessage) {
00060     bool resultAll = true;
00061     //...
00062     return resultAll;
00063 }
00064
00065 PluginInformation* Separate::GetPluginInformation(){
00066     PluginInformation* info = new PluginInformation(
00067         Util::TypeOf<Separate>(), &Separate::LoadInstance);
00068     // ...
00069     return info;
00070 }

```

7.399 Separate.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class Separate

7.400 Separate.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Separate.h
00009  * Author: rlcancian
00010  *
00011  * Created on 03 de Junho de 2019, 15:14
00012 */
00013
00014 #ifndef SEPARATE_H
00015 #define SEPARATE_H
00016
00017 #include "ModelComponent.h"
00018
00065 class Separate : public ModelComponent {
00066 public: // constructors
00067     Separate(Model* model, std::string name(""));
00068     virtual ~Separate() = default;
00069 public: // virtual
00070     virtual std::string show();
00071 public: // static
00072     static PluginInformation* GetPluginInformation();
00073     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00074                                         std::string>* fields);
00074 protected: // virtual
00075     virtual void _execute(Entity* entity);
00076     virtual void _initBetweenReplications();
00077     virtual bool _loadInstance(std::map<std::string, std::string>* fields);

```

```

00078     virtual std::map<std::string, std::string>* _saveInstance();
00079     virtual bool _check(std::string* errorMessage);
00080 private: // methods
00081 private: // attributes 1:1
00082 private: // attributes 1:n
00083 };
00084
00085
00086 #endif /* SEPARATE_H */
00087

```

7.401 Sequence.cpp File Reference

```

#include "Sequence.h"
#include "Attribute.h"
#include "Model.h"

```

7.402 Sequence.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Sequence.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #include "Sequence.h"
00015 #include "Attribute.h"
00016 #include "Model.h"
00017
00018 Sequence::Sequence(Model* model, std::string name) :
    ModelElement(model, Util::TypeOf<Sequence>(), name) {
00019 }
00020
00021
00022 std::string Sequence::show() {
00023     std::string msg = ModelElement::show();
00024     return msg;
00025 }
00026
00027
00028 PluginInformation* Sequence::GetPluginInformation() {
00029     PluginInformation* info = new PluginInformation(
00030         Util::TypeOf<Sequence>(), &Sequence::LoadInstance);
00031     return info;
00032 }
00033 ModelElement* Sequence::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00034     Sequence* newElement = new Sequence(model);
00035     try {
00036         newElement->_loadInstance(fields);
00037     } catch (const std::exception& e) {
00038     }
00039 }
00040     return newElement;
00041 }
00042
00043 bool Sequence::_loadInstance(std::map<std::string, std::string>* fields) {
00044     bool res = ModelElement::_loadInstance(fields);
00045     if (res) {
00046         try {
00047             } catch (...) {
00048             }
00049     }
00050     return res;
00051 }
00052
00053 std::map<std::string, std::string>* Sequence::_saveInstance() {

```

```

00054     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00055     //Util::TypeOf<Sequence>());
00056     return fields;
00057 }
00058 bool Sequence::_check(std::string* errorMessage) {
00059     /* include attributes needed */
00060     std::vector<std::string> neededNames = {"Entity.Sequence"};
00061     std::string neededName;
00062     for (unsigned int i = 0; i < neededNames.size(); i++) {
00063         neededName = neededNames[i];
00064         if (_parentModel->elements()->element(Util::TypeOf<Attribute>(), neededName)
00065             == nullptr) {
00066             Attribute* attr1 = new Attribute(_parentModel, neededName);
00067             //_parentModel->insert(attr1);
00068         }
00069     }
00070     //
00071     return true;
00072 }
```

7.403 Sequence.h File Reference

```
#include "ModelElement.h"
#include "ElementManager.h"
#include "PluginInformation.h"
#include "Station.h"
```

Classes

- class [Sequence](#)
- class [Sequence::SequenceStep](#)

7.404 Sequence.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Sequence.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #ifndef SEQUENCE_H
00015 #define SEQUENCE_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "PluginInformation.h"
00020 #include "Station.h"
00021
00022 class Sequence : public ModelElement {
00023 public:
00024     class SequenceStep {
00025     public:
00026         Station* _station;
00027         std::list<std::string>* _assignments;
00028     };
00029 public:
00030     Sequence(Model* model, std::string name = "");
00031     virtual ~Sequence() = default;
00032     public:
00033         virtual std::string show();
00034     public: // static
```

```

00060     static PluginInformation* GetPluginInformation();
00061     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00062                                         std::string>* fields);
00063 public:
00064     protected:
00065         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00066         virtual std::map<std::string, std::string>* _saveInstance();
00067         virtual bool _check(std::string* errorMessage);
00068     private:
00069 };
00070
00071 #endif /* SEQUENCE_H */
00072

```

7.405 Set.cpp File Reference

```
#include "Set.h"
```

7.406 Set.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Set.cpp
00009  * Author:  rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:11
00012 */
00013
00014 #include "Set.h"
00015
00016 Set::Set(Model* model, std::string name) : ModelElement(model,
00017   Util::TypeOf<Set>(), name) {
00018
00019
00020 std::string Set::show() {
00021     return ModelElement::show() +
00022         "";
00023 }
00024
00025 void Set::setSetOfType(std::string _setOfType) {
00026     this->_setOfType = _setOfType;
00027 }
00028
00029 std::string Set::getSetOfType() const {
00030     return _setOfType;
00031 }
00032
00033 List<ModelElement*>* Set::getElementSet() const {
00034     return _elementSet;
00035 }
00036
00037 PluginInformation* Set::GetPluginInformation() {
00038     PluginInformation* info = new PluginInformation(Util::TypeOf<Set>(),
00039         &Set::LoadInstance);
00040     return info;
00041 }
00042 ModelElement* Set::LoadInstance(Model* model, std::map<std::string,
00043                                     std::string>* fields) {
00044     Set* newElement = new Set(model);
00045     try {
00046         newElement->_loadInstance(fields);
00047     } catch (const std::exception& e) {
00048     }
00049     return newElement;
00050 }
00051
00052 bool Set::_loadInstance(std::map<std::string, std::string>* fields) {

```

```

00053     bool res = ModelElement::_loadInstance(fields);
00054     if (res) {
00055         try {
00056             //this->_attributeName = (*fields->find("attributeName")).second;
00057             //this->_orderRule = static_cast<OrderRule> (std::stoi((*fields->find("orderRule")).second));
00058         } catch (...) {
00059         }
00060     }
00061     return res;
00062 }
00063
00064 std::map<std::string, std::string>* Set::_saveInstance() {
00065     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00066     //Util::TypeOf<Set>());
00067     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00068     //fields->emplace("attributeName", this->_attributeName);
00069     return fields;
00070 }
00071 bool Set::_check(std::string* errorMessage) {
00072     bool resultAll = true;
00073     // resultAll |= ...
00074     return resultAll;
00075 }
00076
00077 ParserChangesInformation*
00078     Set::_getParserChangesInformation() {
00079         ParserChangesInformation* changes = new
00080             ParserChangesInformation();
00081         //changes->getProductionToAdd()->insert(...);
00082         //changes->getTokensToAdd()->insert(...);
00083         return changes;
00082 }
00083

```

7.407 Set.h File Reference

```

#include "ModelElement.h"
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"

```

Classes

- class [Set](#)

7.408 Set.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Set.h
00009  * Author: rlcancian
00010 *
00011  * Created on 03 de Junho de 2019, 15:11
00012 */
00013
00014 #ifndef SET_H
00015 #define SET_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "ParserChangesInformation.h"
00020 #include "PluginInformation.h"
00021
00055 class Set: public ModelElement {
00056 public:

```

```

00057     Set(Model* model, std::string name(""));
00058     virtual ~Set() = default;
00059 public: // static
00060     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00061                                         std::string>* fields);
00062     static PluginInformation* GetPluginInformation();
00063 public:
00064     virtual std::string show();
00065     void setSetOfType(std::string _setOfType);
00066     std::string getSetOfType() const;
00067     List<ModelElement*>* getElementSet() const;
00068 protected: // must be overridden by derived classes
00069     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00070     virtual std::map<std::string, std::string>* _saveInstance();
00071 protected: // could be overridden by derived classes
00072     virtual bool _check(std::string* errorMessage);
00073     virtual ParserChangesInformation*
00074         _getParserChangesInformation();
00075 private:
00076     //ElementManager* _elems;
00077     List<ModelElement*>* _elementSet = new
00078         List<ModelElement*>();
00079     std::string _setOfType;
00080 };
00081 #endif /* SET_H */
00082

```

7.409 Signal.cpp File Reference

```
#include "Signal.h"
#include "Model.h"
```

7.410 Signal.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Signal.cpp
00009  * Author: rlcancian
00010  *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #include "Signal.h"
00015
00016 #include "Model.h"
00017
00018 Signal::Signal(Model* model, std::string name) :
00019     ModelComponent(model, Util::TypeOf<Signal>(), name) {
00020
00021
00022     std::string Signal::show() {
00023         return ModelComponent::show() + "";
00024     }
00025
00026     ModelComponent* Signal::LoadInstance(Model* model,
00027                                         std::map<std::string, std::string>* fields) {
00028         Signal* newComponent = new Signal(model);
00029         try {
00030             newComponent->_loadInstance(fields);
00031         } catch (const std::exception& e) {
00032         }
00033         return newComponent;
00034     }
00035
00036     void Signal::_execute(Entity* entity) {
00037         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the

```

```

        entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
00039         nextComponents()->frontConnection(), 0.0);
00040
00041     bool Signal::_loadInstance(std::map<std::string, std::string>* fields) {
00042         bool res = ModelComponent::_loadInstance(fields);
00043         if (res) {
00044             //...
00045         }
00046         return res;
00047     }
00048
00049     void Signal::_initBetweenReplications() {
00050 }
00051
00052     std::map<std::string, std::string>* Signal::_saveInstance() {
00053         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00054         ();
00055         //...
00056         return fields;
00057     }
00058     bool Signal::_check(std::string* errorMessage) {
00059         bool resultAll = true;
00060         //...
00061         return resultAll;
00062     }
00063
00064     PluginInformation* Signal::GetPluginInformation(){
00065         PluginInformation* info = new PluginInformation(Util::TypeOf<Signal>(
00066             ), &Signal::LoadInstance);
00067         // ...
00068         return info;
00069     }
00070

```

7.411 Signal.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Signal](#)

7.412 Signal.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Signal.h
00009  * Author: rlcancian
00010  *
00011  * Created on 03 de Junho de 2019, 15:20
00012 */
00013
00014 #ifndef SIGNAL_H
00015 #define SIGNAL_H
00016
00017 #include "ModelComponent.h"
00018
00038 class Signal : public ModelComponent {
00039     public: // constructors
00040         Signal(Model* model, std::string name="");
00041         virtual ~Signal() = default;
00042     public: // virtual
00043         virtual std::string show();
00044     public: // static

```

```

00045     static PluginInformation* GetPluginInformation();
00046     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00047                                         std::string>* fields);
00047 protected: // virtual
00048     virtual void _execute(Entity* entity);
00049     virtual void _initBetweenReplications();
00050     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00051     virtual std::map<std::string, std::string>* _saveInstance();
00052     virtual bool _check(std::string* errorMessage);
00053 private: // methods
00054 private: // attributes 1:1
00055 private: // attributes 1:n
00056 };
00057
00058
00059 #endif /* SIGNAL_H */
00060

```

7.413 SimulationControl.cpp File Reference

```
#include "SimulationControl.h"
```

7.414 SimulationControl.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationControl.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:01
00012 */
00013
00014 #include "SimulationControl.h"
00015
00016 SimulationControl::SimulationControl(std::string type, std::string name
00017 , GetterMember getterMember, SetterMember setterMember) :
00018     SimulationResponse(type, name, getterMember) {
00019     this->_type = type;
00020     this->_setMemberFunction = setterMember;
00021 }
00022 std::string SimulationControl::show() {
00023     return "name="+this->_name+", type="+this->_type;
00024 }
00025
00026 void SimulationControl::setValue(double value) {
00027     this->_setMemberFunction(value);
00028 }
00029

```

7.415 SimulationControl.h File Reference

```
#include "SimulationResponse.h"
#include "DefineGetterSetter.h"
```

Classes

- class [SimulationControl](#)

7.416 SimulationControl.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationControl.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:01
00012 */
00013
00014 #ifndef SIMULATIONCONTROL_H
00015 #define SIMULATIONCONTROL_H
00016
00017 #include "SimulationResponse.h"
00018 #include "DefineGetterSetter.h"
00019
00020 class SimulationControl : public SimulationResponse {
00021 public:
00022     SimulationControl(std::string type, std::string name,
00023                         GetterMember getterMember, SetterMember setterMember);
00024     virtual ~SimulationControl() = default;
00025 public:
00026     std::string show();
00027 public:
00028     void setValue(double value);
00029 private:
00030     SetterMember _setMemberFunction;
00031 };
00032
00033 #endif /* SIMULATIONCONTROL_H */
00034
00035
00036

```

7.417 SimulationReporter_if.h File Reference

```
#include "List.h"
```

Classes

- class [SimulationReporter_if](#)

7.418 SimulationReporter_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationReporter_if.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 8 de Agosto de 2018, 10:56
00012 */
00013
00014 #ifndef SIMULATIONREPORTER_IF_H
00015 #define SIMULATIONREPORTER_IF_H
00016
00017 #include "List.h"
00018 //##include "StatisticsCollector.h"
00019
00020 class SimulationReporter_if {
00021 public:
00022     virtual void showReplicationStatistics() = 0;
00023     virtual void showSimulationStatistics() = 0; //List<StatisticsCollector*>*
00024     cstatsSimulation) = 0;
00025
00026 #endif /* SIMULATIONREPORTER_IF_H */
00027

```

7.419 SimulationReporterDefaultImpl1.cpp File Reference

```
#include "SimulationReporterDefaultImpl1.h"
#include <assert.h>
#include <iomanip>
#include <iostream>
```

7.420 SimulationReporterDefaultImpl1.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: SimulationReporterDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 8 de Agosto de 2018, 10:59
00012 */
00013
00014 #include "SimulationReporterDefaultImpl1.h"
00015 #include <assert.h>
00016 #include <iomanip>
00017 #include <iostream>
00018
00019 SimulationReporterDefaultImpl1::SimulationReporterDefaultImpl1
00020     (ModelSimulation* simulation, Model* model,
00021      List<ModelElement*>* statsCountersSimulation) {
00022     _simulation = simulation;
00023     _model = model;
00024     _statsCountersSimulation = statsCountersSimulation;
00025 }
00026 void SimulationReporterDefaultImpl1::showReplicationStatistics
00027     ()
00028     {
00029         _model->tracer()->traceReport("");
00030         _model->tracer()->traceReport("Begin of Report for replication " + std::to_string(
00031             _simulation->currentReplicationNumber()) + " of " + std::to_string(_model->
00032             infos()->numberOfReplications()));
00033         /* \todo: StatisticsCollector and Counter should NOT be special classes. It should iterate classes
00034             looking for classes that can generate reports.
00035             StatisticsCollector and Counter should override an inherited attribute from ModelElement to specify
00036             they generate report information
00037             look for _generateReportInformation = true; using bool generateReportInformation() const;
00038             */
00039         const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00040         const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00041         // runs over all elements and list the statistics for each one, and then the statistics with no parent
00042         Util::IncIndent();
00043         // copy the list of statistics and counters into a single new list
00044         std::list<ModelElement*>* statisticsAndCounters = new std::list<ModelElement*>(*(_model->
00045             elements()->elementList(UtilTypeOfStatisticsCollector)->list()));
00046         std::list<ModelElement*>* counters = new std::list<ModelElement*>(*(_model->
00047             elements()->elementList(UtilTypeOfCounter)->list()));
00048         statisticsAndCounters->merge(*counters);
00049         // organizes statistics into a map of maps
00050         std::map< std::string, std::map<std::string, std::list<ModelElement*>>>* mapMapTypeStat = new
00051             std::map<std::string, std::map<std::string, std::list<ModelElement*>>>();
00052
00053         for (std::list<ModelElement*>::iterator it = statisticsAndCounters->begin(); it !=
00054             statisticsAndCounters->end(); it++) {
00055             std::string parentName, parentTypename;
00056             ModelElement* statOrCnt = (*it);
00057             //std::cout << statOrCnt->getName() << ":" << statOrCnt->getTypename() << std::endl;
00058             if ((*it)->classname() == UtilTypeOfStatisticsCollector) {
00059                 StatisticsCollector* stat = dynamic_cast<
00060                     StatisticsCollector*> (statOrCnt);
00061                 parentName = stat->getParent()->name();
00062                 parentTypename = stat->getParent()->classname();
00063             } else {
00064                 if ((*it)->classname() == UtilTypeOfCounter) {
00065                     Counter* cnt = dynamic_cast<Counter*> (statOrCnt);
00066                     parentName = cnt->getParent()->name();
00067                     parentTypename = cnt->getParent()->classname();
00068                 }
00069             }
00070         }
00071     }
```

```

00058         }
00059     }
00060     // look for key=parentTypename
00061     std::map<std::string, std::map<std::string, std::list<ModelElement*>>>::iterator mapMapIt =
00062     mapMapTypeStat->find(parentTypename);
00063     if (mapMapIt == mapMapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00064         std::pair<std::string, std::map<std::string, std::list<ModelElement*>>> newPair = new
00065         std::pair<std::string, std::map<std::string, std::list<ModelElement*>>>(parentTypename, new std::map<std::string,
00066             std::list<ModelElement*>>>());
00067         mapMapTypeStat->insert(*newPair);
00068         mapMapIt = mapMapTypeStat->find(parentTypename); // find again. Now it will.
00069     }
00070     //assert(mapMapIt != mapMapTypeStat->end());
00071     std::map<std::string, std::list<ModelElement*>>> mapTypeStat = (*mapMapIt).second;
00072     assert(mapTypeStat != nullptr);
00073     // look for key=parentName
00074     std::map<std::string, std::list<ModelElement*>>>::iterator mapIt = mapTypeStat->find(parentName);
00075     if (mapIt == mapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00076         std::pair<std::string, std::list<ModelElement*>>> newPair = new std::pair<std::string,
00077             std::list<ModelElement*>>>(parentName, new std::list<ModelElement*>());
00078         mapTypeStat->insert(*newPair);
00079         mapIt = mapTypeStat->find(parentName); // find again. Now it will.
00080     }
00081     // get the list and insert the stat in that list
00082     std::list<ModelElement*>> listStatAndCount = (*mapIt).second;
00083     assert(listStatAndCount != nullptr);
00084     listStatAndCount->insert(listStatAndCount->end(), statOrCnt);
00085     _model->getTraceManager()->traceReport(parentTypename + " -> " + parentName + " -> " + stat->show());
00086 }
00087 // now runs over that map of maps showing the statistics
00088 for (auto const mapmapItem : *mapMapTypeStat) {
00089     _model->tracer()->traceReport("Statistics for " + mapmapItem.first + ":" );
00090     Util::IncIndent();
00091     {
00092         for (auto const mapItem : *(mapmapItem.second)) {
00093             _model->tracer()->traceReport(Util::SetW("name", _nameW) +
00094                 Util::SetW("elems", _w) + Util::SetW("min", _w) +
00095                 Util::SetW("max", _w) + Util::SetW("average", _w) +
00096                 Util::SetW("variance", _w) + Util::SetW("stddev", _w) +
00097                 Util::SetW("varCoef", _w) + Util::SetW("confInterv", _w) +
00098                 Util::SetW("confLevel", _w));
00099         for (ModelElement * const item : *(mapItem.second)) {
00100             if (item->classname() == UtilTypeOfStatisticsCollector) {
00101                 Statistics_if* stat = dynamic_cast<
00102                     StatisticsCollector*> (item)->getStatistics();
00103                 _model->tracer()->traceReport(
00104                     Util::TraceLevel::report,
00105                     Util::SetW(item->name() + std::string(_nameW, '.'), _nameW - 1) + " " +
00106                     Util::SetW(std::to_string(stat->numElements()), _w) +
00107                     Util::SetW(std::to_string(stat->min()), _w) +
00108                     Util::SetW(std::to_string(stat->max()), _w) +
00109                     Util::SetW(std::to_string(stat->average()), _w) +
00110                     Util::SetW(std::to_string(stat->variance()), _w) +
00111                     Util::SetW(std::to_string(stat->stddeviation()), _w) +
00112                     Util::SetW(std::to_string(stat->variationCoef()), _w) +
00113                     Util::SetW(std::to_string(stat->
00114                         halfWidthConfidenceInterval()), _w) +
00115                     Util::SetW(std::to_string(stat->getConfidenceLevel()), _w)
00116                 );
00117             } else {
00118                 if (item->classname() == UtilTypeOfCounter) {
00119                     Counter* count = dynamic_cast<Counter*> (item);
00120                     _model->tracer()->traceReport(
00121                         Util::TraceLevel::report,
00122                         Util::SetW(count->name() + std::string(_nameW, '.'), _nameW - 1) + " " +
00123                         Util::SetW(std::to_string(count->getCountValue()), _w)
00124                     );
00125             }
00126             Util::DecIndent();
00127         _model->tracer()->traceReport("End of Report for replication " + std::to_string(
00128             _simulation->currentReplicationNumber()) + " of " + std::to_string(_model->
00129             infos()->numberOfReplications()));
00130         _model->tracer()->traceReport("-----");
00131     }
00132 }
00133 }
```

```

00131 void SimulationReporterDefaultImpl1::showSimulationStatistics
00132   () //>List<StatisticsCollector*>* cstatsSimulation) {
00133     _model->tracer()->traceReport("");
00134     _model->tracer()->traceReport("Begin of Report for Simulation (based on " +
00135       std::to_string(_model->infos()->numberOfReplications()) + " replications)");
00136     const std::string UtilTypeOfStatisticsCollector = Util::TypeOf<StatisticsCollector>();
00137     const std::string UtilTypeOfCounter = Util::TypeOf<Counter>();
00138     // runs over all elements and list the statistics for each one, and then the statistics with no parent
00139     Util::IncIndent();
00140     // COPY the list of statistics and counters into a single new list
00141     //std::list<ModelElement*>* statisticsAndCounters = //new
00142     std::list<ModelElement*>(*(this->_statsCountersSimulation->list()));
00143     // organizes statistics into a map of maps
00144     std::map< std::string, std::map<std::string, std::list<ModelElement*>>>* mapMapTypeStat = new
00145     std::map<std::string, std::map<std::string, std::list<ModelElement*>>>();
00146
00147     for (std::list<ModelElement*>::iterator it = _statsCountersSimulation->list()->begin(); it != _statsCountersSimulation->list()->end(); it++) {
00148       std::string parentName, parentTypename;
00149       ModelElement* statOrCnt = (*it);
00150       //std::cout << statOrCnt->getName() << ":" << statOrCnt->getTypename() << std::endl;
00151       if ((*it)->classname() == UtilTypeOfStatisticsCollector) {
00152         StatisticsCollector* stat = dynamic_cast<
00153           StatisticsCollector*> (statOrCnt);
00154         parentName = stat->getParent()->name();
00155         parentTypename = stat->getParent()->classname();
00156       }
00157     }
00158     // look for key=parentTypename
00159     std::map<std::string, std::map<std::string, std::list<ModelElement*>>>::iterator mapMapIt =
00160     mapMapTypeStat->find(parentTypename);
00161     if (mapMapIt == mapMapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00162       std::pair< std::string, std::map<std::string, std::list<ModelElement*>>>* newPair = new
00163       std::pair< std::string, std::map<std::string, std::list<ModelElement*>>>(parentTypename, new std::map<std::string,
00164         std::list<ModelElement*>>());
00165       mapMapTypeStat->insert(*newPair);
00166       mapMapIt = mapMapTypeStat->find(parentTypename); // find again. Now it will.
00167     }
00168     assert(mapMapIt != mapMapTypeStat->end());
00169     std::map<std::string, std::list<ModelElement*>>> mapTypeStat = (*mapMapIt).second;
00170     assert(mapTypeStat != nullptr);
00171     // look for key=parentName
00172     std::map<std::string, std::list<ModelElement*>>>::iterator mapIt = mapTypeStat->find(parentName);
00173     if (mapIt == mapTypeStat->end()) { // parentTypename does not exists in map. Include it.
00174       std::pair< std::string, std::list<ModelElement*>>>* newPair = new std::pair<std::string,
00175         std::list<ModelElement*>>>(parentName, new std::list<ModelElement*>());
00176       mapTypeStat->insert(*newPair);
00177     }
00178     mapIt = mapTypeStat->find(parentName); // find again. Now it will.
00179     // get the list and insert the stat in that list
00180     std::list<ModelElement*>> listStat = (*mapIt).second;
00181     listStat->insert(listStat->end(), statOrCnt);
00182     //_model->getTraceManager()->traceReport(parentTypename + " -> " + parentName + " -> " + stat->show());
00183     // now runs over that map of maps showing the statistics
00184     //int w = 12;
00185     for (auto const mapmapItem : *mapMapTypeStat) {
00186       _model->tracer()->traceReport("Statistics for " + mapmapItem.first + ":" );
00187       Util::IncIndent();
00188     }
00189     for (auto const mapItem : *(mapmapItem.second)) {
00190       _model->tracer()->traceReport(mapItem.first + ":" );
00191       Util::IncIndent();
00192       for (auto const mapItem : *(mapmapItem.second)) {
00193         _model->tracer()->traceReport(Util::SetW("name", _nameW) +
00194           Util::SetW("max", _w) + Util::SetW("average", _w) +
00195           Util::SetW("variance", _w) + Util::SetW("stddev", _w) +
00196           Util::SetW("varCoef", _w) + Util::SetW("confInterv", _w) +
00197           Util::SetW("confLevel", _w));
00198         for (ModelElement * const item : *(mapItem.second)) {
00199           if (item->classname() == UtilTypeOfStatisticsCollector) {
00200             Statistics_if* stat = dynamic_cast<
00201               StatisticsCollector*> (item)->getStatistics();
00202             _model->tracer()->traceReport(
00203               Util::TraceLevel::report,
00204               Util::SetW(item->name() + std::string(_nameW, '.'), _nameW - 1) + " " +
00205               Util::SetW(std::to_string(stat->numElements()), _w) +
00206               Util::SetW(std::to_string(stat->min()), _w) +
00207               Util::SetW(std::to_string(stat->max()), _w) +
00208               Util::SetW(std::to_string(stat->average()), _w) +

```

```

00202     Util::SetW(std::to_string(stat->variance()), _w) +
00203     Util::SetW(std::to_string(stat->stddeviation()), _w) +
00204     Util::SetW(std::to_string(stat->variationCoef()), _w) +
00205     Util::SetW(std::to_string(stat->
00206         halfWidthConfidenceInterval()), _w) +
00207         Util::SetW(std::to_string(stat->getConfidenceLevel()), _w)
00208     } else {
00209         if (item->classname() == UtilTypeOfCounter) {
00210             Counter* cnt = dynamic_cast<Counter*> (item);
00211             _model->tracer()->traceReport(
00212                 Util::TraceLevel::report,
00213                     Util::SetW(cnt->name() + std::string(_nameW, '.'), _nameW - 1) + " " +
00214                     Util::SetW(std::to_string(cnt->getCountValue()), _w)
00215             );
00216         }
00217     }
00218 }
00219 }
00220 Util::DecIndent ();
00221 }
00222 }
00223 Util::DecIndent ();
00224 }
00225
00226 Util::DecIndent ();
00227 _model->tracer()->traceReport("End of Report for Simulation");
00228 }
```

7.421 SimulationReporterDefaultImpl1.h File Reference

```
#include "SimulationReporter_if.h"
#include "ModelSimulation.h"
#include "Model.h"
```

Classes

- class [SimulationReporterDefaultImpl1](#)

7.422 SimulationReporterDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  SimulationReporterDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 8 de Agosto de 2018, 10:59
00012 */
00013
00014 #ifndef SIMULATIONREPORTERDEFAULTIMPL1_H
00015 #define SIMULATIONREPORTERDEFAULTIMPL1_H
00016
00017 #include "SimulationReporter_if.h"
00018 #include "ModelSimulation.h"
00019 #include "Model.h"
00020
00024 class SimulationReporterDefaultImpl1 : public
00025     SimulationReporter_if {
00026     public:
00027         SimulationReporterDefaultImpl1(ModelSimulation* simulation
00028             , Model* model, List<ModelElement*>* statsCountersSimulation);
00029         virtual ~SimulationReporterDefaultImpl1() = default;
00028 public:
00029     virtual void showReplicationStatistics();
00030     virtual void showSimulationStatistics(); //List<StatisticsCollector*>*
00031         cstatsSimulation;
```

```

00031 private:
00032     //List<StatisticsCollector*>* getStatisticsFromParent( std::string parent );
00033 private:
00034     ModelSimulation* _simulation;
00035     Model* _model;
00036 private:
00037     List<ModelElement*>* _statsCountersSimulation;
00038 private:
00039     const unsigned short _w = 12;
00040     const unsigned short _nameW = 40;
00041 };
00042
00043 #endif /* SIMULATIONREPORTERDEFAULTIMPL1_H */
00044

```

7.423 SimulationResponse.cpp File Reference

```

#include "SimulationResponse.h"
#include "AnalysisOfVariance_if.h"

```

7.424 SimulationResponse.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SimulationResponse.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 16:18
00012 */
00013
00014 #include "SimulationResponse.h"
00015 #include "AnalysisOfVariance_if.h"
00016
00017 SimulationResponse::SimulationResponse(std::string type, std::string
    name, GetterMember getterMember) {
00018     _type = type;
00019     _name = name;
00020     _getterMemberFunction = getterMember;
00021 }
00022
00023
00024 std::string SimulationResponse::show() {
00025     return "name='"+this->_name+", type='"+this->_type';
00026 }
00027
00028 std::string SimulationResponse::name() const {
00029     return _name;
00030 }
00031
00032 std::string SimulationResponse::type() const {
00033     return _type;
00034 }
00035
00036 double SimulationResponse::value() {
00037     return this->_getterMemberFunction();
00038 }

```

7.425 SimulationResponse.h File Reference

```

#include <string>
#include "DefineGetterSetter.h"

```

Classes

- class [SimulationResponse](#)

7.426 SimulationResponse.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  SimulationResponse.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 16:18
00012 */
00013
00014 #ifndef SIMULATIONRESPONSE_H
00015 #define SIMULATIONRESPONSE_H
00016
00017 #include <string>
00018 #include "DefineGetterSetter.h"
00019
00023 class SimulationResponse {
00024 public:
00025     SimulationResponse(std::string type, std::string name,
00026                          GetterMember getterMember);
00026     virtual ~SimulationResponse() = default;
00027 public:
00028     std::string show();
00029 public:
00030     double value();
00031     std::string name() const;
00032     std::string type() const;
00033 protected:
00034     std::string _type;
00035     std::string _name;
00036     GetterMember _getterMemberFunction;
00037 };
00038
00039 #endif /* SIMULATIONRESPONSE_H */
00040

```

7.427 SimulationScenario.cpp File Reference

```
#include "SimulationScenario.h"
```

7.428 SimulationScenario.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  SimulationScenario.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 10 de Outubro de 2018, 18:21
00012 */
00013
00014 #include "SimulationScenario.h"
00015
00016 SimulationScenario::SimulationScenario() {
00017 }
00018
00019 bool SimulationScenario::startSimulation(std::string* errorMessage) {

```

```

00021     // model->loadModel _modelFilename
00022     // set values for the _selectedControls
00023     // model->startSimulation
00024     // get the value of the _selectedResponses from the model and store results on _responseValues
00025     // clear selected controls and responses
00026     // close the model
00027     return false;
00028 }
00029
00030 void SimulationScenario::setScenarioName(std::string _name) {
00031     this->_scenarioName = _name;
00032 }
00033
00034 std::string SimulationScenario::getScenarioName() const {
00035     return _scenarioName;
00036 }
00037
00038
00039 void SimulationScenario::setModelFilename(std::string _modelFilename) {
00040     this->_modelFilename = _modelFilename;
00041 }
00042
00043 std::string SimulationScenario::getModelFilename() const {
00044     return _modelFilename;
00045 }
00046
00047
00048 std::list<SimulationResponse*>* SimulationScenario::getSelectedResponses
00049     () const {
00050         return _selectedResponses;
00051     }
00052
00053 std::list<SimulationControl*>* SimulationScenario::getSelectedControls
00054     () const {
00055         return _selectedControls;
00056     }
00057
00058 void SimulationScenario::setScenarioDescription(std::string
00059     _scenarioDescription) {
00060     this->_scenarioDescription = _scenarioDescription;
00061 }
00062 }
```

7.429 SimulationScenario.h File Reference

```
#include <string>
#include <list>
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Classes

- class [SimulationScenario](#)

7.430 SimulationScenario.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    SimulationScenario.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 10 de Outubro de 2018, 18:21
00012 */
```

```

00013
00014 #ifndef SIMULATIONSCENARIO_H
00015 #define SIMULATIONSCENARIO_H
00016
00017 #include <string>
00018 #include <list>
00019 #include "SimulationResponse.h"
00020 #include "SimulationControl.h"
00021
00022 class SimulationScenario {
00023 public:
00024     SimulationScenario();
00025     virtual ~SimulationScenario() = default;
00026 public: // results
00027     bool startSimulation(std::string* errorMessage);
00028     std::list<std::pair<std::string, double>>* getResponseValues() const;
00029     std::list<std::pair<std::string, double>>* getControlValues() const;
00030     double getResponseValue(std::string responseName);
00031     public: //
00032     double getControlValue(std::string controlName);
00033     void setControlValue(std::string controlName, double value);
00034 public: // gets
00035     void setModelFilename(std::string _modelFilename);
00036     std::string getModelFilename() const;
00037     void setScenarioName(std::string _name);
00038     std::string getScenarioName() const;
00039     std::list<SimulationResponse*>* getSelectedResponses() const; // access to the list
00040     to insert or remove responses
00041     std::list<SimulationControl*>* getSelectedControls() const; // access to the list to
00042     insert or remove controls
00043     void setScenarioDescription(std::string _scenarioDescription);
00044     std::string getScenarioDescription() const;
00045 private:
00046     std::string _scenarioName;
00047     std::string _scenarioDescription;
00048     std::string _modelFilename;
00049     std::list<SimulationControl*>* _selectedControls = new std::list<SimulationControl*>();
00050     std::list<SimulationResponse*>* _selectedResponses = new std::list<SimulationResponse*>();
00051     std::list<std::pair<std::string, double>>* _controlValues;
00052     std::list<std::pair<std::string, double>>* _responseValues;
00053
00054 };
00055
00056 #endif /* SIMULATIONSCENARIO_H */
00057

```

7.431 Simulator.cpp File Reference

```

#include "Simulator.h"
#include "LicenceManager.h"
#include "ToolManager.h"

```

7.432 Simulator.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Genesys.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:48
00012 */
00013
00014 #include "Simulator.h"
00015 #include "LicenceManager.h"
00016 #include "ToolManager.h"
00017
00018 Simulator::Simulator() {
00019     // This is the ONLY method in the entire software where std::cout is allowed.
00020     std::cout << "STARTING " << _name << ", version " << _version << std::endl;
00021     _licenceManager = new LicenceManager(this);
00022     _pluginManager = new PluginManager(this);

```

```

00023     _modelManager = new ModelManager(this);
00024     _toolManager = new ToolManager(this);
00025     _traceManager = new TraceManager(this);
00026     std::cout << '|' << '\t' << _licenceManager->showLicence() << std::endl;
00027     //std::cout << '|' << '\t' << _licenceManager->showActivationCode() << std::endl;
00028     //std::cout << '|' << '\t' << _licenceManager->showLimits() << std::endl;
00029 }
00030
00031 PluginManager* Simulator::plugins() const {
00032     return _pluginManager;
00033 }
00034
00035 ModelManager* Simulator::models() const {
00036     return _modelManager;
00037 }
00038
00039 ToolManager* Simulator::tools() const {
00040     return _toolManager;
00041 }
00042
00043 TraceManager* Simulator::tracer() const {
00044     return _traceManager;
00045 }
00046
00047 ParserManager* Simulator::parser() const {
00048     return _parserManager;
00049 }
00050
00051 std::string Simulator::version() const {
00052     return _version;
00053 }
00054
00055 std::string Simulator::name() const {
00056     return _name;
00057 }
00058
00059 LicenceManager* Simulator::licenceManager() const {
00060     return _licenceManager;
00061 }
00062

```

7.433 Simulator.h File Reference

```

#include <string>
#include <iostream>
#include "Model.h"
#include "Plugin.h"
#include "List.h"
#include "LicenceManager.h"
#include "PluginManager.h"
#include "ModelManager.h"
#include "ToolManager.h"
#include "ParserManager.h"

```

Classes

- class [Simulator](#)

7.434 Simulator.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Genesys.h

```

```

00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 12:48
00012 */
00013
00014 #ifndef GENESYS_H
00015 #define GENESYS_H
00016
00017 #include <string>
00018 #include <iostream>
00019 #include "Model.h"
00020 #include "Plugin.h"
00021 #include "List.h"
00022 #include "LicenceManager.h"
00023 #include "PluginManager.h"
00024 #include "ModelManager.h"
00025 #include "ToolManager.h"
00026 #include "ParserManager.h"
00027
00028 class Simulator {
00029     typedef void (*eventHandler)();
00030 public:
00031     Simulator();
00032     virtual ~Simulator() = default;
00033 public: // only get
00034     std::string version() const;
00035     std::string name() const;
00036     LicenceManager* licenceManager() const;
00037     PluginManager* plugins() const;
00038     ModelManager* models() const;
00039     ToolManager* tools() const;
00040     TraceManager* tracer() const;
00041     ParserManager* parser() const;
00042 private:
00043     private: // attributes 1:1 objects
00044     LicenceManager* _licenceManager;
00045     PluginManager* _pluginManager;
00046     ModelManager* _modelManager;
00047     ToolManager* _toolManager;
00048     TraceManager* _traceManager;
00049     ParserManager* _parserManager;
00050 private: // attributes 1:1 native
00051     const std::string _name = "ReGenESyS - REborn GENeric and Expansible SYstem Simulator";
00052     const std::string _version = "19.10 (Halloween19)";
00053 };
00054
00055 #endif /* GENESYS_H */
00056

```

7.435 SinkModelComponent.cpp File Reference

```
#include "SinkModelComponent.h"
```

7.436 SinkModelComponent.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   SinkModelComponent.cpp
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 14 de Agosto de 2018, 14:29
00012 */
00013
00014 #include "SinkModelComponent.h"
00015
00016 SinkModelComponent::SinkModelComponent(
    Model* model, std::string componentTypename, std::string name) :
    ModelComponent(model, componentTypename, name) {
00017 }
00018
00019

```

```

00020 void SinkModelComponent::setCollectStatistics(bool
    _collectStatistics) {
00021     this->_collectStatistics = _collectStatistics;
00022 }
00023
00024 bool SinkModelComponent::isCollectStatistics() const {
00025     return _collectStatistics;
00026 }
00027
00028 bool SinkModelComponent::_loadInstance(std::map<std::string, std::string>*
    fields) {
00029     bool res = ModelComponent::_loadInstance(fields);
00030     if (res) {
00031         this->_collectStatistics = std::stoi((*fields->find("collectStatistics")).second) != 0;
00032     }
00033     return res;
00034 }
00035
00036 void SinkModelComponent::_initBetweenReplications() {
00037 }
00038
00039 std::map<std::string, std::string>* SinkModelComponent::_saveInstance() {
00040     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance()
00041     ();
00042     fields->emplace("collectStatistics", std::to_string(this->_collectStatistics));
00043     return fields;
00044 }
00045 bool SinkModelComponent::_check(std::string* errorMessage) {
00046     return true;
00047 }

```

7.437 SinkModelComponent.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [SinkModelComponent](#)

7.438 SinkModelComponent.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   SinkModelComponent.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 14 de Agosto de 2018, 14:29
00012 */
00013
00014 #ifndef SINKMODELCOMPONENT_H
00015 #define SINKMODELCOMPONENT_H
00016
00017 #include "ModelComponent.h"
00018
00023 class SinkModelComponent : public ModelComponent {
00024 public:
00025     SinkModelComponent(Model* model, std::string componentTypename, std::string
        name="");
00026     virtual ~SinkModelComponent() = default;
00027 public:
00028     void setCollectStatistics(bool _collectStatistics);
00029     bool isCollectStatistics() const;
00030 public:
00031 protected:
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual void _initBetweenReplications();
00034     virtual std::map<std::string, std::string>* _saveInstance();

```

```

00035     virtual bool _check(std::string* errorMessage);
00036 private:
00037     bool _collectStatistics = true;
00038 };
00039
00040 #endif /* SINKMODELCOMPONENT_H */
00041

```

7.439 SourceModelComponent.cpp File Reference

```

#include "SourceModelComponent.h"
#include "Model.h"
#include "Attribute.h"

```

7.440 SourceModelComponent.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: SourceModelComponent.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 19:50
00012 */
00013
00014 #include "SourceModelComponent.h"
00015 #include "Model.h"
00016 #include "Attribute.h"
00017
00018 SourceModelComponent::SourceModelComponent(
    Model* model, std::string componentTypename, std::string name) :
    ModelComponent(model, componentTypename, name) {
00019 }
00020
00021
00022 std::string SourceModelComponent::show() {
00023     std::string text = ModelComponent::show() +
00024         ",entityType=\"" + _entityType->name() + "\"\" +
00025         ",firstCreation=" + std::to_string(_firstCreation);
00026     return text;
00027 }
00028
00029 bool SourceModelComponent::_loadInstance(std::map<std::string,
    std::string>* fields) {
00030     bool res = ModelComponent::_loadInstance(fields);
00031     if (res) {
00032         this->_entitiesPerCreation = std::stoi((*fields->find("entitiesPerCreation")).second);
00033         this->_firstCreation = std::stod((*fields->find("firstCreation")).second);
00034         this->_timeBetweenCreationsExpression = (*fields->find("timeBetweenCreations")).second;
00035         this->_timeBetweenCreationsTimeUnit = static_cast<
            Util::TimeUnit> (std::stoi((*fields->find("timeBetweenCreationsTimeUnit")).second));
00036         this->_maxCreationsExpression = (*fields->find("maxCreations")).second;
00037         std::string entityTypename = (*fields->find("entityTypename")).second;
00038         this->_entityType = dynamic_cast<EntityType*> (
            _parentModel->elements()->element(Util::TypeOf<EntityType>(), entityTypename));
00039     }
00040     return res;
00041 }
00042
00043 void SourceModelComponent::_initBetweenReplications() {
00044     this->_entitiesCreatedSoFar = 0;
00045 }
00046
00047 std::map<std::string, std::string>* SourceModelComponent::_saveInstance(
    ) {
00048     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
        ();
00049     fields->emplace("entitiesPerCreation", std::to_string(this->
        _entitiesPerCreation));

```

```
00050     fields->emplace("firstCreation", std::to_string(this->_firstCreation));
00051     fields->emplace("timeBetweenCreations", this->
00052         _timeBetweenCreationsExpression);
00053     fields->emplace("timeBetweenCreationsTimeUnit", std::to_string(static_cast<int> (this->
00054         _timeBetweenCreationsTimeUnit)));
00055     fields->emplace("maxCreations", this->_maxCreationsExpression);
00056     fields->emplace("entityTypename", (this->_entityType->name())); // save the name
00057     //fields->emplace("collectStatistics", std::to_string(this->_collectStatistics));
00058     return fields;
00059 }
00060
00059 bool SourceModelComponent::_check(std::string* errorMessage) {
00060     /* include attributes needed */
00061     ElementManager* elements = _parentModel->elements();
00062     std::vector<std::string> neededNames = {"Entity.ArrivalTime", "Entity.NVATime",
00063     "Entity.WaitTime", "Entity.TransferTime", "Entity.OtherTime"};
00064     std::string neededName;
00065     for (unsigned int i = 0; i < neededNames.size(); i++) {
00066         neededName = neededNames[i];
00067         if (elements->element(Util::TypeOf<Attribute>(), neededName) == nullptr) {
00068             Attribute* attr1 = new Attribute(_parentModel, neededName);
00069             elements->insert(attr1);
00070         }
00071         bool resultAll = true;
00072         resultAll &= _parentModel->elements()->check(Util::TypeOf<EntityType>(),
00073             _entityType, "entitytype", errorMessage);
00074         resultAll &= _parentModel->checkExpression(this->
00075             _timeBetweenCreationsExpression, "time between creations", errorMessage);
00076     }
00077     return resultAll;
00078 }
00077 void SourceModelComponent::setFirstCreation(double
00078     _firstCreation) {
00079     this->_firstCreation = _firstCreation;
00080 }
00081 double SourceModelComponent::firstCreation() const {
00082     return _firstCreation;
00083 }
00084
00085 //void SourceModelComponent::setCollectStatistics(bool _collectStatistics) {
00086 //    this->_collectStatistics = _collectStatistics;
00087 //}
00088 //
00089 //bool SourceModelComponent::isCollectStatistics() const {
00090 //    return _collectStatistics;
00091 //}
00092
00093 void SourceModelComponent::setEntityType(
00094     EntityType* entityType) {
00095     _entityType = entityType;
00096 }
00097 EntityType* SourceModelComponent::entityType() const {
00098     return _entityType;
00099 }
00100
00101 void SourceModelComponent::setTimeUnit(
00102     Util::TimeUnit _timeUnit) {
00103     this->_timeBetweenCreationsTimeUnit = _timeUnit;
00104 }
00105 Util::TimeUnit SourceModelComponent::timeUnit() const {
00106     return this->_timeBetweenCreationsTimeUnit;
00107 }
00108
00109 void SourceModelComponent:: setTimeBetweenCreationsExpression
00110     (std::string _timeBetweenCreations) {
00111     this->_timeBetweenCreationsExpression = _timeBetweenCreations;
00112 }
00113 std::string SourceModelComponent::timeBetweenCreationsExpression
00114     () const {
00115     return _timeBetweenCreationsExpression;
00116 }
00117 void SourceModelComponent::setMaxCreations(std::string
00118     _maxCreationsExpression) {
00119     this->_maxCreationsExpression = _maxCreationsExpression;
00120 }
00121 std::string SourceModelComponent::maxCreations() const {
00122     return _maxCreationsExpression;
00123 }
00124
00125 unsigned int SourceModelComponent::entitiesCreated() const {
```

```

00126     return _entitiesCreatedSoFar;
00127 }
00128
00129 void SourceModelComponent::setEntitiesCreated(unsigned int
00130     _entitiesCreated) {
00131     this->_entitiesCreatedSoFar = _entitiesCreated;
00132 }
00133 void SourceModelComponent::setEntitiesPerCreation(unsigned int
00134     _entitiesPerCreation) {
00135     this->_entitiesPerCreation = _entitiesPerCreation;
00136     //this->_entitiesCreatedSoFar = _entitiesPerCreation; // that's because "entitiesPerCreation" entities
00137     // are included in the future events list BEFORE replication starts (to initialize events list)
00138 }
00139
00138 unsigned int SourceModelComponent::entitiesPerCreation() const {
00140     return _entitiesPerCreation;
00141 }
```

7.441 SourceModelComponent.h File Reference

```
#include "ModelComponent.h"
#include <string>
#include <limits>
```

Classes

- class [SourceModelComponent](#)

7.442 SourceModelComponent.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  SourceModelComponent.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 21 de Junho de 2018, 19:50
00012 */
00013
00014 #ifndef SOURCEMODELCOMPONENT_H
00015 #define SOURCEMODELCOMPONENT_H
00016
00017 #include "ModelComponent.h"
00018 #include <string>
00019 #include <limits>
00020 //##include <numeric_limits>
00021
00022 class SourceModelComponent : public ModelComponent {
00023 public:
00024     SourceModelComponent(Model* model, std::string componentTypename, std::string
00025     name="");
00026     virtual ~SourceModelComponent() = default;
00027 public: // get & set
00028     void setFirstCreation(double _firstCreation);
00029     double firstCreation() const;
00030     void setCollectStatistics(bool _collectStatistics);
00031     bool isCollectStatistics() const;
00032     void setEntityType(EntityType* _entityType);
00033     EntityType* entityType() const;
00034     void setTimeUnit(Util::TimeUnit _timeUnit);
00035     Util::TimeUnit timeUnit() const;
00036     void setTimeBetweenCreationsExpression(std::string
00037     _timeBetweenCreations);
00038     std::string timeBetweenCreationsExpression() const;
00039     void setMaxCreations(std::string _maxCreationsExpression);
00040     std::string maxCreations() const;
```

```

00044     unsigned int entitiesCreated() const;
00045     void setEntitiesCreated(unsigned int _entitiesCreated);
00046     void setEntitiesPerCreation(unsigned int
00047         _entitiesPerCreation);
00047     unsigned int entitiesPerCreation() const;
00048 public:
00049     virtual std::string show();
00050 protected:
00051     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052     virtual void _initBetweenReplications();
00053     virtual std::map<std::string, std::string>* _saveInstance();
00054     virtual bool _check(std::string* errorMessage);
00055 protected: // get & set
00056     EntityType _entityType;
00057     double _firstCreation = 0.0;
00058     unsigned int _entitiesPerCreation = 1;
00059     std::string _maxCreationsExpression = std::to_string(std::numeric_limits<unsigned int>::max()); //
00060     std::string _timeBetweenCreationsExpression = "EXPO(1)";
00061     Util::TimeUnit _timeBetweenCreationsTimeUnit =
00062         Util::TimeUnit::second;
00063     //bool _collectStatistics = true;
00063     unsigned int _entitiesCreatedSoFar = 0;
00064 };
00065
00066 #endif /* SOURCEMODELCOMPONENT_H */
00067

```

7.443 Start.cpp File Reference

```
#include "Start.h"
#include "Model.h"
```

7.444 Start.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Start.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #include "Start.h"
00015
00016 #include "Model.h"
00017
00018 Start::Start(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Start>(), name) {
00020
00021
00022     std::string Start::show() {
00023         return ModelComponent::show() + "";
00024     }
00025
00026     ModelComponent* Start::LoadInstance(Model* model,
00027         std::map<std::string, std::string>* fields) {
00028         Start* newComponent = new Start(model);
00029         try {
00030             newComponent->_loadInstance(fields);
00031         } catch (const std::exception& e) {
00032
00033         }
00034     }
00035
00036     void Start::_execute(Entity* entity) {
00037         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038             entity forward");
00039         this->_parentModel->sendEntityToComponent(entity, this->

```

```

nextComponents() -> frontConnection(), 0.0);
00039 }
00040
00041 bool Start::__loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelComponent::__loadInstance(fields);
00043     if (res) {
00044         //...
00045     }
00046     return res;
00047 }
00048
00049 void Start::__initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* Start::__saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::__saveInstance
        ();
00054     //...
00055     return fields;
00056 }
00057
00058 bool Start::__check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Start::GetPluginInformation() {
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Start>()
        , &Start::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069
00070

```

7.445 Start.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class Start

7.446 Start.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Start.h
00009  * Author: rlcancian
00010 *
00011  * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #ifndef START_H
00015 #define START_H
00016
00017 #include "ModelComponent.h"
00018
00039 class Start : public ModelComponent {
00040 public: // constructors
00041     Start(Model* model, std::string name = "");
00042     virtual ~Start() = default;
00043 public: // virtual
00044     virtual std::string show();
00045 public: // static
00046     static PluginInformation* GetPluginInformation();
00047     static ModelComponent* LoadInstance(Model* model, std::map<std::string,

```

```

        std::string>* fields);
00048 protected: // virtual
00049     virtual void _execute(Entity* entity);
00050     virtual void _initBetweenReplications();
00051     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00052     virtual std::map<std::string, std::string>* _saveInstance();
00053     virtual bool _check(std::string* errorMessage);
00054 private: // methods
00055 private: // attributes 1:1
00056 private: // attributes 1:n
00057 };
00058
00059
00060 #endif /* START_H */
00061

```

7.447 Station.cpp File Reference

```

#include "Station.h"
#include "Entity.h"
#include "Model.h"
#include "Attribute.h"

```

7.448 Station.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Station.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #include "Station.h"
00015 #include "Entity.h"
00016 #include "Model.h"
00017 #include "Attribute.h"
00018
00019 Station::Station(Model* model, std::string name) :
    ModelElement(model, Util::TypeOf<Station>(), name) {
00020     _initCStats();
00021 }
00022
00023 void Station::_initCStats() {
00024     _cstatNumberInStation = new StatisticsCollector(
00025         _parentModel, _name+"."+Number_In_Station, this);
00026     _cstatTimeInStation = new StatisticsCollector(
00027         _parentModel, _name+"."+Time_In_Station, this);
00028     _childrenElements->insert({"NumberInStation", _cstatNumberInStation});
00029     _childrenElements->insert({"TimeInStation", _cstatTimeInStation});
00030 }
00031 Station::~Station() {
00032     // _parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatNumberInStation);
00033     // _parentModel->elements()->remove(Util::TypeOf<StatisticsCollector>(), _cstatTimeInStation);
00034 }
00035 std::string Station::show() {
00036     std::string msg = ModelElement::show() + ",enterIntoStationComponent=";
00037     if (_enterIntoStationComponent == nullptr)
00038         msg += "NULL";
00039     else
00040         msg += _enterIntoStationComponent->name();
00041     return msg;
00042 }
00043
00044 void Station::initBetweenReplications() {
00045     //this->_list->clear();
00046 }
00047

```

```

00048 void Station::enter(Entity* entity) {
00049     std::string attributeName = "Entity.ArrivalAt" + this->name();
00050     trimwithin(attributeName);
00051     entity->setAttributeValue(attributeName, _parentModel->
00052         simulation()->simulatedTime());
00052     entity->attributeValue("Entity.Station", _id);
00053     _numberInStation++;
00054     this->_cstatNumberInStation->getStatistics()->getCollector()->
00055         addValue(_numberInStation);
00056 }
00056
00057 void Station::leave(Entity* entity) {
00058     std::string attributeName = "Entity.ArrivalAt" + this->name();
00059     trimwithin(attributeName);
00060     double arrivalTime = entity->attributeValue(attributeName);
00061     double timeInStation = _parentModel->simulation()->
00062         simulatedTime() - arrivalTime;
00063     _cstatTimeInStation->getStatistics()->getCollector()->
00064         addValue(timeInStation);
00065     entity->entityType()->statisticsCollector("Time in Stations")->
00066         getStatistics()->getCollector()->addValue(timeInStation);
00067     entity->attributeValue("Entity.Station", 0.0);
00068     _numberInStation--;
00069     _cstatNumberInStation->getStatistics()->getCollector()->
00070         addValue(_numberInStation);
00071 }
00072
00073 ModelComponent* Station::getEnterIntoStationComponent()
00074     const {
00075         return _enterIntoStationComponent;
00076     }
00077 PluginInformation* Station::GetPluginInformation() {
00078     PluginInformation* info = new PluginInformation(Util::TypeOf<Station>
00079         (), &Station::LoadInstance);
00080     return info;
00081 }
00082 ModelElement* Station::LoadInstance(Model* model,
00083     std::map<std::string, std::string>* fields) {
00084     Station* newElement = new Station(model);
00085     try {
00086         newElement->_loadInstance(fields);
00087     } catch (const std::exception& e) {
00088     }
00089     return newElement;
00090 }
00091
00092 bool Station::_loadInstance(std::map<std::string, std::string>* fields) {
00093     bool res = ModelElement::_loadInstance(fields);
00094     if (res) {
00095         try {
00096         } catch (...) {
00097         }
00098     }
00099     return res;
00100 }
00101
00102 std::map<std::string, std::string>* Station::_saveInstance() {
00103     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00104     //Util::TypeOf<Station>();
00105     return fields;
00106 }
00107 bool Station::_check(std::string* errorMessage) {
00108     /* include attributes needed */
00109     std::vector<std::string> neededNames = {"Entity.Station"};
00110     neededNames.insert(neededNames.begin(), "Entity.ArrivalAt" + this->name());
00111     std::string neededName;
00112     for (unsigned int i = 0; i < neededNames.size(); i++) {
00113         neededName = neededNames[i];
00114         if (_parentModel->elements()->element(Util::TypeOf<Attribute>(), neededName)
00115             == nullptr) {
00116             Attribute* attr1 = new Attribute(_parentModel, neededName);
00117             //_parentModel->insert(attr1);
00118         }
00119         // include StatisticsCollector needed in EntityType
00120         std::list<ModelElement*>* enttypes = _parentModel->elements()->
00121             elementList(Util::TypeOf<EntityType>())->list();
00122         for (std::list<ModelElement*>::iterator it = enttypes->begin(); it != enttypes->end(); it++) {

```

```

00122     static_cast<EntityType*> ((*it))->statisticsCollector("Time in Stations"); // force create
00123     this CStat before simulation starts
00124     //
00125     return true;
00126 }
00127
00128 void Station::_createInternalElements() {
00129     //_initCStats();
00130 }
00131

```

7.449 Station.h File Reference

```

#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"
#include "Entity.h"

```

Classes

- class [Station](#)

7.450 Station.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:   Station.h
00009  * Author: rlcancian
00010  *
00011  * Created on 03 de Junho de 2019, 15:12
00012 */
00013
00014 #ifndef STATION_H
00015 #define STATION_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020 #include "Entity.h"
00021
00022 /*
00023  Station module
00024 DESCRIPTION
00025 The Station module defines a station (or a set of stations) corresponding to a physical
00026 or logical location where processing occurs. If the Station module defines a station
00027 set, it is effectively defining multiple processing locations.
00028 The station (or each station within the defined set) has a matching Activity Area that
00029 is used to report all times and costs accrued by the entities in this station. This
00030 Activity Area's name is the same as the station. If a parent Activity Area is defined,
00031 then it also accrues any times and costs by the entities in this station.
00032 TYPICAL USES
00033 Defining a lathe area
00034 Defining a set of toll booths
00035 Defining a food preparation area
00036 PROMPTS
00037 Prompt Description
00038 Name Unique name of the module that will be displayed in the
00039 flowchart.
00040 Station Type Type of station being defined, either as an individual Station or a
00041 station Set.
00042 Station Name Name of the individual station.
00043 Set Name Name of the station set.
00044 Parent Activity Area Name of the Activity Area's parent.
00045 Associated Intersection Name of the intersection associated with this station in a guided
00046 transporter network.

```

```

00047 Report Statistics Specifies whether or not statistics will automatically be collected
00048 and stored in the report database for this station and its
00049 corresponding activity area.
00050 Save Attribute Attribute name used to store the index number into the station set
00051 of the member that is selected.
00052 Station Set Members Names of the stations that are members of this station set.
00053 Station Name A given station can only exist once within a model. Therefore, an
00054 individual station can only be the member of one station set, and
00055 that individual station may not be the name of a station in
00056 another module.
00057 Parent Activity Area Name of the Activity Area's parent for the station set member.
00058 Associated Intersection Name of the intersection associated with this station set in a
00059 guided transporter network.
00060 Report Statistics Specifies whether or not statistics will automatically be collected
00061 and stored in the report database for this station set member and
00062 its corresponding activity area.
00063 */
00064 class Station : public ModelElement {
00065 public:
00066     Station(Model* model, std::string name = "");
00067     virtual ~Station();
00068 public:
00069     virtual std::string show();
00070 public: // static
00071     static PluginInformation* GetPluginInformation();
00072     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00073                                         std::string>* fields);
00074 public:
00075     void initBetweenReplications();
00076     void enter(Entity* entity);
00077     void leave(Entity* entity);
00078     void setEnterIntoStationComponent(ModelComponent*
00079                                         _enterIntoStationComponent);
00080     ModelComponent* getEnterIntoStationComponent() const;
00081 protected:
00082     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00083     virtual std::map<std::string, std::string>* _saveInstance();
00084     virtual bool _check(std::string* errorMessage);
00085     virtual void _createInternalElements();
00086 private:
00087     void _initCStats();
00088     unsigned int _numberInStation = 0;
00089     ModelComponent* _enterIntoStationComponent;
00090     StatisticsCollector* _cstatNumberInStation;
00091     StatisticsCollector* _cstatTimeInStation;
00092
00093 };
00094 #endif /* STATION_H */
00096

```

7.451 Statistics_if.h File Reference

```
#include <string>
#include "Collector_if.h"
```

Classes

- class [Statistics_if](#)

7.452 Statistics_if.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Statistics_if.h

```

```

00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 14 de Agosto de 2018, 13:47
00012 */
00013
00014 #ifndef STATISTICS_IF_H
00015 #define STATISTICS_IF_H
00016
00017 #include <string>
00018 #include "Collector_if.h"
00019
00023 class Statistics_if {
00024 public:
00025     virtual Collector_if* getCollector() = 0;
00026     virtual void setCollector(Collector_if* collector) = 0;
00027 public:
00028     virtual unsigned int numElements() = 0;
00029     virtual double min() = 0;
00030     virtual double max() = 0;
00031     virtual double average() = 0;
00032     virtual double variance() = 0;
00033     virtual double stdDeviation() = 0;
00034     virtual double variationCoef() = 0;
00035     virtual double halfWidthConfidenceInterval() = 0;
00036     virtual unsigned int newSampleSize(double halfWidth) = 0;
00037     virtual double getConfidenceLevel() = 0;
00038     virtual void setConfidenceLevel(double confidencelevel) = 0;
00039 };
00040
00041 #endif /* STATISTICS_IF_H */
00042

```

7.453 StatisticsCollector.cpp File Reference

```
#include "StatisticsCollector.h"
#include "Traits.h"
```

Typedefs

- **typedef Traits< ModelComponent >::StatisticsCollector_StatisticsImplementation StatisticsClass**

7.453.1 Typedef Documentation

7.453.1.1 StatisticsClass

```
typedef Traits<ModelComponent>::StatisticsCollector_StatisticsImplementation StatisticsClass
```

Definition at line 17 of file [StatisticsCollector.cpp](#).

7.454 StatisticsCollector.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: StatisticsCollector.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Agosto de 2018, 17:24
00012 */
00013
00014 #include "StatisticsCollector.h"
00015 #include "Traits.h"
00016
00017 typedef Traits<ModelComponent>::StatisticsCollector_StatisticsImplementation
00018 StatisticsClass;
00019 StatisticsCollector::StatisticsCollector(
00020     Model* model, std::string name, ModelElement* parent, bool insertIntoModel) :
00021     ModelElement(model, Util::TypeOf<StatisticsCollector>(), name,
00022     insertIntoModel) {
00023     _parent = parent;
00024     _initStaticsAndCollector();
00025     _addSimulationResponse();
00026 }
00027 void StatisticsCollector::_addSimulationResponse() {
00028     GetterMember getterMember = DefineGetterMember<StatisticsClass>(static_cast<
00029         StatisticsClass*>(this->_statistics), &StatisticsClass::average);
00030     std::string parentName = "";
00031     if (_parent != nullptr)
00032         parentName = _parent->name();
00033     SimulationResponse* resp = new SimulationResponse(
00034         Util::TypeOf<StatisticsClass>(), parentName + ":" + _name + ".average", getterMember);
00035     _parentModel->responses()->insert(resp);
00036 }
00037 void StatisticsCollector::_initStaticsAndCollector() {
00038     Collector_if* collector = new
00039         Traits<ModelComponent>::StatisticsCollector_CollectorImplementation
00040         ();
00041     _statistics = new StatisticsClass(collector);//
00042     Traits<ModelComponent>::StatisticsCollector_StatisticsImplementation(collector);
00043 }
00044 std::string StatisticsCollector::show() {
00045     std::string parentStr = "";
00046     if (_parent != nullptr) {
00047         try {
00048             parentStr = _parent->name();
00049         } catch (...) { // if parent changed or deleted, can cause seg fault
00050             parentStr = "<<INCONSISTENT>>"; /* \todo: +++ */
00051         }
00052         return ModelElement::show() +
00053             ",parent=\"" + parentStr + "\"\n";
00054     }
00055     ModelElement* StatisticsCollector::getParent() const {
00056         return _parent;
00057     }
00058     Statistics_if* StatisticsCollector::getStatistics() const {
00059         return _statistics;
00060     }
00061
00062     PluginInformation* StatisticsCollector::GetPluginInformation
00063     () {
00064         PluginInformation* info = new PluginInformation(
00065             Util::TypeOf<StatisticsCollector>(), &StatisticsCollector::LoadInstance);
00066         info->setGenerateReport(true);
00067         return info;
00068     }
00069     ModelElement* StatisticsCollector::LoadInstance(
00070         Model* model, std::map<std::string, std::string>* fields) {
00071         StatisticsCollector* newElement = new StatisticsCollector(model);
00072         try {
00073             newElement->_loadInstance(fields);
00074         }
00075     }

```

```

00073     } catch (const std::exception& e) {
00074     }
00075     return newElement;
00076 }
00077
00078 bool StatisticsCollector::_loadInstance(std::map<std::string,
00079                                         std::string>*& fields) {
00080     bool res = ModelElement::_loadInstance(fields);
00081     if (res) {
00082     }
00083     return res;
00084 }
00085
00086 std::map<std::string, std::string>* StatisticsCollector::_saveInstance()
00087 {
00088     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00089     //Util::TypeOf<StatisticsCollector>());
00090     std::string parentId = "", parentTypename = "";
00091     if (this->_parent != nullptr) {
00092         parentId = std::to_string(_parent->id());
00093         parentTypename = _parent->classname();
00094     }
00095     fields->emplace("parentTypename", parentTypename);
00096     fields->emplace("parentId", parentId);
00097     return fields;
00098 }
00099
00100 bool StatisticsCollector::_check(std::string* errorMessage) {
00101     return true;
00102 }
```

7.455 StatisticsCollector.h File Reference

```
#include "ModelElement.h"
#include "Statistics_if.h"
#include "ElementManager.h"
#include "Plugin.h"
```

Classes

- class [StatisticsCollector](#)

7.456 StatisticsCollector.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: StatisticsCollector.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 30 de Agosto de 2018, 17:24
00012 */
00013
00014 #ifndef STATISTICS_COLLECTOR_H
00015 #define STATISTICS_COLLECTOR_H
00016
00017 #include "ModelElement.h"
00018 #include "Statistics_if.h"
00019 #include "ElementManager.h"
00020 #include "Plugin.h"
00021
00022 class StatisticsCollector : public ModelElement { //, public Statistics_if {
00023 public:
00024     StatisticsCollector(Model* model, std::string name="",
00025                         ModelElement* parent=nullptr, bool insertIntoModel = true);
00026     virtual ~StatisticsCollector() = default;
```

```

00029 public:
00030     virtual std::string show();
00031 public:
00032     static PluginInformation* GetPluginInformation();
00033     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00034                                         std::string>* fields);
00035 public:
00036     ModelElement* getParent() const;
00037     Statistics_if* getStatistics() const;
00038 protected:
00039     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00040     virtual std::map<std::string, std::string>* _saveInstance();
00041     virtual bool _check(std::string* errorMessage);
00042 protected:
00043     void _addSimulationResponse();
00044 private:
00045     void _initStaticsAndCollector();
00046 private:
00047     ModelElement* _parent;
00048     Statistics_if* _statistics; //= new
        Traits<ModelComponent>::StatisticsCollector_StatisticsImplementation();
00049 };
00050 #endif /* STATISTICSCOLLECTOR_H */
00052

```

7.457 StatisticsDataFile_if.h File Reference

```
#include "Collector_if.h"
#include "Statistics_if.h"
```

Classes

- class [StatisticsDatafile_if](#)

7.458 StatisticsDataFile_if.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: StatisticsDataFile.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Novembro de 2018, 01:16
00012 */
00013
00014 #ifndef STATISTICSDATAFILE_IF_H
00015 #define STATISTICSDATAFILE_IF_H
00016
00017 #include "Collector_if.h"
00018 #include "Statistics_if.h"
00019
00020 class StatisticsDatafile_if : public Statistics_if {
00021 public:
00022     virtual double mode() = 0;
00023     virtual double mediane() = 0;
00024     virtual double quartil(unsigned short num) = 0;
00025     virtual double decil(unsigned short num) = 0;
00026     virtual double centil(unsigned short num) = 0;
00027     virtual void setHistogramNumClasses(unsigned short num) = 0;
00028     virtual unsigned short histogramNumClasses() = 0;
00029     virtual double histogramClassLowerLimit(unsigned short className) = 0;
00030     virtual unsigned int histogramClassFrequency(unsigned short className) = 0;
00031
00032 };
00033
00034 #endif /* STATISTICSDATAFILE_IF_H */
00035

```

7.459 StatisticsDataFileDefaultImpl.cpp File Reference

```
#include "StatisticsDataFileDefaultImpl.h"
#include "Traits.h"
```

7.460 StatisticsDataFileDefaultImpl.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: StatisticsDataFileDummyImpl.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 22 de Novembro de 2018, 01:24
00012 */
00013
00014 #include "StatisticsDataFileDefaultImpl.h"
00015
00016 #include "Traits.h"
00017
00018 StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl
00019 () {
00020     _collector = new Traits<Statistics_if>::CollectorImplementation
00021 }
00022
00023 unsigned int StatisticsDataFileDummyImpl::numElements() {
00024     return 0; // dummy
00025 }
00026
00027 double StatisticsDataFileDummyImpl::min() {
00028     return 0.0; // dummy
00029 }
00030
00031 double StatisticsDataFileDummyImpl::max() {
00032     return 0.0; // dummy
00033 }
00034
00035 double StatisticsDataFileDummyImpl::average() {
00036     return 0.0; // dummy
00037 }
00038
00039 double StatisticsDataFileDummyImpl::mode() {
00040     return 0.0; // dummy
00041 }
00042
00043 double StatisticsDataFileDummyImpl::mediane() {
00044     return 0.0; // dummy
00045 }
00046
00047 double StatisticsDataFileDummyImpl::variance() {
00048     return 0.0; // dummy
00049 }
00050
00051 double StatisticsDataFileDummyImpl::stddeviation() {
00052     return 0.0; // dummy
00053 }
00054
00055 double StatisticsDataFileDummyImpl::variationCoef() {
00056     return 0.0; // dummy
00057 }
00058
00059 double StatisticsDataFileDummyImpl::halfWidthConfidenceInterval
00060 (double confidencelevel) {
00061     return 0.0; // dummy
00062 }
00063 unsigned int StatisticsDataFileDummyImpl::newSampleSize(double
00064     confidencelevel, double halfWidth) {
00065     return 0; // dummy
00066 }
00067 double StatisticsDataFileDummyImpl::quartil(unsigned short num) {
00068     return 0.0; // dummy
```

```

00069 }
00070
00071 double StatisticsDataFileDummyImpl::decil(unsigned short num) {
00072     return 0.0; // dummy
00073 }
00074
00075 double StatisticsDataFileDummyImpl::centil(unsigned short num) {
00076     return 0.0; // dummy
00077 }
00078
00079 void StatisticsDataFileDummyImpl::setHistogramNumClasses
    (unsigned short num) {
00080 }
00081
00082 unsigned short StatisticsDataFileDummyImpl::histogramNumClasses
() {
00083     return 0; // dummy
00084 }
00085
00086 double StatisticsDataFileDummyImpl::histogramClassLowerLimit
    (unsigned short classNum) {
00087     return 0.0; // dummy
00088 }
00089
00090 unsigned int StatisticsDataFileDummyImpl::histogramClassFrequency
    (unsigned short classNum) {
00091     return 0; // dummy
00092 }
00093
00094 Collector_if* StatisticsDataFileDummyImpl::getCollector
() {
00095     return this->_collector;
00096 }
00097
00098 void StatisticsDataFileDummyImpl::setCollector(
    Collector_if* collector) {
00099
00100 }
```

7.461 StatisticsDataFileDefaultImpl.h File Reference

```
#include "StatisticsDataFile_if.h"
```

Classes

- class [StatisticsDataFileDummyImpl](#)

7.462 StatisticsDataFileDefaultImpl.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  StatisticsDataFileDefaultImpl.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 22 de Novembro de 2018, 01:24
00012 */
00013
00014 #ifndef STATISTICSDATAFILEDEFAULTIMPL_H
00015 #define STATISTICSDATAFILEDEFAULTIMPL_H
00016
00017 #include "StatisticsDataFile_if.h"
00018
00019 class StatisticsDataFileDummyImpl : public
    StatisticsDatafile_if {
00020 public:
00021     StatisticsDataFileDummyImpl();
00022     virtual ~StatisticsDataFileDummyImpl() = default;
00023 public:
```

```

00024     virtual Collector_if* getCollector();
00025     void setCollector(Collector_if* collector);
00026 public:
00027     virtual unsigned int numElements();
00028     virtual double min();
00029     virtual double max();
00030     virtual double average();
00031     virtual double variance();
00032     virtual double stdDeviation();
00033     virtual double variationCoef();
00034     virtual double halfWidthConfidenceInterval(double confidencelevel);
00035     virtual unsigned int newSampleSize(double confidencelevel, double halfWidth);
00036
00037     virtual double mode();
00038     virtual double mediane();
00039     virtual double quartil(unsigned short num);
00040     virtual double decil(unsigned short num);
00041     virtual double centil(unsigned short num);
00042     virtual void setHistogramNumClasses(unsigned short num);
00043     virtual unsigned short histogramNumClasses();
00044     virtual double histogramClassLowerLimit(unsigned short className);
00045     virtual unsigned int histogramClassFrequency(unsigned short className);
00046 private:
00047     Collector_if* _collector;
00048 };
00049
00050 #endif /* STATISTICSDEFAULTIMPL_H */
00051

```

7.463 StatisticsDefaultImpl1.cpp File Reference

```

#include <complex>
#include "StatisticsDefaultImpl1.h"
#include "Traits.h"
#include "Integrator_if.h"
#include "ProbDistrib.h"

```

7.464 StatisticsDefaultImpl1.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: StatisticsDefaultImpl1.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 1 de Agosto de 2018, 21:03
00012 */
00013
00014 #include <complex>
00015
00016 #include "StatisticsDefaultImpl1.h"
00017 #include "Traits.h"
00018 #include "Integrator_if.h"
00019 #include "ProbDistrib.h"
00020
00021 StatisticsDefaultImpl1::StatisticsDefaultImpl1() {
00022     _collector = new Traits<Statistics_if>::CollectorImplementation
00023     ();
00024     _collector->setAddValueHandler(SetCollectorAddValueHandler
00025         (&StatisticsDefaultImpl1::collectorAddHandler, this));
00026     _collector->setClearHandler(SetCollectorClearHandler(&
00027         StatisticsDefaultImpl1::collectorClearHandler, this));
00028     // _collector->setAddValueHandler(std::bind(&StatisticsDefaultImpl1::collectorAddHandler, this,
00029     std::placeholders::_1));
00030     this->initStatistics();
00031 }
00032
00033 StatisticsDefaultImpl1::StatisticsDefaultImpl1(
00034     Collector_if* collector) {
00035     _collector = collector;

```

```

00031     _collector->setAddValueHandler(SetCollectorAddValueHandler
00032     (&StatisticsDefaultImpl1::collectorAddHandler, this));
00033     _collector->setClearHandler(SetCollectorClearHandler(&
00034     StatisticsDefaultImpl1::collectorClearHandler, this));
00035     //_collector->setAddValueHandler(std::bind(&StatisticsDefaultImpl1::collectorAddHandler, this,
00036     std::placeholders::_1));
00037     this->initStatistics();
00038 }
00039
00040 void StatisticsDefaultImpl1::collectorAddHandler(double newValue) {
00041     _elems = _collector->numElements();
00042     if (newValue < _min) {
00043         _min = newValue;
00044     }
00045     if (newValue > _max) {
00046         _max = newValue;
00047     }
00048     _sum += newValue;
00049     _sumSquare += newValue*newValue;
00050     _average = _sum / _elems;
00051     _variance = _sumSquare / _elems - _average*_average;
00052     // _average = (_average * (elems - 1) + newValue) / elems; // this approach propagates the numeric
00053     // error
00054     //_variance = (_variance * (elems - 1) + pow(newValue - _average, 2)) / elems; // this approach
00055     // propagates the numeric error
00056     _stddeviation = sqrt(_variance);
00057     _variationCoef = (_stddeviation != 0 ? _stddeviation / _average : 0.0);
00058     _halfWidth = _criticalTn_1 * (_stddeviation / std::sqrt(_elems));
00059 }
00060
00061 void StatisticsDefaultImpl1::initStatistics() {
00062     _elems = 0;
00063     _min = 1e+99;
00064     _max = -1e+99;
00065     _sum = 0.0;
00066     _sumSquare = 0.0;
00067     _average = 0.0;
00068     _variance = 0.0;
00069     _stddeviation = 0.0;
00070     _variationCoef = 0.0;
00071     _halfWidth = 0.0;
00072 }
00073
00074 unsigned int StatisticsDefaultImpl1::numElements() {
00075     return this->getCollector()->numElements();
00076 }
00077
00078 double StatisticsDefaultImpl1::min() {
00079     if (_elems > 0)
00080         return _min;
00081     else
00082         return 0.0;
00083 }
00084
00085 double StatisticsDefaultImpl1::max() {
00086     if (_elems > 0)
00087         return _max;
00088     else
00089         return 0.0;
00090 }
00091
00092 double StatisticsDefaultImpl1::average() {
00093     return _average;
00094 }
00095
00096 double StatisticsDefaultImpl1::variance() {
00097     return _variance;
00098 }
00099
00100 double StatisticsDefaultImpl1::stddeviation() {
00101     return _stddeviation;
00102 }
00103
00104 double StatisticsDefaultImpl1::variationCoef() {
00105     return _variationCoef;
00106 }
00107
00108 double StatisticsDefaultImpl1::halfWidthConfidenceInterval
00109     () {
00110     return _halfWidth;
00111 }
00112

```

```

00112 void StatisticsDefaultImpl1::setConfidenceLevel(double
00113     confidencelevel) {
00114     _confidenceLevel = confidencelevel;
00115     //Integrator_if* integrator = new Traits<Integrator_if>::Implementation();
00116     _criticalTn_1 = 1.96; //integrator->integrate()
00117 }
00118
00119 double StatisticsDefaultImpl1::getConfidenceLevel() {
00120     return _confidenceLevel;
00121 }
00122
00123 unsigned int StatisticsDefaultImpl1::newSampleSize(double halfWidth) {
00124     return 0;
00125 }
00126
00127 Collector_if* StatisticsDefaultImpl1::getCollector() {
00128     return this->_collector;
00129 }
00130
00131 void StatisticsDefaultImpl1::setCollector(
00132     Collector_if* collector) {
00132     this->_collector = collector;
00133 }

```

7.465 StatisticsDefaultImpl1.h File Reference

```
#include "Statistics_if.h"
#include "Collector_if.h"
```

Classes

- class [StatisticsDefaultImpl1](#)

7.466 StatisticsDefaultImpl1.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  StatisticsDefaultImpl1.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 1 de Agosto de 2018, 21:03
00012 */
00013
00014 #ifndef STATISTICSDEFAULTIMPL1_H
00015 #define STATISTICSDEFAULTIMPL1_H
00016
00017 #include "Statistics_if.h"
00018 #include "Collector_if.h"
00019
00020 class StatisticsDefaultImpl1 : public Statistics_if {
00021 public:
00022     StatisticsDefaultImpl1();
00023     StatisticsDefaultImpl1(Collector_if* collector);
00024     virtual ~StatisticsDefaultImpl1() = default;
00025 public:
00026     virtual Collector_if* getCollector();
00027     virtual void setCollector(Collector_if* collector);
00028 public:
00029     virtual unsigned int numElements();
00030     virtual double min();
00031     virtual double max();
00032     virtual double average();
00033     virtual double variance();
00034     virtual double stdDeviation();
00035     virtual double variationCoef();
00036     virtual double halfWidthConfidenceInterval();

```

```
00037     virtual unsigned int newSampleSize(double halfWidth);
00038     virtual double getConfidenceLevel();
00039     virtual void setConfidenceLevel(double confidencelevel);
00040 private:
00041     void collectorAddHandler(double newValue);
00042     void collectorClearHandler();
00043     void initStatistics();
00044 private:
00045     Collector_if* _collector;
00046     unsigned long _elems;
00047     double _sum;
00048     double _sumSquare;
00049     double _min;
00050     double _max;
00051     double _average;
00052     double _variance;
00053     double _stddeviation;
00054     double _variationCoef;
00055     double _confidenceLevel = 0.95;
00056     double _criticalTn_1 = 1.96;
00057     double _halfWidth;
00058 };
00059
00060 #endif /* STATISTICSDEFAULTIMPL1_H */
00061
```

7.467 Stop.cpp File Reference

```
#include "Stop.h"  
#include "Model.h"
```

7.468 Stop.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Stop.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #include "Stop.h"
00015
00016 #include "Model.h"
00017
00018 Stop::Stop(Model* model, std::string name) : ModelComponent(model,
00019     Util::TypeOf<Stop>(), name) {
00020
00021
00022 std::string Stop::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Stop::LoadInstance(Model* model, std::map<std::string,
00027     std::string>* fields) {
00028     Stop* newComponent = new Stop(model);
00029     try {
00030         newComponent->_loadInstance(fields);
00031     } catch (const std::exception& e) {
00032     }
00033     return newComponent;
00034 }
00035
00036 void Stop::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00038     entity forward");
00039     this->_parentModel->sendEntityToComponent(entity, this->
00040         nextComponents()->frontConnection(), 0.0);
00041 }
```

```

00040
00041 bool Stop::_loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelComponent::_loadInstance(fields);
00043     if (res) {
00044         //...
00045     }
00046     return res;
00047 }
00048
00049 void Stop::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* Stop::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00054     ();
00055     //...
00056     return fields;
00057 }
00058
00059 bool Stop::_check(std::string* errorMessage) {
00060     bool resultAll = true;
00061     //...
00062     return resultAll;
00063 }
00064 PluginInformation* Stop::GetPluginInformation(){
00065     PluginInformation* info = new PluginInformation(Util::TypeOf<Stop>(),
00066     &Stop::LoadInstance);
00067     // ...
00068     return info;
00069 }
00070

```

7.469 Stop.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class Stop

7.470 Stop.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Stop.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:15
00012 */
00013
00014 #ifndef STOP_H
00015 #define STOP_H
00016
00017 #include "ModelComponent.h"
00018
00036 class Stop : public ModelComponent {
00037 public: // constructors
00038     Stop(Model* model, std::string name="");
00039     virtual ~Stop() = default;
00040 public: // virtual
00041     virtual std::string show();
00042 public: // static
00043     static PluginInformation* GetPluginInformation();
00044     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00045     std::string>* fields);
00045 protected: // virtual

```

```

00046     virtual void _execute(Entity* entity);
00047     virtual void _initBetweenReplications();
00048     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00049     virtual std::map<std::string, std::string>* _saveInstance();
00050     virtual bool _check(std::string* errorMessage);
00051 private: // methods
00052 private: // attributes 1:1
00053 private: // attributes 1:n
00054 };
00055
00056
00057 #endif /* STOP_H */
00058

```

7.471 Storage.cpp File Reference

```
#include "Storage.h"
```

7.472 Storage.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Storage.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 20 de Storageembro de 2019, 20:06
00012 */
00013
00014 #include "Storage.h"
00015
00016 Storage::Storage(Model* model, std::string name) :
00017     ModelElement(model, Util::TypeOf<Storage>(), name) {
00018 }
00019
00020 std::string Storage::show() {
00021     return ModelElement::show() +
00022         "";
00023 }
00024
00025 PluginInformation* Storage::GetPluginInformation() {
00026     PluginInformation* info = new PluginInformation(Util::TypeOf<Storage>
00027         (), &Storage::LoadInstance);
00028     return info;
00029 }
00030 ModelElement* Storage::LoadInstance(Model* model,
00031     std::map<std::string, std::string>* fields) {
00032     Storage* newElement = new Storage(model);
00033     try {
00034         newElement->_loadInstance(fields);
00035     } catch (const std::exception& e) {
00036     }
00037     return newElement;
00038 }
00039
00040 bool Storage::_loadInstance(std::map<std::string, std::string>* fields) {
00041     bool res = ModelElement::_loadInstance(fields);
00042     if (res) {
00043         try {
00044             //this->_attributeName = (*fields->find("attributeName")).second;
00045             //this->_orderRule = static_cast<OrderRule> (std::stoi((*fields->find("orderRule")).second));
00046         } catch (...) {
00047         }
00048     }
00049     return res;
00050 }
00051
00052 std::map<std::string, std::string>* Storage::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();

```

```

00054     //Util::TypeOf<Storage>());
00055     //fields->emplace("orderRule", std::to_string(static_cast<int> (this->_orderRule)));
00056     //fields->emplace("attributeName", this->_attributeName);
00057     return fields;
00058 }
00059 bool Storage::_check(std::string* errorMessage) {
00060     bool resultAll = true;
00061     // resultAll |= ...
00062     return resultAll;
00063 }
00064
00065 ParserChangesInformation*
00066     Storage::_getParserChangesInformation() {
00067         ParserChangesInformation* changes = new
00068             ParserChangesInformation();
00069         //changes->getProductionToAdd()->insert(...);
00070         //changes->getTokensToAdd()->insert(...);
00071         return changes;
00072 }
00073

```

7.473 Storage.h File Reference

```

#include "ModelElement.h"
#include "ElementManager.h"
#include "ParserChangesInformation.h"
#include "PluginInformation.h"

```

Classes

- class [Storage](#)

7.474 Storage.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Storage.h
00009  * Author: rlcancian
00010  *
00011  * Created on 20 de Storageembro de 2019, 20:06
00012 */
00013
00014 #ifndef STORAGE_H
00015 #define STORAGE_H
00016
00017
00018 #include "ModelElement.h"
00019 #include "ElementManager.h"
00020 #include "ParserChangesInformation.h"
00021 #include "PluginInformation.h"
00022
00023 /*
00024 Storage module
00025 DESCRIPTION
00026 The Storage module defines the name of a storage. Storages are automatically created
00027 by any module that references the storage so that this module is seldom needed. The
00028 only time this module is needed is when a storage is defined as a member of a storage
00029 set or specified using an attribute or expression.
00030 TYPICAL USES
00031 Defining an animate storage for a set of storages
00032 PROMPTS
00033 Prompt Description
00034 Name The name of the storage set being defined. This name must be
00035 unique.
00036 */

```

```
00037 class Storage: public ModelElement {
00038 public:
00039     Storage(Model* model, std::string name = "");
00040     virtual ~Storage() = default;
00041 public: // static
00042     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00043                                         std::string*>* fields);
00044     static PluginInformation* GetPluginInformation();
00045 public:
00046     virtual std::string show();
00047 protected: // must be overridden by derived classes
00048     virtual bool _loadInstance(std::map<std::string, std::string*>* fields),
00049     virtual std::map<std::string, std::string*>* _saveInstance();
00050 protected: // could be overridden by derived classes
00051     virtual bool _check(std::string* errorMessage);
00052     virtual ParserChangesInformation*
00053     _getParserChangesInformation();
00054 };
00054 #endif /* STORAGE_H */
00055
```

7.475 Store.cpp File Reference

```
#include "Store.h"  
#include "Model.h"
```

7.476 Store.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Store.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:07
00012 */
00013
00014 #include "Store.h"
00015 #include "Model.h"
00016
00017 Store::Store(Model* model, std::string name) : ModelComponent(model,
00018     Util::TypeOf<Store>(), name) {
00019 }
00020
00021 std::string Store::show() {
00022     return ModelComponent::show() + "";
00023 }
00024
00025 ModelComponent* Store::LoadInstance(Model* model,
00026     std::map<std::string, std::string>* fields) {
00027     Store* newComponent = new Store(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031     }
00032     return newComponent;
00033 }
00034
00035 void Store::_execute(Entity* entity) {
00036     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00037     entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
00039         nextComponents()->frontConnection(), 0.0);
00040 }
00041
00042 bool Store::_loadInstance(std::map<std::string, std::string>* fields) {
00043     bool res = ModelComponent::_loadInstance(fields);
00044     if (res) {
00045         //...
00046     }
00047 }
```

```

00044      }
00045      return res;
00046  }
00047
00048 void Store::_initBetweenReplications() {
00049 }
00050
00051 std::map<std::string, std::string>* Store::_saveInstance() {
00052     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053     ();
00054     //...
00055     return fields;
00056 }
00057 bool Store::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Store::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Store>()
00065     , &Store::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069

```

7.477 Store.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Store](#)

7.478 Store.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Store.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:07
00012 */
00013
00014 #ifndef STORE_H
00015 #define STORE_H
00016
00017 #include "ModelComponent.h"
00018
00050 class Store : public ModelComponent {
00051 public: // constructors
00052     Store(Model* model, std::string name(""));
00053     virtual ~Store() = default;
00054 public: // virtual
00055     virtual std::string show();
00056 public: // static
00057     static PluginInformation* GetPluginInformation();
00058     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00059     std::string>* fields);
00059 protected: // virtual
00060     virtual void _execute(Entity* entity);
00061     virtual void _initBetweenReplications();
00062     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00063     virtual std::map<std::string, std::string>* _saveInstance();
00064     virtual bool _check(std::string* errorMessage);

```

```

00065 private: // methods
00066 private: // attributes 1:1
00067 private: // attributes 1:n
00068 };
00069
00070
00071 #endif /* STORE_H */
00072

```

7.479 Submodel.cpp File Reference

```
#include "Submodel.h"
#include "Model.h"
```

7.480 Submodel.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Submodel.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:04
00012 */
00013
00014 #include "Submodel.h"
00015
00016 #include "Model.h"
00017
00018 Submodel::Submodel(Model* model, std::string name) :
    ModelComponent(model, Util::TypeOf<Submodel>(), name) {
00019 }
00020
00021
00022 std::string Submodel::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Submodel::LoadInstance(Model* model,
    std::map<std::string, std::string>* fields) {
00027     Submodel* newComponent = new Submodel(model);
00028     try {
00029         newComponent->_loadInstance(fields);
00030     } catch (const std::exception& e) {
00031     }
00032 }
00033     return newComponent;
00034 }
00035
00036 void Submodel::_execute(Entity* entity) {
00037     _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
entity forward");
00038     this->_parentModel->sendEntityToComponent(entity, this->
nextComponents()->frontConnection(), 0.0);
00039 }
00040
00041 bool Submodel::_loadInstance(std::map<std::string, std::string>* fields) {
00042     bool res = ModelComponent::_loadInstance(fields);
00043     if (res) {
00044         //...
00045     }
00046     return res;
00047 }
00048
00049 void Submodel::_initBetweenReplications() {
00050 }
00051
00052 std::map<std::string, std::string>* Submodel::_saveInstance() {
00053     std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
        ();
00054     //...
00055     return fields;

```

```

00056 }
00057
00058 bool Submodel::_check(std::string* errorMessage) {
00059     bool resultAll = true;
00060     //...
00061     return resultAll;
00062 }
00063
00064 PluginInformation* Submodel::GetPluginInformation() {
00065     PluginInformation* info = new PluginInformation(
00066         Util::TypeOf<Submodel>(), &Submodel::LoadInstance);
00067     // ...
00068     return info;
00069 }
00070

```

7.481 Submodel.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Submodel](#)

7.482 Submodel.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: Submodel.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:04
00012 */
00013
00014 #ifndef SUBMODEL_H
00015 #define SUBMODEL_H
00016
00017 #include "ModelComponent.h"
00018
00019 class Submodel : public ModelComponent {
00020 public: // constructors
00021     Submodel(Model* model, std::string name = "");
00022     virtual ~Submodel() = default;
00023 public: // virtual
00024     virtual std::string show();
00025 public: // static
00026     static PluginInformation* GetPluginInformation();
00027     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00028                                         std::string>* fields);
00029 protected: // virtual
00030     virtual void _execute(Entity* entity);
00031     virtual void _initBetweenReplications();
00032     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00033     virtual std::map<std::string, std::string>* _saveInstance();
00034     virtual bool _check(std::string* errorMessage);
00035 private: // methods
00036 private: // attributes 1:1
00037 private: // attributes 1:n
00038 };
00039
00040 */
00041
00042
00043 #endif /* SUBMODEL_H */
00044

```

7.483 TestEnterLeaveRoute.cpp File Reference

```
#include "TestEnterLeaveRoute.h"
#include "ComponentManager.h"
#include "Simulator.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Enter.h"
#include "Route.h"
#include "Leave.h"
#include "EntityType.h"
#include "Station.h"
```

7.484 TestEnterLeaveRoute.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestEnterLeaveRoute.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 18:04
00012 */
00013
00014 #include "TestEnterLeaveRoute.h"
00015 #include "ComponentManager.h"
00016
00017
00018 // you have to included need libs
00019
00020 // GEnSys Simulator
00021 #include "Simulator.h"
00022
00023 // Model Components
00024 #include "Create.h"
00025 #include "Delay.h"
00026 #include "Dispose.h"
00027 #include "Enter.h"
00028 #include "Route.h"
00029 #include "Leave.h"
00030
00031 // Model elements
00032 #include "EntityType.h"
00033 #include "Station.h"
00034
00035 TestEnterLeaveRoute::TestEnterLeaveRoute() {
00036 }
00037
00038 int TestEnterLeaveRoute::main(int argc, char** argv) {
00039     Simulator* simulator = new Simulator();
00040     // creates an empty model
00041     Model* model = new Model(simulator);
00042     // Handle traces and simulation events to output them
00043     TraceManager* tm = model->tracer();
00044     this->setDefaultTraceHandlers(tm);
00045     // set the trace level of simulation to "blockArrival" level, which is an intermediate level of tracing
00046     tm->setTraceLevel(Util::TraceLevel::componentArrival);
00047     // insert "fake plugins" since plugins based on dynamic loaded library are not implemented yet
00048     this->insertFakePluginsByHand(simulator);
00049     // get easy access to classes used to insert components and elements into a model
00050     ComponentManager* components = model->components();
00051     ElementManager* elements = model->elements();
00052     //
00053     // build the simulation model
00054     //
00055     // set general info about the model
00056     ModelInfo* infos = model->infos();
00057     infos->setReplicationLength(30);
00058     infos->setNumberOfReplications(3);
```

```

00059 // create a (Source)ModelElement of type EntityType, used by a ModelComponent that follows
00060 EntityType* entityType1 = new EntityType(model, "AnyEntityType");
00061 elements->insert(entityType1);
00062 // create a ModelComponent of type Create, used to insert entities into the model
00063 Create* create1 = new Create(model);
00064 create1->setEntityType(entityType1);
00065 create1->setTimeBetweenCreationsExpression("5.0");
00066 create1->setEntitiesPerCreation(1);
00067 components->insert(create1);
00068 // create stations to enter and route to
00069 Station* station1 = new Station(model, "Station 1");
00070 Station* station2 = new Station(model, "Station 2");
00071 Station* station3 = new Station(model, "Station 3");
00072 elements->insert(station1);
00073 elements->insert(station2);
00074 elements->insert(station3);
00075 // create components to Enter into Stations
00076 Enter* enter1 = new Enter(model);
00077 enter1->setStation(station1);
00078 components->insert(enter1);
00079 Enter* enter2 = new Enter(model);
00080 enter2->setStation(station2);
00081 components->insert(enter2);
00082 Enter* enter3 = new Enter(model);
00083 enter3->setStation(station3);
00084 components->insert(enter3);
00085 // create components to Leave stations
00086 Leave* leave1 = new Leave(model);
00087 leave1->setStation(station1);
00088 components->insert(leave1);
00089 Leave* leave2 = new Leave(model);
00090 leave2->setStation(station2);
00091 components->insert(leave2);
00092 Leave* leave3 = new Leave(model);
00093 leave3->setStation(station3);
00094 components->insert(leave3);
00095 // create route components
00096 Route* route0 = new Route(model);
00097 route0->setStation(station1);
00098 route0->setRouteTimeExpression("0.5");
00099 components->insert(route0);
00100 Route* route1 = new Route(model);
00101 route1->setStation(station2);
00102 route1->setRouteTimeExpression("0.5");
00103 components->insert(route1);
00104 Route* route2 = new Route(model);
00105 route2->setStation(station3);
00106 route2->setRouteTimeExpression("0.5");
00107 components->insert(route2);
00108 // create delay components
00109 Delay* delay1 = new Delay(model);
00110 delay1->setDelayExpression("1.0");
00111 components->insert(delay1);
00112 Delay* delay2 = new Delay(model);
00113 delay2->setDelayExpression("1.0");
00114 components->insert(delay2);
00115 Delay* delay3 = new Delay(model);
00116 delay3->setDelayExpression("1.0");
00117 components->insert(delay3);
00118 // create a (Sink)ModelComponent of type Dispose, used to remove entities from the model
00119 Dispose* dispose1 = new Dispose(model);
00120 components->insert(dispose1);
00121 // connect model components to create a "workflow"
00122 create1->nextComponents()->insert(route0);
00123 //
00124 enter1->nextComponents()->insert(delay1);
00125 delay1->nextComponents()->insert(leave1);
00126 leave1->nextComponents()->insert(route1);
00127 //
00128 enter2->nextComponents()->insert(delay2);
00129 delay2->nextComponents()->insert(leave2);
00130 leave2->nextComponents()->insert(route2);
00131 //
00132 enter3->nextComponents()->insert(delay3);
00133 delay3->nextComponents()->insert(leave3);
00134 leave3->nextComponents()->insert(dispose1);
00135 // insert the model into the simulator
00136 simulator->models()->insert(model);
00137 // check the model
00138 model->check();
00139 // save the model into a text file
00140 model->save("./temp/testEnterLeaveRoute.txt");
00141 // show the model
00142 model->show();
00143 // execute the simulation
00144 model->simulation()->start();
00145 return 0;

```

```
00146 }
00147
```

7.485 TestEnterLeaveRoute.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [TestEnterLeaveRoute](#)

7.486 TestEnterLeaveRoute.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: TestEnterLeaveRoute.h
00009 * Author: rlcancian
00010 *
00011 * Created on 9 de Setembro de 2019, 18:04
00012 */
00013
00014 #ifndef TESTENTERLEAVEROUTE_H
00015 #define TESTENTERLEAVEROUTE_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestEnterLeaveRoute: public
00020     BaseConsoleGenesysApplication {
00021     public:
00022         TestEnterLeaveRoute();
00023     public:
00024         int main(int argc, char** argv);
00025     };
00026
00027 #endif /* TESTENTERLEAVEROUTE_H */
```

7.487 TestFunctions.cpp File Reference

```
#include "TestFunctions.h"
#include "ProbDistrib.h"
#include <iostream>
#include <vector>
#include <boost/numeric/odeint.hpp>
#include <boost/math/distributions.hpp>
```

Typedefs

- [typedef std::vector< double > state_type](#)

Functions

- void `harmonic_oscillator` (const `state_type` &`x`, `state_type` &`dxdt`, const double `t`)

Variables

- const double `gam` = 0.15

7.487.1 Typedef Documentation

7.487.1.1 `state_type`

```
typedef std::vector< double > state_type
```

Definition at line 23 of file [TestFunctions.cpp](#).

7.487.2 Function Documentation

7.487.2.1 `harmonic_oscillator()`

```
void harmonic_oscillator (
    const state_type & x,
    state_type & dxdt,
    const double t )
```

Definition at line 26 of file [TestFunctions.cpp](#).

7.487.3 Variable Documentation

7.487.3.1 `gam`

```
const double gam = 0.15
```

Definition at line 24 of file [TestFunctions.cpp](#).

7.488 TestFunctions.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 26 de Setembro de 2019, 13:00
00012 */
00013
00014 #include "TestFunctions.h"
00015 #include "ProbDistrib.h"
00016
00017 #include <iostream>
00018 #include <vector>
00019
00020 #include <boost/numeric/odeint.hpp>
00021 #include <boost/math/distributions.hpp>
00022
00023 typedef std::vector< double > state_type;
00024 const double gam = 0.15;
00025
00026 void harmonic_oscillator(const state_type &x,
00027     state_type &dxdt, const double t) {
00028     dxdt[0] = x[1];
00029     dxdt[1] = x[0] + exp(t);
00030 }
00031 TestODE::TestODE() {
00032 }
00033
00034 int TestODE::main(int argc, char** argv) {
00035     using namespace std;
00036     using namespace boost::numeric::odeint;
00037     using namespace boost::math;
00038
00039 //beta_distribution<>* betadist = new beta_distribution<>(2.0, 2.0);
00040 //betadist->alpha();
00041     for (double x = -2.0; x <= 2.0; x += 0.1) {
00042         std::cout << x << ":" << ProbDistrib::gamma(x, 2.0, 2.0) << std::endl;
00043     }
00044 //normdist->mean();
00045 //beta<double,double>* b = new beta<double,double>(2.0, 2.0);
00046 //b->
00047
00048     state_type x(2);
00049
00050 /*
00051 x[0] = 1.0; // start at x=1.0, p=0.0
00052 x[1] = 0.0;
00053 runge_kutta4< state_type > stepper;
00054 integrate_const(stepper, harmonic_oscillator, x, 0.0, 0.2, 0.01);
00055 std::cout << x[0] << " ; " << x[1] << std::endl;
00056 */
00057
00058 /*
00059 x[0] = 1.0; // start at x=1.0, p=0.0
00060 x[1] = 0.0;
00061 const double dt = 0.01;
00062 double t;
00063 for (t = 0.0; t < 0.2; t += dt) {
00064     std::cout << t << ":" << x[0] << " ; " << x[1] << std::endl;
00065     stepper.do_step(harmonic_oscillator, x, t, dt);
00066 }
00067 std::cout << t << ":" << x[0] << " ; " << x[1] << std::endl;
00068 */
00069
00070 /*
00071 x[0] = 1.0; // start at x=1.0, p=0.0
00072 x[1] = 0.0;
00073 typedef runge_kutta_cash_karp54< state_type > error_stepper_type;
00074 typedef controlled_runge_kutta< error_stepper_type > controlled_stepper_type;
00075 controlled_stepper_type controlled_stepper;
00076 //integrate_adaptive(controlled_stepper, harmonic_oscillator, x, 0.0, 0.2, 0.01);
00077 integrate_adaptive(make_controlled< error_stepper_type >(1.0e-20, 1.0e-16), harmonic_oscillator, x,
00078 0.0, 0.2, 0.01);
00079     std::cout << x[0] << " ; " << x[1] << std::endl;
00080 */
00081     return 0;
00082 }
```

7.489 TestFunctions.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [TestODE](#)

7.490 TestFunctions.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    TestODE.h
00009  * Author:  rlcancian
00010 *
00011  * Created on 26 de Setembro de 2019, 13:00
00012 */
00013
00014 #ifndef TESTFUNCTIONS_H
00015 #define TESTFUNCTIONS_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestODE : public BaseConsoleGenesysApplication {
00020 public:
00021     TestODE();
00022 public:
00023     virtual int main(int argc, char** argv) ;
00024 };
00025
00026 #endif /* TESTFUNCTIONS_H */
00027
```

7.491 TestInputAnalyserTools.cpp File Reference

```
#include "TestInputAnalyserTools.h"
#include "Simulator.h"
#include "Sampler_if.h"
#include "ProbDistrib.h"
#include "Traits.h"
```

Functions

- void [testStudentSoftwareDevelopments \(\)](#)

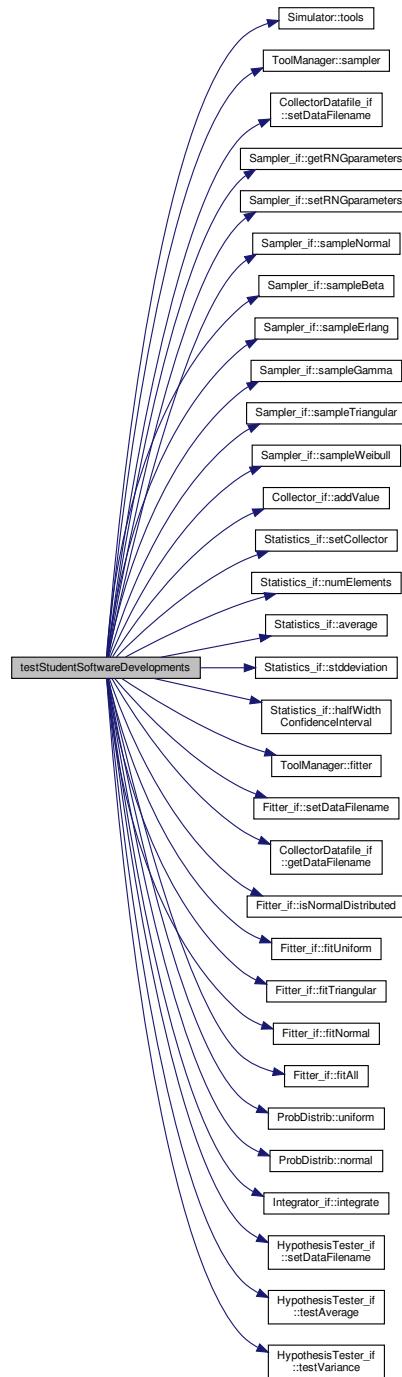
7.491.1 Function Documentation

7.491.1.1 testStudentSoftwareDevelopments()

```
void testStudentSoftwareDevelopments ( )
```

Definition at line 13 of file [TestInputAnalyserTools.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



7.492 TestInputAnalyserTools.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 #include "TestInputAnalyserTools.h"
00008 #include "Simulator.h"
00009 #include "Sampler_if.h"
00010 #include "Probbistribut.h"
00011 #include "Traits.h"
00012
00013 void testStudentSoftwareDevelopments() {
00014     Simulator* simulator = new Simulator();
00015     Sampler_if* mmc = simulator->tools()->sampler(); // Sampler is the new MMC
00016     CollectorDatafile_if* collector = (new
00017         Traits<Collector_if>::Implementation());
00018     collector->setDataFilename("./datafile.txt");
00019     // just to show how to change MMC parameters
00020     Traits<Sampler_if>::Parameters parameters =
00021         Traits<Sampler_if>::Parameters*) mmc->
00022             getRNGparameters();
00023     parameters->module = 2147483648;
00024     parameters->multiplier = 1103515245;
00025     parameters->seed = 1234;
00026     mmc->setRNGparameters(parameters);
00027
00028     // generate a datafile with a thousand values that should be normal distributed == NORM(5000,350)
00029     double value;
00030     for (unsigned int i = 0; i < 1e3; i++) {
00031         value = mmc->sampleNormal(5000, 350);
00032         value = mmc->sampleBeta(2, 4, 1000, 2000);
00033         value = mmc->sampleErlang(1000, 10);
00034         value = mmc->sampleGamma(1000, 2);
00035         value = mmc->sampleTriangular(1000, 1900, 2000);
00036         value = mmc->sampleWeibull(1000, 1);
00037         collector->addValue(value);
00038     }
00039
00040     // generate statistics about that datafile
00041     Statistics_if* statistics = new
00042         Traits<Statistics_if>::Implementation();
00043     statistics->setCollector(collector); //setDataFilename(collector->getDataFilename());
00044     double statVal;
00045     statVal = statistics->numElements();
00046     statVal = statistics->average();
00047     statVal = statistics->stddeviation();
00048     statVal = statistics->halfWidthConfidenceInterval();
00049     std::cout << statVal << std::endl;
00050
00051     /*
00052         statVal = statistics->quartil(2);
00053         statVal = statistics->mediane();
00054         unsigned int numclasses = statistics->histogramNumClasses();
00055         for (unsigned int i = 0; i < numclasses; i++) {
00056             statVal = statistics->histogramClassFrequency(i);
00057             statVal = statistics->histogramClassLowerLimit(i);
00058         }
00059     */
00060     // fit datafile to different probability distributions
00061     Fitter_if* fitter = simulator->tools()->fitter();
00062     fitter->setDataFilename(collector->getDataFilename());
00063     double sqrerror, p1, p2, p3; //, p4;
00064     std::string distribName;
00065     fitter->isNormalDistributed(0.95);

```

```

00061     fitter->fitUniform(&sqrerror, &p1, &p2);
00062     fitter->fitTriangular(&sqrerror, &p1, &p2, &p3);
00063     fitter->fitNormal(&sqrerror, &p1, &p2);
00064     fitter->fitAll(&sqrerror, &distribName);
00065
00066     // get some values from the PDF (Probability Density Functions)
00067     value = ProbDistrib::uniform(0.25, 0.0, 1.0);
00068     value = ProbDistrib::uniform(0.5, 0.0, 1.0); // should return the same value than
00069     above
00070     value = ProbDistrib::uniform(0.75, 0.0, 1.0); // should return the same value than
00071     above
00072     value = ProbDistrib::normal(10.0, 50.0, 20.0);
00073     value = ProbDistrib::normal(30.0, 50.0, 20.0); // should return a greater value than
00074     above
00075     value = ProbDistrib::normal(50.0, 50.0, 20.0); // should return a greater value than
00076     above
00077
00078     // calculate some integrals just for fun
00079     Integrator_if* integrator = new
00080     Traits<Integrator_if>::Implementation();
00081     value = integrator->integrate(0.0, 0.4, &ProbDistrib::uniform, 0.0, 0.1);
00082     // should return 0.4
00083     value = integrator->integrate(0.0, 100, &ProbDistrib::normal, 100, 10); //
00084     should return 0.5
00085
00086     // Test some hypothesis about the datafile
00087     HypothesisTester_if* tester = new
00088     Traits<HypothesisTester_if>::Implementation();
00089     tester->setDataFilename(collector->getDataFilename());
00090     bool res;
00091     res = tester->testAverage(0.95, 5500,
00092     HypothesisTester_if::GREATER_THAN);
00093     res = tester->testAverage(0.95, 4500,
00094     HypothesisTester_if::LESS_THAN);
00095     res = tester->testAverage(0.95, 5100,
00096     HypothesisTester_if::DIFFERENT);
00097     res = tester->testVariance(0.95, 350 * 350,
00098     HypothesisTester_if::LESS_THAN);
00099     std::cout << res << std::endl;
00100 }
00101
00102 TestInputAnalyserTools::TestInputAnalyserTools() {
00103 }
00104
00105 int TestInputAnalyserTools::main(int argc, char** argv) {
00106     testStudentSoftwareDevelopments();
00107     return 0;
00108 }
```

7.493 TestInputAnalyserTools.h File Reference

```
#include "GenesysApplication_if.h"
```

Classes

- class **TestInputAnalyserTools**

7.494 TestInputAnalyserTools.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: testInputAnalyserTools.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 2 de Outubro de 2018, 19:31
00012 */

```

```

00013
00014 #ifndef TESTINPUTANALYSERTOOLS_H
00015 #define TESTINPUTANALYSERTOOLS_H
00016
00017 using namespace std;
00018
00019 #include "GenesysApplication_if.h"
00020
00021 class TestInputAnalyserTools : public
00022     GenesysApplication_if {
00023 public:
00024     TestInputAnalyserTools();
00025     int main(int argc, char** argv);
00026 };
00027
00028 #endif /* TESTINPUTANALYSERTOOLS_H */
00029

```

7.495 TestLSODE.cpp File Reference

```

#include "TestLSODE.h"
#include "Simulator.h"
#include "Model.h"
#include "Create.h"
#include "Assign.h"
#include "Formula.h"
#include "Delay.h"
#include "LSODE.h"
#include "Dispose.h"
#include "Variable.h"

```

7.496 TestLSODE.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestLSODE.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 23 de Outubro de 2019, 13:07
00012 */
00013
00014 #include "TestLSODE.h"
00015 //
00016 #include "Simulator.h"
00017 #include "Model.h"
00018 #include "Create.h"
00019 #include "Assign.h"
00020 #include "Formula.h"
00021 #include "Delay.h"
00022 #include "LSODE.h"
00023 #include "Dispose.h"
00024 #include "Variable.h"
00025
00026 TestLSODE::TestLSODE() {
00027 }
00028
00029 int TestLSODE::main(int argc, char** argv) {
00030     Simulator* simulator = new Simulator();
00031     TraceManager* tm = simulator->tracer();
00032     tm->setTraceLevel(Util::TraceLevel::componentDetailed);
00033     this->setDefaultTraceHandlers(tm);
00034     this->insertFakePluginsByHand(simulator);
00035     Model* model = new Model(simulator);
00036     EntityType* entityType1 = new EntityType(model, "EntType_1");
00037     Create* create1 = new Create(model);
00038     create1->setEntityType(entityType1);

```

```

00039     create1->setTimeBetweenCreationsExpression("1.0");
00040     /*
00041     new Variable(model, "var1");
00042     Variable* var2 = new Variable(model, "var2");
00043     var2->setValue(1.0);
00044     Formula* formula1 = new Formula(model);
00045     formula1->getFormulaExpressions()->insert("var1 + cos(var2)");
00046     formula1->getFormulaExpressions()->insert("var1*var1 -6*var2");
00047     */
00048     Variable* varTime = new Variable(model, "x");
00049     Variable* vary = new Variable(model, "y");
00050     vary->setInitialValue("0", 1.0);
00051     vary->setInitialValue("1", 0.0);
00052     vary->dimensionSizes()->insert(2);
00053     Variable* varDy = new Variable(model, "dy");
00054     varDy->dimensionSizes()->insert(2);
00055     Formula* diffeq1 = new Formula(model);
00056     diffeq1->setExpression("0", "dy[0]=y[1]");
00057     diffeq1->setExpression("1", "dy[1]=y[0]+exp(x)");
00058     LSODE* odel = new LSODE(model);
00059     odel->setDiffEquations(diffeq1);
00060     odel->setVariables(vary);
00061     odel->setTimeVariable(varTime);
00062     odel->setStep(0.1);
00063     //Delay* delay1 = new Delay(model);
00064     Dispose* dispose1 = new Dispose(model);
00065     create1->nextComponents()->insert(odel);
00066     odel->nextComponents()->insert(dispose1);
00067     //delay1->getNextComponents()->insert(dispose1);
00068     simulator->models()->insert(model);
00069     model->save("./temp/testLSODE.txt");
00070     model->simulation()->start();
00071     return 0;
00072 }

```

7.497 TestLSODE.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
#include "List.h"
```

Classes

- class [TestLSODE](#)

7.498 TestLSODE.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: TestLSODE.h
00009  * Author: rlcancian
00010  *
00011  * Created on 23 de Outubro de 2019, 13:07
00012 */
00013
00014 #ifndef TESTLSODE_H
00015 #define TESTLSODE_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018 #include "List.h"
00019
00020 class TestLSODE : public BaseConsoleGenesysApplication {
00021 public:
00022     TestLSODE();
00023     virtual ~TestLSODE()=default;
00024 public:
00025     virtual int main(int argc, char** argv) ;
00026 private:
00027     List<std::string>* _differentialequation = new
00028     List<std::string>();
00029 };
00030 #endif /* TESTLSODE_H */
00031

```

7.499 TestMarkovChain.cpp File Reference

```
#include "TestMarkovChain.h"
#include "Simulator.h"
#include "Variable.h"
#include "EntityType.h"
#include "Create.h"
#include "MarkovChain.h"
#include "Write.h"
#include "Dispose.h"
```

7.500 TestMarkovChain.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: TestMarkovChain.cpp
00009  * Author: rlcancian
00010 *
00011  * Created on 24 de Outubro de 2019, 18:22
00012 */
00013
00014 #include "TestMarkovChain.h"
00015
00016 #include "Simulator.h"
00017 #include "Variable.h"
00018 #include "EntityType.h"
00019 #include "Create.h"
00020 #include "MarkovChain.h"
00021 #include "Write.h"
00022 #include "Dispose.h"
00023
00024 TestMarkovChain::TestMarkovChain() {
00025 }
00026
00027 int TestMarkovChain::main(int argc, char** argv) {
00028     Simulator* simulator = new Simulator();
00029     TraceManager* tm = simulator->tracer();
00030     //tm->setTraceLevel(Util::TraceLevel::mostDetailed);
00031     setDefaultTraceHandlers(tm);
00032     insertFakePluginsByHand(simulator);
00033     Model* model = new Model(simulator);
00034     model->infos()->setNumberOfReplications(5);
00035     model->infos()->setReplicationLength(100);
00036     Variable* varInitDist = new Variable(model, "InitialDistribution");
00037     varInitDist->setValue("0", 0.7);
00038     varInitDist->setValue("1", 0.1);
00039     varInitDist->setValue("2", 0.1);
00040     varInitDist->setValue("3", 0.1);
00041     varInitDist->dimensionSizes()->insert(4);
00042     Variable* varTransitProb = new Variable(model, "transitionProbMatrix'");
00043     varTransitProb->setValue("0,0", 0.1);
00044     varTransitProb->setValue("0,1", 0.7);
00045     varTransitProb->setValue("0,2", 0.1);
00046     varTransitProb->setValue("0,3", 0.1);
00047     varTransitProb->setValue("1,0", 0.1);
00048     varTransitProb->setValue("1,1", 0.1);
00049     varTransitProb->setValue("1,2", 0.7);
00050     varTransitProb->setValue("1,3", 0.1);
00051     varTransitProb->setValue("2,0", 0.1);
00052     varTransitProb->setValue("2,1", 0.1);
00053     varTransitProb->setValue("2,2", 0.1);
00054     varTransitProb->setValue("2,3", 0.7);
00055     varTransitProb->setValue("3,0", 0.7);
00056     varTransitProb->setValue("3,1", 0.1);
00057     varTransitProb->setValue("3,2", 0.1);
00058     varTransitProb->setValue("3,3", 0.1);
00059     varTransitProb->dimensionSizes()->insert(4);
00060     varTransitProb->dimensionSizes()->insert(4);
00061     Variable* varCurrStat = new Variable(model, "currentState");
00062     EntityType* enttypel = new EntityType(model);
00063     Create* createl = new Create(model);
```

```

00064     create1->setEntityType(enttype1);
00065     create1-> setTimeBetweenCreationsExpression("1");
00066     MarkovChain* markov1 = new MarkovChain(model);
00067     markov1->setInitialDistribution(varInitDist);
00068     markov1->setTransitionProbabilityMatrix(varTransitProb);
00069     markov1->setCurrentState(varCurrStat);
00070     Write* write1 = new Write(model);
00071     write1->setWriteToType(Write::WriteToType::FILE);
00072     write1->setFilename("./temp/markov1_realizations.txt");
00073     write1->writeElements()->insert(new WriteElement("tnow", true));
00074     write1->writeElements()->insert(new WriteElement("\t"));
00075     write1->writeElements()->insert(new WriteElement("currentState", true,
00076                                         true));
00076     Dispose* dispose1 = new Dispose(model);
00077     create1->nextComponents()->insert(markov1);
00078     markov1->nextComponents()->insert(write1);
00079     write1->nextComponents()->insert(dispose1);
00080     simulator->models()->insert(model);
00081     model->save("./temp/testmarkovchain.txt");
00082     model->simulation()->start();
00083     return 0;
00084 }

```

7.501 TestMarkovChain.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [TestMarkovChain](#)

7.502 TestMarkovChain.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: TestMarkovChain.h
00009  * Author: rlcancian
00010 *
00011  * Created on 24 de Outubro de 2019, 18:22
00012 */
00013
00014 #ifndef TESTMARKOVCHAIN_H
00015 #define TESTMARKOVCHAIN_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestMarkovChain : public BaseConsoleGenesysApplication {
00020 public:
00021     TestMarkovChain();
00022     virtual ~TestMarkovChain() = default;
00023 public:
00024     virtual int main(int argc, char** argv);
00025 private:
00026
00027 };
00028
00029 #endif /* TESTMARKOVCHAIN_H */
00030

```

7.503 TestMatricesOfAttributesAndVariables.cpp File Reference

```
#include "TestMatricesOfAttributesAndVariables.h"
#include "Simulator.h"
#include "SourceModelComponent.h"
#include "Assign.h"
#include "Attribute.h"
#include "Variable.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Separate.h"
#include "Formula.h"
#include "Write.h"
```

7.504 TestMatricesOfAttributesAndVariables.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: TestMatricesOfAttributesAndVariables.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 1 de Novembro de 2019, 18:10
00012 */
00013
00014 #include "TestMatricesOfAttributesAndVariables.h"
00015 #include "Simulator.h"
00016 #include "SourceModelComponent.h"
00017 #include "Assign.h"
00018 #include "Attribute.h"
00019 #include "Variable.h"
00020 #include "Create.h"
00021 #include "Delay.h"
00022 #include "Dispose.h"
00023 #include "Separate.h"
00024 #include "Formula.h"
00025 #include "Write.h"
00026
00027 TestMatricesOfAttributesAndVariables::TestMatricesOfAttributesAndVariables
00028 {}
00029
00030 TestMatricesOfAttributesAndVariables::TestMatricesOfAttributesAndVariables
00031     (const TestMatricesOfAttributesAndVariables& orig) {
00032 }
00033 TestMatricesOfAttributesAndVariables::~TestMatricesOfAttributesAndVariables
00034 {}
00035
00036 int TestMatricesOfAttributesAndVariables::main(int argc, char** argv) {
00037     Simulator* sim = new Simulator();
00038     setDefaultTraceHandlers(sim->tracer());
00039     sim->tracer()->setTraceLevel(
00040         Util::TraceLevel::modelSimulationInternal);
00041     insertFakePluginsByHand(sim);
00042     Model* m = new Model(sim);
00043     sim->models()->insert(m);
00044     m->infos()->setProjectTitle("Stochastic Simulation of Chemical Reactions");
00045     m->infos()->setReplicationLength(500);
00046     Create* crl = new Create(m);
00047     Write* wl = new Write(m);
00048     Assign* as1 = new Assign(m, "Define próxima reação a ocorrer");
00049     Delay* del = new Delay(m, "Aguarda tempo em que a reação ocorre");
00050     Dispose* dil = new Dispose(m);
00051     crl->nextComponents()->insert(wl);
00052     wl->nextComponents()->insert(as1);
00053     as1->nextComponents()->insert(del);
```

```

00053     de1->nextComponents()->insert(w1);
00054     de1->nextComponents()->insert(d1l); // trick to be connected
00055     cr1->setEntityType(new EntityType(m));
00056     cr1->setMaxCreations("1");
00057     Variable* s = new Variable(m, "s");
00058     Variable* k = new Variable(m, "k");
00059     Variable* N = new Variable(m, "N");
00060     new Variable(m, "temp");
00061     Formula* prop = new Formula(m, "prop");
00062     s->setInitialValue("1,1", -1);
00063     s->setInitialValue("1,2", -1);
00064     s->setInitialValue("1,3", 1);
00065     s->setInitialValue("2,1", 1);
00066     s->setInitialValue("2,2", 1);
00067     s->setInitialValue("2,3", -1);
00068     k->setInitialValue("1", 0.1);
00069     k->setInitialValue("2", 0.2);
00070     N->setInitialValue("1", 100);
00071     N->setInitialValue("2", 100);
00072     N->setInitialValue("3", 0);
00073     prop->setExpression("1", "k[1]*N[1]*N[2]");
00074     prop->setExpression("2", "k[2]*N[3]");
00075     w1->setWriteToType(WriteToType::FILE);
00076     w1->setFilename("./temp/molecules.txt");
00077     w1->writeElements()->insert(new WriteElement("tnow", true));
00078     w1->writeElements()->insert(new WriteElement(" "));
00079     w1->writeElements()->insert(new WriteElement("N[1]", true));
00080     w1->writeElements()->insert(new WriteElement(" "));
00081     w1->writeElements()->insert(new WriteElement("N[2]", true));
00082     w1->writeElements()->insert(new WriteElement(" "));
00083     w1->writeElements()->insert(new WriteElement("N[3]", true, true));
00084     //w1->writeElements()->insert(new WriteElement("temp[6]", true, true));
00085     //w1->writeElements()->insert(new WriteElement("tnow", true, true));
00086     as1->assignments()->insert(new Assign::Assignment("temp[1]", "
00087         k[1]*N[1]*N[2]"));
00088     as1->assignments()->insert(new Assign::Assignment("temp[2]", "k[2]*N[3]"))
00089     ;
00090     as1->assignments()->insert(new Assign::Assignment("temp[3]", "
00091         temp[1]+temp[2]"));
00092     as1->assignments()->insert(new Assign::Assignment("temp[4]", "if
00093         (temp[3]>0) temp[1]/temp[3] else 0"));
00094     as1->assignments()->insert(new Assign::Assignment("temp[5]", "if
00095         (temp[3]>0) (if (rnd<temp[4]) 1 else 2) else 0"));
00096     as1->assignments()->insert(new Assign::Assignment("N[1]", "
00097         N[1]+[temp[5],1]"));
00098     as1->assignments()->insert(new Assign::Assignment("N[2]", "
00099         N[2]+[temp[5],2]"));
00100    as1->assignments()->insert(new Assign::Assignment("N[3]", "
00101        N[3]+[temp[5],3]"));
00102    as1->assignments()->insert(new Assign::Assignment("temp[6]", "
00103        1-exp(-temp[3]))");
00104    as1->assignments()->insert(new Assign::Assignment("temp[7]", "expo(temp[6])
00105        "));
00106    de1->setDelayExpression("temp[7]");
00107    m->infos()->setTerminatingCondition("(N[1]+N[2]+N[3]==0");
00108    m->simulation()->start();
00109    return 0;
00110 /*
00111 Create* createl = new Create(m);
00112 Assign* assign1 = new Assign(m);
00113 //Separate* sep1 = new Separate(m);
00114 Dispose* dispose1 = new Dispose(m);
00115 createl->nextComponents()->insert(assign1);
00116 assign1->nextComponents()->insert(dispose1);
00117 //sep1->nextComponents()->insert(dispose1);
00118 //sep1->nextComponents()->insert(assign1);
00119 createl->setEntityType(new EntityType(m));
00120 createl->setMaxCreations("1");
00121 new Attribute(m, "attr1");
00122 Variable* var1 = new Variable(m, "var1");
00123 std::string expression, index;
00124 for (int i = 1; i < 3; i++) {
00125     for (int j = 1; j < 3; j++) {
00126         index = std::to_string(i) + "," + std::to_string(j);
00127         var1->setInitialValue(index, 1.0*i + j / 10.0);
00128         expression = "attr1[" + index + "] + var1[" + index + "]";
00129         assign1->assignments()->insert(new Assign::Assignment("attr1[" + index + "]", expression));
00130     }
00131 }
00132 m->simulation()->start();
00133 return 0;
00134 */
00135 }
```

7.505 TestMatricesOfAttributesAndVariables.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [TestMatricesOfAttributesAndVariables](#)

7.506 TestMatricesOfAttributesAndVariables.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  TestMatricesOfAttributesAndVariables.h
00009  * Author: rlcancian
00010 *
00011  * Created on 1 de Novembro de 2019, 18:10
00012 */
00013
00014 #ifndef TESTMATRICESOFATTRIBUTESANDVARIABLES_H
00015 #define TESTMATRICESOFATTRIBUTESANDVARIABLES_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019
00020 class TestMatricesOfAttributesAndVariables : public
00021     BaseConsoleGenesysApplication
00022 {
00023     public:
00024         TestMatricesOfAttributesAndVariables();
00025         TestMatricesOfAttributesAndVariables(const
00026             TestMatricesOfAttributesAndVariables& orig);
00027         virtual ~TestMatricesOfAttributesAndVariables();
00028         virtual int main(int argc, char** argv);
00029     private:
00030 };
00031 #endif /* TESTMATRICESOFATTRIBUTESANDVARIABLES_H */
00032
```

7.507 TestParser.cpp File Reference

```
#include "TestParser.h"
#include <string>
#include "Model.h"
#include "Simulator.h"
```

7.508 TestParser.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  TestParser.cpp
00009  * Author: rafael.luiz.cancian
```

```

00010  /*
00011  * Created on 7 de Novembro de 2018, 20:17
00012  */
00013
00014 #include "TestParser.h"
00015 #include <string>
00016 #include "Model.h"
00017 #include "Simulator.h"
00018
00019 TestParser::TestParser() {
00020 }
00021
00022
00023 int TestParser::main(int argc, char** argv) {
00024     Simulator* simulator = new Simulator();
00025     Model* model = new Model(simulator);
00026     simulator->models()->insert(model);
00027     double value;
00028     bool success;
00029     std::string errorMsg = "";
00030     value = model->parseExpression("NORM(20,10)", &success, &errorMsg);
00031     std::cout << value << std::endl;
00032     value = model->parseExpression("-10+1+2+3", &success, &errorMsg);
00033     std::cout << value << std::endl;
00034     value = model->parseExpression("1+2+3-10", &success, &errorMsg);
00035     std::cout << value << std::endl;
00036     return 0;
00037 }
```

7.509 TestParser.h File Reference

```
#include "GenesysApplication_if.h"
```

Classes

- class **TestParser**

7.510 TestParser.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    TestParser.h
00009  * Author:  rafael.luiz.cancian
00010 *
00011  * Created on 7 de Novembro de 2018, 20:17
00012 */
00013
00014 #ifndef TESTPARSER_H
00015 #define TESTPARSER_H
00016
00017 #include "GenesysApplication_if.h"
00018
00019 class TestParser : public GenesysApplication_if {
00020 public:
00021     TestParser();
00022     virtual ~TestParser() = default;
00023 public:
00024     virtual int main(int argc, char** argv);
00025 private:
00026
00027 };
00028
00029 #endif /* TESTPARSER_H */
00030
```

7.511 TestSimulationControlAndSimulationResponse.cpp File Reference

```
#include "TestSimulationControlAndSimulationResponse.h"
#include "Simulator.h"
#include "Model.h"
#include "SimulationControl.h"
#include "SimulationResponse.h"
#include <iostream>
```

7.512 TestSimulationControlAndSimulationResponse.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  TestSimulationControlAndSimulationResponse.cpp
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 17:44
00012 */
00013
00014 #include "TestSimulationControlAndSimulationResponse.h"
00015 #include "Simulator.h"
00016 #include "Model.h"
00017 #include "SimulationControl.h"
00018 #include "SimulationResponse.h"
00019 #include <iostream>
00020
00021 TestSimulationControlAndSimulationResponse::TestSimulationControlAndSimulationResponse
00022 {}
00023
00024
00025 int TestSimulationControlAndSimulationResponse::main(int
00026     argc, char** argv){
00027     Simulator* simulator = new Simulator();
00028     TraceManager* tm = simulator->tracer();
00029     this->setDefaultTraceHandlers(tm);
00030     tm->setTraceLevel(Util::TraceLevel::componentDetailed);
00031     Model* model = new Model(simulator);
00032     model->show();
00033
00034     std::cout << "NumRepl antes: " << model->infos()->numberOfReplications() <<
00035         std::endl;
00036     model->infos()->setNumberOfReplications(10);
00037     std::cout << "NumRepl depois: " << model->infos()->numberOfReplications() <<
00038         std::endl;
00039     SimulationControl* control = model->controls()->
00040         front();
00041     std::cout << control->name() << " antes: " << control->value() << std::endl;
00042     control->setValue(20);
00043     std::cout << control->name() << " depois: " << control->value() << std::endl;
00044     std::cout << "NumRepl depois: " << model->infos()->numberOfReplications() <<
00045         std::endl;
00046     return 0;
00047 }
```

7.513 TestSimulationControlAndSimulationResponse.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [TestSimulationControlAndSimulationResponse](#)

7.514 TestSimulationControlAndSimulationResponse.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  TestSimulationControlAndSimulationResponse.h
00009  * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 17:44
00012 */
00013
00014 #ifndef TESTSIMULATIONCONTROLANDSIMULATIONRESPONSE_H
00015 #define TESTSIMULATIONCONTROLANDSIMULATIONRESPONSE_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class TestSimulationControlAndSimulationResponse: public
00020 BaseConsoleGenesysApplication {
00021 public:
00022     TestSimulationControlAndSimulationResponse();
00023     virtual ~TestSimulationControlAndSimulationResponse() =
00024     default;
00025 public:
00026     virtual int main(int argc, char** argv);
00027 private:
00028 };
00029 #endif /* TESTSIMULATIONCONTROLANDSIMULATIONRESPONSE_H */
00030

```

7.515 TestStatistics.cpp File Reference

```

#include "TestStatistics.h"
#include <fstream>
#include <iostream>
#include "Traits.h"

```

7.516 TestStatistics.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  TestStatistics.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 30 de Agosto de 2018, 19:28
00012 */
00013
00014 #include "TestStatistics.h"
00015 #include <fstream>
00016 #include <iostream>
00017
00018 #include "Traits.h"
00019
00020 TestStatistics::TestStatistics() {
00021 }
00022
00023 int TestStatistics::main(int argc, char** argv) {
00024     Statistics_if* stat = new Traits<Statistics_if>::Implementation
00025     ();
00026     std::ifstream datafile;
00027     datafile.open("datafileLarge.dat");
00028     //double value;
00029     //while (datafile >> value) {
00030     //    stat->getCollector()->addValue(value);
00031 }

```

```

00030     //)
00031     stat->getCollector()->addValue(1001);
00032     stat->getCollector()->addValue(1002);
00033     stat->getCollector()->addValue(1007);
00034     std::cout << "Elems\t" << "Average\t" << "StdDev\t" << "CoefVar\t" << "Min\t" << "Max\t" << std::endl;
00035     std::cout << stat->numElements() << "\t"
00036     << stat->average() << "\t"
00037     << stat->stddeviation() << "\t"
00038     << stat->variationCoef() << "\t"
00039     << stat->min() << "\t"
00040     << stat->max() << "\t"
00041     << std::endl;
00042     //datafile.close();
00043     return 0;
00044 }
00045

```

7.517 TestStatistics.h File Reference

```
#include "GenesysApplication_if.h"
```

Classes

- class **TestStatistics**

7.518 TestStatistics.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    TestStatistics.h
00009  * Author: rafael.luiz.cancian
00010  *
00011  * Created on 30 de Agosto de 2018, 19:28
00012 */
00013
00014 #ifndef TESTSTATISTICS_H
00015 #define TESTSTATISTICS_H
00016
00017 #include "GenesysApplication_if.h"
00018
00019 class TestStatistics : public GenesysApplication_if {
00020 public:
00021     TestStatistics();
00022 public:
00023     int main(int argc, char** argv);
00024 };
00025
00026
00027 #endif /* TESTSTATISTICS_H */
00028

```

7.519 ThirdExampleOfSimulation.cpp File Reference

```
#include "ThirdExampleOfSimulation.h"
#include "Simulator.h"
#include "Create.h"
#include "Seize.h"
#include "Delay.h"
#include "Release.h"
#include "Dispose.h"
#include "EntityType.h"
```

7.520 ThirdExampleOfSimulation.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: ThirdExampleOfSimulation.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 24 de Setembro de 2019, 16:43
00012 */
00013
00014 #include "ThirdExampleOfSimulation.h"
00015
00016 // you have to included need libs
00017
00018 // GEnSyS Simulator
00019 #include "Simulator.h"
00020
00021 // Model Components
00022 #include "Create.h"
00023 #include "Seize.h"
00024 #include "Delay.h"
00025 #include "Release.h"
00026 #include "Dispose.h"
00027
00028 // Model elements
00029 #include "EntityType.h"
00030
00031 ThirdExampleOfSimulation::ThirdExampleOfSimulation() {
00032 }
00033
00034 int ThirdExampleOfSimulation::main(int argc, char** argv) {
00035     Simulator* simulator = new Simulator();
00036     // Handle traces and simulation events to output them
00037     TraceManager* tm = simulator->tracer();
00038     this->setDefaultTraceHandlers(tm);
00039     tm->setTraceLevel(Util::TraceLevel::componentDetailed);
00040     this->insertFakePluginsByHand(simulator);
00041     bool wantToCreateNewModelAndSaveInsteadOfJustLoad = true;
00042     Model* model;
00043     if (wantToCreateNewModelAndSaveInsteadOfJustLoad) {
00044         model = new Model(simulator);
00045         // build the simulation model
00046         // set model infos
00047         ModelInfo* infos = model->infos();
00048         infos->setReplicationLength(60);
00049         infos->setReplicationLengthTimeUnit(
00050             Util::TimeUnit::second);
00051         infos->setNumberOfReplications(3);
00052         //
00053         EntityType* customer = new EntityType(model, "Customer");
00054         // model->insert(customer);
00055         //
00056         Create* createl = new Create(model);
00057         createl->setEntityType(customer);
00058         createl->setTimeBetweenCreationsExpression("norm(3,1)");
00059         createl->setTimeUnit(Util::TimeUnit::second);
00060         createl->setEntitiesPerCreation(1);
00061         // model->insert(createl);
00062         //
00063         Resource* machine1 = new Resource(model, "Machine_1");
00064         machine1->setCapacity(1);
00065         // model->insert(machine1);
00066         //
00067         Queue* queueSeizel = new Queue(model, "Queue_Machine_1");
00068         queueSeizel->setOrderRule(Queue::OrderRule::FIFO);
00069         // model->insert(queueSeizel);
00070         //
00071         Seize* seizel = new Seize(model);
00072         seizel->setResource(machine1);
00073         seizel->setQueue(queueSeizel);
00074         // model->insert(seizel);
00075         //
00076         Delay* delay1 = new Delay(model);
00077         delay1->setDelayExpression("norm(3,1)");
00078         delay1->setDelayTimeUnit(Util::TimeUnit::second);
00079         // model->insert(delay1);
00080         //
00081         Release* release1 = new Release(model);
00082         release1->setResource(machine1);
00083         // model->insert(release1);
00084         //
00085         Dispose* dispose1 = new Dispose(model);
00086         dispose1->setResource(machine1);
00087         // model->insert(dispose1);
00088         //

```

```

00088     Dispose* dispose1 = new Dispose(model);
00089     // model->insert(dispose1);
00090     // connect model components to create a "workflow"
00091     create1->nextComponents()->insert(seize1);
00092     seize1->nextComponents()->insert(delay1);
00093     delay1->nextComponents()->insert(release1);
00094     release1->nextComponents()->insert(dispose1);
00095     // insert the model into the simulator
00096     simulator->models()->insert(model);
00097     // if the model is ok then save the model into a text file
00098     if (model->check()) {
00099         model->save("./temp/thirdExampleOfSimulation.txt");
00100    }
00101 } else {
00102     simulator->models()->loadModel("./temp/thirdExampleOfSimulation.txt");
00103     model = simulator->models()->current();
00104 }
00105 // execute the simulation
00106 model->simulation()->start();
00107
00108 return 0;
00109 };

```

7.521 ThirdExampleOfSimultion.h File Reference

```
#include "BaseConsoleGenesysApplication.h"
```

Classes

- class [ThirdExampleOfSimulation](#)

7.522 ThirdExampleOfSimultion.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File: ThirdExampleOfSimultion.h
00009  * Author: rlcancian
00010 *
00011  * Created on 24 de Setembro de 2019, 16:43
00012 */
00013
00014 #ifndef THIRDEXAMPLEOFSIMULTION_H
00015 #define THIRDEXAMPLEOFSIMULTION_H
00016
00017 #include "BaseConsoleGenesysApplication.h"
00018
00019 class ThirdExampleOfSimulation : public
00020     BaseConsoleGenesysApplication{
00021     public:
00022         ThirdExampleOfSimulation();
00023     public:
00024         virtual int main(int argc, char** argv);
00025     };
00026 #endif /* THIRDEXAMPLEOFSIMULTION_H */
00027

```

7.523 ToolManager.cpp File Reference

```
#include "ToolManager.h"
#include "Traits.h"
```

7.524 ToolManager.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    ToolManager.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Maio de 2019, 09:04
00012 */
00013
00014 #include "ToolManager.h"
00015 #include "Traits.h"
00016
00017
00018 ToolManager::ToolManager(Simulator* simulator) {
00019     _simulator = simulator;
00020     //
00021     _fitter = new Traits<Fitter_if>::Implementation();
00022     _sampler = new Traits<Sampler_if>::Implementation();
00023     _processAnalyser = new Traits<ProcessAnalyser_if>::Implementation
00024     ();
00025     Sampler_if* ToolManager::sampler() const {
00026         return _sampler;
00027     }
00028
00029     Fitter_if* ToolManager::fitter() const {
00030         return _fitter;
00031     }
00032
00033     ProcessAnalyser_if* ToolManager::experimentDesigner()
00034         const {
00035             return _processAnalyser;
00036         }

```

7.525 ToolManager.h File Reference

```

#include "Sampler_if.h"
#include "Fitter_if.h"
#include "ProcessAnalyser_if.h"

```

Classes

- class [ToolManager](#)

7.526 ToolManager.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    ToolManager.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 31 de Maio de 2019, 09:04
00012 */
00013
00014 #ifndef TOOLMANAGER_H
00015 #define TOOLMANAGER_H
00016
00017 #include "Sampler_if.h"
00018 #include "Fitter_if.h"

```

```

00019 #include "ProcessAnalyser_if.h"
00020
00021 class Simulator;
00022
00023 class ToolManager {
00024 public:
00025     ToolManager(Simulator* _simulator);
00026     virtual ~ToolManager() = default;
00027 public:
00028     Sampler_if* sampler() const;
00029     Fitter_if* fitter() const;
00030     ProcessAnalyser_if* experimentDesigner() const;
00031 public: // event handlers
00032 private:
00033     Fitter_if* _fitter; // = new Traits<Fitter_if>::Implementation();
00034     Sampler_if* _sampler; // = new Traits<Sampler_if>::Implementation();
00035     ProcessAnalyser_if* _processAnalyser;
00036 private:
00037     Simulator* _simulator;
00038 };
00039
00040 #endif /* TOOLMANAGER_H */
00041

```

7.527 TraceManager.cpp File Reference

```
#include "TraceManager.h"
#include "Traits.h"
```

7.528 TraceManager.cpp

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File:    TraceManager.cpp
00009 * Author:  rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 11:59
00012 */
00013
00014 #include "TraceManager.h"
00015 #include "Traits.h"
00016
00017 TraceManager::TraceManager(Simulator* simulator) { // (Model* model) {
00018     _simulator = simulator;
00019     _debugged = Traits<Model>::debugged;
00020     _traceLevel = Traits<Model>::traceLevel;
00021 }
00022
00023 void TraceManager::setTraceLevel(Util::TraceLevel _traceLevel) {
00024     this->_traceLevel = _traceLevel;
00025 }
00026
00027 Util::TraceLevel TraceManager::traceLevel() const {
00028     return _traceLevel;
00029 }
00030
00031 Simulator* TraceManager::parentSimulator() const {
00032     return _simulator;
00033 }
00034
00035 /*
00036 void TraceManager::traceSimulation(Util::TraceLevel level, std::string text) {
00037     if (_traceConditionPassed(tracelevel)) {
00038         TraceSimulationEvent e = TraceEvent(tracelevel, text);
00039         for (std::list::iterator it =
00040             this->_traceSimulationHandlers->list()->begin(); it != _traceSimulationHandlers->list()->end(); it++) {
00041             (*it)(e);
00042         }
00043     }
00044 */

```

```
00045
00046
00047
00048 void TraceManager::addTraceHandler(traceListener
    traceListener) {
00049     this->_traceHandlers->insert(traceListener);
00050 }
00051
00052 //void TraceManager::addTraceSimulationHandler(traceListener traceListener) {
00053 //    this->_traceSimulationHandlers->insert(_traceSimulationHandlers->list()->end(), traceListener);
00054 //}
00055
00056 void TraceManager::addTraceSimulationHandler(
    traceSimulationListener traceSimulationListener) {
00057     this->_traceSimulationHandlers->insert(traceSimulationListener);
00058 }
00059
00060 void TraceManager::addTraceErrorHandler(
    traceErrorListener traceErrorListener) {
00061     this->_traceErrorHandlers->insert(traceErrorListener);
00062 }
00063
00064 void TraceManager::addTraceReportHandler(
    traceListener traceReportListener) {
00065     this->_traceReportHandlers->insert(traceReportListener);
00066 }
00067
00068 void TraceManager::trace(Util::TraceLevel level, std::string text) {
00069     trace(text, level);
00070 }
00071
00072 void TraceManager::trace(std::string text, Util::TraceLevel level) {
00073     if (_traceConditionPassed(level)) {
00074         //text = std::to_string(static_cast<int> (level)) + ". " + Util::Indent() + text;
00075         TraceEvent e = TraceEvent(level, text);
00076         /* \todo--- somewhere in future it should be interesting to use "auto" and c++17 at least */
00077         for (std::list<traceListener>::iterator it = this->_traceHandlers->list()->begin(); it != _traceHandlers->list()->end(); it++) {
00078             (*it)(e);
00079         }
00080         for (std::list<traceListenerMethod>::iterator it = this->_traceHandlersMethod->list()->begin(); it != _traceHandlersMethod->list()->end(); it++) {
00081             (*it)(e);
00082         }
00083     }
00084 }
00085 }
00086
00087 void TraceManager::traceError(std::exception e, std::string text) {
00088     traceError(text, e);
00089 }
00090
00091 void TraceManager::traceError(std::string text, std::exception e) {
00092     TraceErrorEvent exceptEvent = TraceErrorEvent(text, e);
00093     /* \todo--- somewhere in future it should be interesting to use "auto" and c++17 at least */
00094     for (std::list<traceErrorListener>::iterator it = this->_traceErrorHandlers->list()->begin(); it != _traceErrorHandlers->list()->end(); it++) {
00095         (*it)(exceptEvent);
00096     }
00097     for (std::list<traceErrorListenerMethod>::iterator it = this->_traceErrorHandlersMethod->list()->begin(); it != _traceErrorHandlersMethod->list()->end(); it++) {
00098         (*it)(exceptEvent);
00099     }
00100 }
00101
00102 void TraceManager::traceSimulation(Util::TraceLevel level,
    double time, Entity* entity, ModelComponent* component, std::string text) {
00103     traceSimulation(time, entity, component, text, level);
00104 }
00105
00106 void TraceManager::traceSimulation(double time,
    Entity* entity, ModelComponent* component, std::string text,
    Util::TraceLevel level) {
00107     if (_traceConditionPassed(level)) {
00108         text = Util::Indent() + text;
00109         TraceSimulationEvent e = TraceSimulationEvent(level, time,
00110             entity, component, text);
00111         for (std::list<traceSimulationListener>::iterator it = this->_traceSimulationHandlers->list()->begin(); it != _traceSimulationHandlers->list()->end(); it++) {
00112             (*it)(e);
00113         }
00114         for (std::list<traceSimulationListenerMethod>::iterator it = this->_traceSimulationHandlersMethod->list()->begin(); it != _traceSimulationHandlersMethod->list()->end(); it++) {
00115             (*it)(e);
00116         }
00117 }
```

```

00118
00119 void TraceManager::traceReport(Util::TraceLevel level, std::string
00120     text) {
00121     traceReport(text, level);
00122 }
00123 void TraceManager::traceReport(std::string text,
00124     Util::TraceLevel level) {
00125     if (_traceConditionPassed(level)) {
00126         text = Util::Indent() + text;
00127         TraceEvent e = TraceEvent(level, text);
00128         for (std::list<traceListener>::iterator it = this->_traceReportHandlers->
00129             list()->begin(); it != _traceReportHandlers->list()->end(); it++) {
00130             (*it)(e);
00131         }
00132         for (std::list<traceListenerMethod>::iterator it = this->_traceReportHandlersMethod->
00133             list()->begin(); it != _traceReportHandlersMethod->list()->end(); it++) {
00134             (*it)(e);
00135         }
00136     List<std::string>* TraceManager::errorMessages() const {
00137         return _errorMessages;
00138     }
00139
00140 bool TraceManager::_traceConditionPassed(Util::TraceLevel level) {
00141     return this->_debugged && static_cast<int> (this->_traceLevel) >= static_cast<int> (level);
00142 }
```

7.529 TraceManager.h File Reference

```
#include "List.h"
#include <functional>
```

Classes

- class [TraceEvent](#)
- class [TraceErrorEvent](#)
- class [TraceSimulationEvent](#)
- class [TraceSimulationProcess](#)
- class [TraceManager](#)

TypeDefs

- typedef void(* [traceListener](#)) ([TraceEvent](#))
- typedef void(* [traceErrorListener](#)) ([TraceErrorEvent](#))
- typedef void(* [traceSimulationListener](#)) ([TraceSimulationEvent](#))
- typedef void(* [traceSimulationProcessListener](#)) ([TraceSimulationProcess](#))
- typedef std::function< void([TraceEvent](#)) > [traceListenerMethod](#)
- typedef std::function< void([TraceErrorEvent](#)) > [traceErrorListenerMethod](#)
- typedef std::function< void([TraceSimulationEvent](#)) > [traceSimulationListenerMethod](#)
- typedef std::function< void([TraceSimulationProcess](#)) > [traceSimulationProcessListenerMethod](#)

7.529.1 TypeDef Documentation

7.529.1.1 traceErrorListener

```
typedef void(* traceErrorListener) (TraceErrorEvent)
```

Definition at line 98 of file [TraceManager.h](#).

7.529.1.2 traceErrorListenerMethod

```
typedef std::function<void(TraceErrorEvent) > traceErrorListenerMethod
```

Definition at line 103 of file [TraceManager.h](#).

7.529.1.3 traceListener

```
typedef void(* traceListener) (TraceEvent)
```

Definition at line 97 of file [TraceManager.h](#).

7.529.1.4 traceListenerMethod

```
typedef std::function<void(TraceEvent) > traceListenerMethod
```

Definition at line 102 of file [TraceManager.h](#).

7.529.1.5 traceSimulationListener

```
typedef void(* traceSimulationListener) (TraceSimulationEvent)
```

Definition at line 99 of file [TraceManager.h](#).

7.529.1.6 traceSimulationListenerMethod

```
typedef std::function<void(TraceSimulationEvent) > traceSimulationListenerMethod
```

Definition at line 104 of file [TraceManager.h](#).

7.529.1.7 traceSimulationProcessListener

```
typedef void(* traceSimulationProcessListener) (TraceSimulationProcess)
```

Definition at line 100 of file [TraceManager.h](#).

7.529.1.8 traceSimulationProcessListenerMethod

```
typedef std::function<void(TraceSimulationProcess) > traceSimulationProcessListenerMethod
```

Definition at line 105 of file [TraceManager.h](#).

7.530 TraceManager.h

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    TraceManager.h
00009  * Author: rafael.luiz.cancian
00010 *
00011 * Created on 7 de Novembro de 2018, 11:59
00012 */
00013
00014 #ifndef TRACEMANAGER_H
00015 #define TRACEMANAGER_H
00016
00017 #include "List.h"
00018 #include <functional>
00019
00020 class Model;
00021 class Entity;
00022 class ModelComponent;
00023
00024 class TraceEvent {
00025 public:
00026
00027     TraceEvent(Util::TraceLevel level, std::string
00028             text) {
00028         _tracelevel = level;
00029         _text = text;
00030     }
00031
00032     Util::TraceLevel tracelevel() const {
00033         return _tracelevel;
00034     }
00035
00036     std::string text() const {
00037         return _text;
00038     }
00039 private:
00040     Util::TraceLevel _tracelevel;
00041     std::string _text;
00042 };
00043
00044 class TraceErrorEvent : public TraceEvent {
00045 public:
00046
00047     TraceErrorEvent(std::string text, std::exception e) :
00048         TraceEvent(Util::TraceLevel::errorFatal, text) {
00049         _e = e;
00050     }
00051
00052     std::exception getException() const {
00053         return _e;
00054     }
00055 private:
00056     std::exception _e;
00057 };
00058 class TraceSimulationEvent : public TraceEvent {
00059 public:
00060
00061     TraceSimulationEvent(Util::TraceLevel level, double time,
00062                          Entity* entity, ModelComponent* component, std::string text) :
00063         TraceEvent(level, text) {
00064         _time = time;
00065         _entity = entity;
00066         _component = component;
00067     }
00068     ModelComponent* component() const {
00069         return _component;
```

```

00069     }
00070
00071     Entity* entity() const {
00072         return _entity;
00073     }
00074
00075     double time() const {
00076         return _time;
00077     }
00078
00079     private:
00080         double _time;
00081         Entity* _entity;
00082         ModelComponent* _component;
00083     };
00084
00085     class TraceSimulationProcess : public TraceEvent {
00086     public:
00087         TraceSimulationProcess(Util::TraceLevel level, std::string
00088             text) : TraceEvent(level, text) {
00089     }
00090     };
00091
00092     // for handlers that are simple functions
00093     typedef void (*traceListener)(TraceEvent);
00094     typedef void (*traceErrorListener)(TraceErrorEvent);
00095     typedef void (*traceSimulationListener)(
00096         TraceSimulationEvent);
00097     typedef void (*traceSimulationProcessListener)(
00098         TraceSimulationProcess);
00099     // for handlers that are class members (methods)
00100     typedef std::function<void(TraceEvent) > traceListenerMethod;
00101     typedef std::function<void(TraceErrorEvent) > traceErrorListenerMethod;
00102     typedef std::function<void(TraceSimulationEvent) > traceSimulationListenerMethod
00103     ;
00104     typedef std::function<void(TraceSimulationProcess) >
00105         traceSimulationProcessListenerMethod;
00106
00107     class TraceManager {
00108     public:
00109         TraceManager(Simulator* simulator); // (Model* model);
00110         virtual ~TraceManager() = default;
00111     public: // add trace handlers
00112         // for handlers that are simple functions
00113         void addTraceHandler(traceListener traceListener);
00114         void addTraceReportHandler(traceListener traceReportListener);
00115         void addTraceSimulationHandler(traceSimulationListener
00116             traceSimulationListener);
00117         void addTraceErrorHandler(traceErrorListener
00118             traceErrorListener);
00119         // for handlers that are class members (methods)
00120         template<typename Class> void addTraceHandler(Class * object, void (Class::*function) (
00121             TraceEvent));
00122         template<typename Class> void addTraceErrorHandler(Class * object, void (Class::*function) (
00123             TraceErrorEvent));
00124         template<typename Class> void addTraceReportHandler(Class * object, void (Class::*function) (
00125             TraceEvent));
00126         template<typename Class> void addTraceSimulationHandler(Class * object, void (Class::*function) (
00127             TraceSimulationEvent));
00128     public: // traces (invoke trace handlers)
00129         void trace(Util::TraceLevel level, std::string text);
00130         void traceError(std::exception e, std::string text);
00131         void traceReport(Util::TraceLevel level, std::string text);
00132         void traceSimulation(Util::TraceLevel level, double time,
00133             Entity* entity, ModelComponent* component, std::string text);
00134     public: // traces (invoke trace handlers) SINCE 20191025 NEW TRACES JUST INVERTED THE PARAMETERS, MAKING
00135         TRACELEVEL OPTIONAL
00136         void trace(std::string text, Util::TraceLevel level =
00137             Util::TraceLevel::componentInternal);
00138         void traceError(std::string text, std::exception e);
00139         void traceReport(std::string text, Util::TraceLevel level =
00140             Util::TraceLevel::report);
00141         void traceSimulation(double time, Entity* entity, ModelComponent* component,
00142             std::string text, Util::TraceLevel level =
00143             Util::TraceLevel::componentInternal);
00144     public:
00145         List<std::string>* errorMessages() const;
00146         void setTraceLevel(Util::TraceLevel _traceLevel);
00147         Util::TraceLevel traceLevel() const;
00148         Simulator* parentSimulator() const;
00149     private:
00150         //void _addHandler(List<traceListener>* list, )
00151         bool _traceConditionPassed(Util::TraceLevel level);
00152     private: // trace listener
00153         List<traceListener>* _traceHandlers = new

```

```

00147     List<traceListener>();
00148     List<traceErrorListener>* _traceErrorHandlers = new
00149     List<traceErrorListener>();
00150     List<traceListener>();
00151     List<traceSimulationListener>* _traceSimulationHandlers = new
00152     List<traceSimulationListener>();
00153     // for handlers that are class members (methods)
00154     List<traceListenerMethod>* _traceHandlersMethod = new
00155     List<traceListenerMethod>();
00156     List<traceErrorListenerMethod>* _traceErrorHandlersMethod = new
00157     List<traceErrorListenerMethod>();
00158     List<traceListenerMethod>();
00159     List<traceSimulationListenerMethod>* _traceSimulationHandlersMethod
00160     = new List<traceSimulationListenerMethod>();
00161     private:
00162     //Model* _model;
00163     Simulator* _simulator;
00164     protected:
00165     Util::TraceLevel _traceLevel; // = Util::TraceLevel::mostDetailed;
00166     bool _debugged;
00167     double _lastTimeTraceSimulation = -1.0;
00168     Util::identification _lastEntityTraceSimulation = 0;
00169     Util::identification _lastModuleTraceSimulation = 0;
00170     List<std::string>* _errorMessages; /* \todo: 18/08/24 this is a new one. several
00171     methods should use it */
00172     };
00173     // implementation for template methods
00174     template<typename Class> void TraceManager::addTraceHandler(Class * object,
00175         void (Class::*function)(TraceEvent)) {
00176         this->_traceHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
00177     }
00178     template<typename Class> void TraceManager::addTraceErrorHandler(Class *
00179         object, void (Class::*function)(TraceErrorEvent)) {
00180         this->_traceErrorHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
00181     }
00182     template<typename Class> void TraceManager::addTraceReportHandler(Class
00183         * object, void (Class::*function)(TraceEvent)) {
00184         this->_traceReportHandlersMethod->insert(std::bind(function, object, std::placeholders::_1));
00185     }
00186 #endif /* TRACEMANAGER_H */
00187

```

7.531 Traits.h File Reference

```

#include "Model.h"
#include "Collector_if.h"
#include "Sampler_if.h"
#include "Fitter_if.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "Statistics_if.h"
#include "Integrator_if.h"
#include "HypothesisTester_if.h"
#include "ModelPersistence_if.h"
#include "GenesysApplication_if.h"
#include "ProcessAnalyser_if.h"
#include "ExperimentDesign_if.h"
#include "SimulationReporter_if.h"
#include "PluginConnector_if.h"
#include "FullSimulationOfComplexModel.h"
#include "BuildSimulationModel03.h"

```

```
#include "FirstExampleOfSimulation.h"
#include "SecondExampleOfSimulation.h"
#include "ThirdExampleOfSimulation.h"
#include "FourthExampleOfSimulation.h"
#include "GenesysGUI.h"
#include "GenesysConsole.h"
#include "TestEnterLeaveRoute.h"
#include "TestFunctions.h"
#include "TestSimulationControlAndSimulationResponse.h"
#include "TestLSODE.h"
#include "TestMarkovChain.h"
#include "TestMatricesOfAttributesAndVariables.h"
#include "CollectorDefaultImpl1.h"
#include "CollectorDatafileDefaultImpl1.h"
#include "StatisticsDefaultImpl1.h"
#include "IntegratorDefaultImpl1.h"
#include "HypothesisTesterDefaultImpl1.h"
#include "SamplerDefaultImpl1.h"
#include "parserBisonFlex/ParserDefaultImpl2.h"
#include "PluginConnectorDummyImpl1.h"
#include "SimulationReporterDefaultImpl1.h"
#include "ModelCheckerDefaultImpl1.h"
#include "ModelPersistenceDefaultImpl1.h"
#include "ExperimentDesignDefaultImpl1.h"
#include "ProcessAnalyserDefaultImpl1.h"
#include "FitterDefaultImpl1.h"
```

Classes

- struct Traits< T >
- struct Traits< GenesysApplication_if >
- struct Traits< PluginConnector_if >
- struct Traits< Parser_if >
- struct Traits< Model >
- struct Traits< ModelPersistence_if >
- struct Traits< SimulationReporter_if >
- struct Traits< ModelChecker_if >
- struct Traits< ModelComponent >
- struct Traits< Collector_if >
- struct Traits< Statistics_if >
- struct Traits< Integrator_if >
- struct Traits< Sampler_if >
- struct Traits< Fitter_if >
- struct Traits< HypothesisTester_if >
- struct Traits< ExperimentDesign_if >
- struct Traits< ProcessAnalyser_if >

7.532 Traits.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
```

```
00007 /*
00008  * File: Traits.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 14 de Agosto de 2018, 19:36
00012 */
00013
00014 #ifndef TRAITS_H
00015 #define TRAITS_H
00016
00017 // interfaces
00018 #include "Model.h"
00019 #include "Collector_if.h"
00020 #include "Sampler_if.h"
00021 #include "Fitter_if.h"
00022 #include "ModelChecker_if.h"
00023 #include "Parser_if.h"
00024 #include "Statistics_if.h"
00025 #include "Integrator_if.h"
00026 #include "HypothesisTester_if.h"
00027 #include "ModelPersistence_if.h"
00028 #include "GenesysApplication_if.h"
00029 #include "ProcessAnalyser_if.h"
00030 #include "ExperimentDesign_if.h"
00031 #include "SimulationReporter_if.h"
00032 #include "PluginConnector_if.h"
00033
00034 // genesys applications
00035 #include "FullSimulationOfComplexModel.h"
00036 #include "BuildSimulationModel03.h"
00037 #include "FirstExampleOfSimulation.h"
00038 #include "SecondExampleOfSimulation.h"
00039 #include "ThirdExampleOfSimulation.h"
00040 #include "FourthExampleOfSimulation.h"
00041 #include "GenesysGUI.h"
00042 #include "GenesysConsole.h"
00043 #include "TestEnterLeaveRoute.h"
00044 #include "TestFunctions.h"
00045 #include "TestSimulationControlAndSimulationResponse.h"
00046 #include "TestLSODE.h"
00047 #include "TestMarkovChain.h"
00048 #include "TestMatricesOfAttributesAndVariables.h"
00049
00050 // Default implementations
00051 //statistics
00052 #include "CollectorDefaultImpl.h"
00053 #include "CollectorDatafileDefaultImpl.h"
00054 #include "StatisticsDefaultImpl.h"
00055 #include "IntegratorDefaultImpl.h"
00056 #include "HypothesisTesterDefaultImpl.h"
00057 #include "SamplerDefaultImpl.h"
00058 //simulator and parts
00059 #include "parserBisonFlex/ParserDefaultImpl2.h"
00060 #include "PluginConnectorDummyImpl.h"
00061 //model and parts
00062 #include "SimulationReporterDefaultImpl.h"
00063 #include "ModelCheckerDefaultImpl.h"
00064 #include "ModelPersistenceDefaultImpl.h"
00065 //tools
00066 #include "ExperimentDesignDefaultImpl.h"
00067 #include "ProcessAnalyserDefaultImpl.h"
00068 #include "FitterDefaultImpl.h"
00069
00070 template <typename T>
00071 struct Traits {
00072 };
00073 /*
00074  * Genesys Application
00075 */
00076 */
00077
00078 template <> struct Traits<GenesysApplication_if> {
00079     //typedef GenesysGUI Application;
00080     //typedef GenesysConsole Application;
00081     //typedef FullSimulationOfComplexModel Application;
00082     typedef FirstExampleOfSimulation Application;
00083     //typedef SecondExampleOfSimulation Application;
00084     //typedef FourthExampleOfSimulation Application;
00085     //typedef TestLSODE Application;
00086     //typedef TestMarkovChain Application;
00087     //typedef TestSimulationControlAndSimulationResponse Application;
00088     //typedef TestMatricesOfAttributesAndVariables Application;
00089 };
00090
00091
00092
00093 */
```

```
00094 * Simulator and Simulator Parts
00095 */
00096
00097
00098 template <> struct Traits<PluginConnector_if> {
00099     typedef PluginConnectorDummyImpl1 Implementation;
00100 };
00101
00102 template <> struct Traits<Parser_if> {
00103     typedef ParserDefaultImpl2 Implementation;
00104 };
00105
00106
00107 /*
00108 * Model and Model Parts
00109 */
00110
00111 template <> struct Traits<Model> {
00112     static const bool debugged = true;
00113     static const Util::TraceLevel traceLevel =
00114         Util::TraceLevel::modelSimulationEvent;
00115 };
00116 template <> struct Traits<ModelPersistence_if> {
00117     typedef ModelPersistenceDefaultImpl1
00118         Implementation;
00119 };
00120 template <> struct Traits<SimulationReporter_if> {
00121     typedef SimulationReporterDefaultImpl1
00122         Implementation;
00123 };
00124 template <> struct Traits<ModelChecker_if> {
00125     typedef ModelCheckerDefaultImpl1 Implementation;
00126 };
00127
00128 template <> struct Traits<ModelComponent> {
00129     typedef StatisticsDefaultImpl1
00130         StatisticsCollector_StatisticsImplementation;
00131     typedef CollectorDefaultImpl1
00132         StatisticsCollector_CollectorImplementation;
00133 };
00134 /*
00135 * Statistics
00136 */
00137 template <> struct Traits<Collector_if> {
00138     typedef CollectorDatafileDefaultImpl1
00139         Implementation;
00140 };
00141 template <> struct Traits<Statistics_if> {
00142     typedef StatisticsDefaultImpl1 Implementation;
00143     typedef CollectorDefaultImpl1 CollectorImplementation;
00144     static constexpr double SignificanceLevel = 0.05;
00145 };
00146
00147 template <> struct Traits<Integrator_if> {
00148     typedef IntegratorDefaultImpl1 Implementation;
00149     static constexpr unsigned int MaxIterations = 1e3;
00150     static constexpr double Precision = 1e-9;
00151 };
00152
00153 template <> struct Traits<Sampler_if> {
00154     typedef SamplerDefaultImpl1 Implementation;
00155     typedef SamplerDefaultImpl1::DefaultImplRNG_Parameters
00156         Parameters;
00157 };
00158 template <> struct Traits<Fitter_if> {
00159     typedef FitterDefaultImpl1 Implementation;
00160 };
00161
00162 template <> struct Traits<HypothesisTester_if> {
00163     typedef IntegratorDefaultImpl1
00164         IntegratorImplementation;
00165     typedef HypothesisTesterDefaultImpl1
00166         Implementation;
00167 };
00168 /*
00169 * Tools
00170 */
00171 template <> struct Traits<ExperimentDesign_if> {
```

```

00172     typedef ExperimentDesignDefaultImpl1
00173     Implementation;
00174 }
00175 template <> struct Traits<ProcessAnalyser_if> {
00176     typedef ProcessAnalyserDefaultImpl1
00177     Implementation;
00178 };
00179 #endif /* TRAITS_H */
00180

```

7.533 Unstore.cpp File Reference

```
#include "Unstore.h"
#include "Model.h"
```

7.534 Unstore.cpp

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Unstore.cpp
00009  * Author: rlcancian
00010  *
00011  * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #include "Unstore.h"
00015 #include "Model.h"
00016
00017 Unstore::Unstore(Model* model, std::string name) :
00018     ModelComponent(model, Util::TypeOf<Unstore>(), name) {
00019
00020
00021     std::string Unstore::show() {
00022         return ModelComponent::show() + "";
00023     }
00024
00025     ModelComponent* Unstore::LoadInstance(Model* model,
00026         std::map<std::string, std::string>* fields) {
00027         Unstore* newComponent = new Unstore(model);
00028         try {
00029             newComponent->_loadInstance(fields);
00030         } catch (const std::exception& e) {
00031
00032         return newComponent;
00033     }
00034
00035     void Unstore::_execute(Entity* entity) {
00036         _parentModel->tracer()->trace("I'm just a dummy model and I'll just send the
00037             entity forward");
00038         this->_parentModel->sendEntityToComponent(entity, this-
00039             nextComponents()->frontConnection(), 0.0);
00040     }
00041
00042     bool Unstore::_loadInstance(std::map<std::string, std::string>* fields) {
00043         ModelComponent::_loadInstance(fields);
00044         if (res) {
00045             //...
00046         }
00047         return res;
00048     }
00049
00050     void Unstore::_initBetweenReplications() {
00051         std::map<std::string, std::string>* Unstore::_saveInstance() {
00052             std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00053             ();

```

```

00053     //...
00054     return fields;
00055 }
00056
00057 bool Unstore::_check(std::string* errorMessage) {
00058     bool resultAll = true;
00059     //...
00060     return resultAll;
00061 }
00062
00063 PluginInformation* Unstore::GetPluginInformation() {
00064     PluginInformation* info = new PluginInformation(Util::TypeOf<Unstore>
00065     (), &Unstore::LoadInstance);
00066     // ...
00067     return info;
00068 }
00069

```

7.535 Unstore.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [Unstore](#)

7.536 Unstore.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:    Unstore.h
00009  * Author:  rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:08
00012 */
00013
00014 #ifndef UNSTORE_H
00015 #define UNSTORE_H
00016
00017 #include "ModelComponent.h"
00018
00045 class Unstore : public ModelComponent {
00046 public: // constructors
00047     Unstore(Model* model, std::string name = "");
00048     virtual ~Unstore() = default;
00049 public: // virtual
00050     virtual std::string show();
00051 public: // static
00052     static PluginInformation* GetPluginInformation();
00053     static ModelComponent* LoadInstance(Model* model, std::map<std::string,
00054                                         std::string>* fields);
00054 protected: // virtual
00055     virtual void _execute(Entity* entity);
00056     virtual void _initBetweenReplications();
00057     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00058     virtual std::map<std::string, std::string>* _saveInstance();
00059     virtual bool _check(std::string* errorMessage);
00060 private: // methods
00061 private: // attributes 1:1
00062 private: // attributes 1:n
00063 };
00064
00065
00066 #endif /* UNSTORE_H */
00067

```

7.537 Util.cpp File Reference

```
#include "Util.h"
#include <typeinfo>
#include <map>
```

7.538 Util.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006 /*
00007 */
00008 * File: Util.cpp
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 13:02
00012 */
00013
00014 #include "Util.h"
00015 #include <typeinfo>
00016 #include <map>
00017
00018
00019 Util::identification Util::_S_lastId = 0;
00020 unsigned int Util::_S_indentation;
00021 std::map<std::string, Util::identification> Util::_S_lastIdOfType = std::map<std::string,
Util::identification>();
00022 std::map<std::string, std::string> Util::_S_TypeOf = std::map<std::string, std::string>();
00023 /*
00024 std::string Util::ToStrTimeUnit(TimeUnit tu) {
00025     switch (tu) {
00026         case Util::TimeUnit::picosecond: return "picosecond";
00027         case Util::TimeUnit::nanosecond: return "nanosecond";
00028         case Util::TimeUnit::microsecond: return "microsecond";
00029         case Util::TimeUnit::millisecond: return "millisecond";
00030         case Util::TimeUnit::second: return "second";
00031         case Util::TimeUnit::minute: return "minute";
00032         case Util::TimeUnit::hour: return "hour";
00033         case Util::TimeUnit::day: return "day";
00034         case Util::TimeUnit::week: return "week";
00035         default: return "";
00036     }
00037 }
00038 */
00039
00040 void Util::IncIndent() {
00041     Util::_S_indentation++;
00042 }
00043
00044 void Util::SetIndent(const unsigned short indent) {
00045     Util::_S_indentation = indent;
00046 }
00047
00048 void Util::DecIndent() {
00049     Util::_S_indentation--;
00050 }
00051
00052 void Util::SepKeyVal(std::string str, std::string *key, std::string *value) {
00053     // \todo: Check pointers when splitting string. There is an error
00054     //char *c;
00055     bool settingKey = true;
00056     //key = new std::string();
00057     //value = new std::string();
00058     key->clear();
00059     value->clear();
00060     for (std::string::iterator it = str.begin(); it != str.end(); it++) {
00061         if (settingKey) {
00062             if ((*it) != '=') {
00063                 key->append(new char((*it)));
00064             } else {
00065                 settingKey = false;
00066             }
00067         } else {
00068             value->append(new char((*it)));
00069     }
```

```

00070     }
00071 }
00072
00073 std::string Util::Indent() {
00074     std::string spaces = "";
00075     for (unsigned int i = 0; i < Util::_S_indentation; i++) {
00076         spaces += " | ";
00077     }
00078     return spaces;
00079 }
00080
00081 std::string Util::SetW(std::string text, unsigned short width) {
00082     std::string spaces(width, ' ');
00083     std::string result = text + spaces;
00084     return result.substr(0, width);
00085 }
00086
00087 std::string Util::StrTimeUnit(Util::TimeUnit timeUnit) {
00088     switch (static_cast<int> (timeUnit)) {
00089     case 1: return "picosecond";
00090     case 2: return "nanosecond";
00091     case 3: return "microsecond";
00092     case 4: return "millisecond";
00093     case 5: return "second";
00094     case 6: return "minute";
00095     case 7: return "hour";
00096     case 8: return "day";
00097     case 9: return "week";
00098     }
00099     return "";
00100 }
00101
00102 Util::identification Util::GenerateNewId() {
00103     Util::_S_lastId++;
00104     return Util::_S_lastId;
00105 }
00106
00107 Util::identification Util::GenerateNewIdOfType(std::string
    objtype) {
00108     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00109     if (it == Util::_S_lastIdOfType.end()) {
00110         // a new one. create the pair
00111         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00112         it = Util::_S_lastIdOfType.find(objtype);
00113     }
00114     Util::identification next = (*it).second;
00115     next++;
00116     (*it).second = next;
00117     return (*it).second;
00118 }
00119
00120 Util::identification Util::GetLastIdOfType(std::string objtype) {
00121     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00122     if (it == Util::_S_lastIdOfType.end()) {
00123         // a new one. create the pair
00124         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00125         it = Util::_S_lastIdOfType.find(objtype);
00126     }
00127     //Util::identification next = (*it).second;
00128     //next++;
00129     //(*it).second = next;
00130     return (*it).second;
00131
00132 }
00133
00134 void Util::ResetIdOfType(std::string objtype) {
00135     std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00136     if (it == Util::_S_lastIdOfType.end()) {
00137         // a new one. create the pair
00138         Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00139         it = Util::_S_lastIdOfType.find(objtype);
00140     }
00141     (*it).second = 0;
00142 }
00143
00144 double Util::TimeUnitConvert(Util::TimeUnit timeUnit1,
    Util::TimeUnit timeUnit2) {
00145     double scaleValues[] = {1.0, 1000.0, 1000.0, 1000.0, 1000.0, 60.0, 60.0, 24.0, 7.0};
00146     // TU_picosecond = 1, TU_microsecond = 3, TU_millisecond = 4, TU_second = 5, TU_minute = 6, TU_hour = 7,
    TU_day = 8, TU_week = 9
00147     double res = 1.0;
00148     int intTimeUnit1, intTimeUnit2;
00149     intTimeUnit1 = static_cast<int> (timeUnit1);
00150     intTimeUnit2 = static_cast<int> (timeUnit2);
00151     if (intTimeUnit1 <= intTimeUnit2) {
00152         for (int i = intTimeUnit1 + 1; i <= intTimeUnit2; i++) {
00153             res /= scaleValues[i];

```

```
00154      }
00155  } else {
00156    for (int i = intTimeUnit2 + 1; i <= intTimeUnit1; i++) {
00157      res *= scaleValues[i];
00158    }
00159  }
00160  return res;
00161 }
```

7.539 Util.h File Reference

```
#include <map>
#include <typeinfo>
#include <string>
#include <cctype>
#include <algorithm>
#include <locale>
```

Classes

- class [Util](#)

Functions

- static void [ltrim](#) (std::string &s)
- static void [rtrim](#) (std::string &s)
- static void [trim](#) (std::string &s)
- static void [trimwithin](#) (std::string &str)
- static std::string [getFileName](#) (const std::string &s)

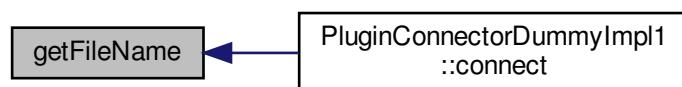
7.539.1 Function Documentation

7.539.1.1 [getFileName\(\)](#)

```
static std::string getFileName (
    const std::string & s ) [inline], [static]
```

Definition at line 53 of file [Util.h](#).

Here is the caller graph for this function:



7.539.1.2 ltrim()

```
static void ltrim (
    std::string & s ) [inline], [static]
```

Definition at line 26 of file [Util.h](#).

Here is the caller graph for this function:



7.539.1.3 rtrim()

```
static void rtrim (
    std::string & s ) [inline], [static]
```

Definition at line 33 of file [Util.h](#).

Here is the caller graph for this function:

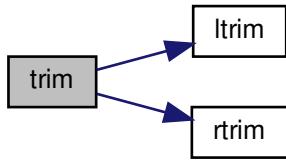


7.539.1.4 trim()

```
static void trim (
    std::string & s ) [inline], [static]
```

Definition at line 40 of file [Util.h](#).

Here is the call graph for this function:



Here is the caller graph for this function:

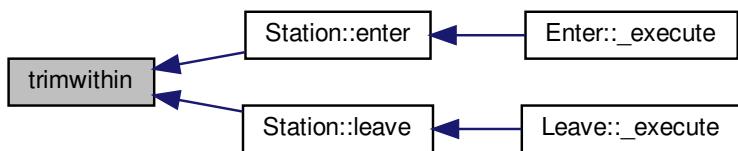


7.539.1.5 trimwithin()

```
static void trimwithin (
    std::string & str ) [inline], [static]
```

Definition at line 46 of file [Util.h](#).

Here is the caller graph for this function:



7.540 Util.h

```

00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Util.h
00009 * Author: rafael.luiz.cancian
00010 *
00011 * Created on 21 de Junho de 2018, 13:02
00012 */
00013
00014 #ifndef UTIL_H
00015 #define UTIL_H
00016
00017 #include <map>
00018 #include <typeinfo>
00019 #include <string>
00020 #include <cctype>
00021 #include <algorithm>
00022 #include <cctype>
00023 #include <locale>
00024
00025 // trim from start (in place)
00026 static inline void ltrim(std::string &s) {
00027     s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](int ch) {
00028         return !std::isspace(ch);
00029     }));
00030 }
00031
00032 // trim from end (in place)
00033 static inline void rtrim(std::string &s) {
00034     s.erase(std::find_if(s.rbegin(), s.rend(), [](int ch) {
00035         return !std::isspace(ch);
00036     }).base(), s.end());
00037 }
00038
00039 // trim from both ends (in place)
00040 static inline void trim(std::string &s) {
00041     ltrim(s);
00042     rtrim(s);
00043 }
00044
00045 // trim all spaces within the string (in place) -- used to transform general names into valid literals
00046 static inline void trimwithin(std::string &str) {
00047     //ltrim(s);
00048     //rtrim(s);
00049     //s.erase(std::remove_if(s.begin(), s.end(), std::isspace), s.end());
00050     str.erase(remove(str.begin(), str.end(), ' '), str.end());
00051 }
00052
00053 static inline std::string getFileName(const std::string& s) {
00054     char sep = '/';
00055     size_t i = s.rfind(sep, s.length());
00056     if (i != std::string::npos) {
00057         return(s.substr(i+1, s.length() - i));
00058     }
00059     return s;
00060 }
00061
00062
00063 class Util {
00064 public:
00065     typedef unsigned long identification;
00066     typedef unsigned int rank;
00067
00068     enum class TimeUnit : int {
00069         picosecond = 1,
00070         nanosecond = 2,
00071         microsecond = 3,
00072         millisecond = 4,
00073         second = 5,
00074         minute = 6,
00075         hour = 7,
00076         day = 8,
00077         week = 9
00078     };
00079     //static std::string ToStrTimeUnit(TimeUnit tu);
00080
00081     enum class TraceLevel : int {
00082         noTraces = 0,
00083         errorFatal = 1,
00084         errorRecover = 2,

```

```

00085     warning = 3,
00086     report = 4,
00087     simulatorResult = 5,
00088     toolResult = 6,
00089     modelResult = 7,
00090     componentResult = 8,
00091     elementResult = 9,
00092     modelSimulationEvent = 10,
00093     componentArrival = 11,
00094     simulatorInternal = 12,
00095     toolInternal = 13,
00096     modelSimulationInternal = 14,
00097     modelInternal = 14,
00098     componentInternal = 15,
00099     elementInternal = 16,
00100    simulatorDetailed = 17,
00101    toolDetailed = 17,
00102    modelSimulationDetailed = 18,
00103    modelDetailed = 19,
00104    componentDetailed = 20,
00105    elementDetailed = 21,
00106    everythingMostDetailed = 30
00107  };
00108 private:
00109     static Util::identification _S_lastId;
00110     static std::map<std::string, Util::identification> _S_lastIdOfType;
00111     static std::map<std::string, std::string> _S_TypeOf;
00112
00113 public: // indentation and string
00114     static unsigned int _S_indentation; // \todo: IS PRIVATE. ITS HERE JUST TO INCLUDE IT AS
00115     A WATCH
00116     static void SetIndent(const unsigned short indent);
00117     static void IncIndent();
00118     static void DecIndent();
00119     static std::string Indent();
00120     static std::string SetW(std::string text, unsigned short width);
00121     static std::string StrTimeUnit(Util::TimeUnit timeUnit);
00122 public: // identification
00123     static Util::identification GenerateNewId();
00124     static Util::identification GenerateNewIdOfType(std::string
00125         objtype);
00126     static Util::identification GetLastIdOfType(std::string objtype);
00127     static void ResetIdOfType(std::string objtype);
00128 public: // simulation support
00129     static double TimeUnitConvert(Util::TimeUnit timeUnit1,
00130                                     Util::TimeUnit timeUnit2);
00131 public: // template implementations
00132
00133     template<class T> static std::string TypeOf() {
00134         std::string name = typeid(T).name();
00135         std::map<std::string, std::string>::iterator it = _S_TypeOf.find(name);
00136         if (it != _S_TypeOf.end()) {
00137             return(*it).second;
00138         } else {
00139             std::string newname(name);
00140             while (std::isdigit(newname[0])) {
00141                 newname.erase(0, 1);
00142             }
00143             _S_TypeOf.insert(std::pair<std::string, std::string>(name, newname));
00144             return newname;
00145         }
00146     }
00147
00148     template<class T> static Util::identification
00149     GenerateNewIdOfType() {
00150         std::string objtype = Util::TypeOf<T>();
00151         std::map<std::string, Util::identification>::iterator it = Util::_S_lastIdOfType.find(objtype);
00152         if (it == Util::_S_lastIdOfType.end()) {
00153             // a new one. create the pair
00154             Util::_S_lastIdOfType.insert(std::pair<std::string, Util::identification>(objtype, 0));
00155             it = Util::_S_lastIdOfType.find(objtype);
00156         }
00157         Util::identification next = (*it).second;
00158         next++;
00159         (*it).second = next;
00160         return(*it).second;
00161     }
00162
00163     private:
00164     Util();
00165     virtual ~Util() = default;
00166 }
00167
00168 #endif /* UTIL_H */

```

00174

7.541 Variable.cpp File Reference

```
#include "Variable.h"
#include "Plugin.h"
```

7.542 Variable.cpp

```
00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Variable.cpp
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 4 de Setembro de 2018, 18:28
00012 */
00013
00014 #include "Variable.h"
00015 #include "Plugin.h"
00016
00017 Variable::Variable(Model* model, std::string name) :
00018     ModelElement(model, Util::TypeOf<Variable>(), name) {
00019     _name = name;
00020 }
00021 std::string Variable::show() {
00022     return ModelElement::show(); // \todo: include values
00023 }
00024
00025 PluginInformation* Variable::GetPluginInformation() {
00026     PluginInformation* info = new PluginInformation(
00027         Util::TypeOf<Variable>(), &Variable::LoadInstance);
00028     return info;
00029 }
00030 double Variable::value() {
00031     return value("");
00032 }
00033
00034 double Variable::value(std::string index) {
00035     std::map<std::string, double>::iterator it = _values->find(index);
00036     if (it == _values->end()) {
00037         return 0.0; // index does not exist. Assuming sparse matrix, it's zero.
00038     } else {
00039         return it->second;
00040     }
00041 }
00042
00043 void Variable::setValue(double value) {
00044     setValue("", value);
00045 }
00046
00047 void Variable::setValue(std::string index, double value) {
00048     std::map<std::string, double>::iterator it = _values->find(index);
00049     if (it == _values->end()) {
00050         // index does not exist. Create it.
00051         _values->insert({index, value}); // (std::pair<std::string, double>(index, value));
00052     } else {
00053         it->second = value;
00054     }
00055 }
00056
00057 double Variable::initialValue() {
00058     return initialValue("");
00059 }
00060
00061 void Variable::setInitialValue(double value) {
00062     setValue("", value);
00063 }
00064
00065 double Variable::initialValue(std::string index) {
```

```

00066     std::map<std::string, double>::iterator it = _initialValues->find(index);
00067     if (it == _initialValues->end()) {
00068         return 0.0; // index does not exist. Assuming sparse matrix, it's zero.
00069     } else {
00070         return it->second;
00071     }
00072 }
00073
00074 void Variable::setInitialValue(std::string index, double
00075     value) {
00076     std::map<std::string, double>::iterator it = _initialValues->find(index);
00077     if (it == _initialValues->end()) {
00078         // index does not exist. Create it.
00079         _initialValues->insert(std::pair<std::string, double>(index, value));
00080     } else {
00081         it->second = value;
00082     }
00083
00084 List<unsigned int>* Variable::dimensionSizes() const {
00085     return _dimensionSizes;
00086 }
00087
00088 ModelElement* Variable::LoadInstance(Model* model,
00089     std::map<std::string, std::string>* fields) {
00090     Variable* newElement = new Variable(model);
00091     try {
00092         newElement->_loadInstance(fields);
00093     } catch (const std::exception& e) {
00094     }
00095     return newElement;
00096 }
00097
00098 bool Variable::_loadInstance(std::map<std::string, std::string>* fields) {
00099     bool res = ModelElement::_loadInstance(fields);
00100     if (res) {
00101         unsigned int nv = std::stoi((*(fields->find("numValues"))).second);
00102         std::string pos;
00103         double value;
00104         for (unsigned int i = 0; i < nv; i++) {
00105             pos = ((*(fields->find("pos" + std::to_string(i)))).second);
00106             value = std::stod((*(fields->find("value" + std::to_string(i)))).second);
00107             this->_initialValues->emplace(pos, value);
00108         }
00109     }
00110     return res;
00111 }
00112
00113 }
00114
00115 std::map<std::string, std::string>* Variable::_saveInstance() {
00116     std::map<std::string, std::string>* fields = ModelElement::_saveInstance();
00117     //Util::TypeOf<Variable>());
00118     fields->emplace("numValues", std::to_string(this->_initialValues->size()));
00119     unsigned int i = 0;
00120     for (std::map<std::string, double>::iterator it = this->_initialValues->begin(); it != _initialValues->
00121         end(); it++) {
00122         fields->emplace("pos" + std::to_string(i), (*it).first);
00123         fields->emplace("value" + std::to_string(i), std::to_string((*it).second));
00124         i++;
00125     }
00126     return fields;
00127 }
00128
00129 bool Variable::_check(std::string* errorMessage) {
00130     return true;
00131 }
00132
00133 void Variable::_initBetweenReplications() {
00134     this->_values->clear();
00135     this->_values = this->_initialValues;
00136 }

```

7.543 Variable.h File Reference

```

#include "ModelElement.h"
#include "ElementManager.h"
#include "Plugin.h"

```

Classes

- class Variable

7.544 Variable.h

```

00001 /*
00002  * To change this license header, choose License Headers in Project Properties.
00003  * To change this template file, choose Tools | Templates
00004  * and open the template in the editor.
00005 */
00006
00007 /*
00008  * File:  Variable.h
00009  * Author: rafael.luiz.cancian
00010 *
00011  * Created on 4 de Setembro de 2018, 18:28
00012 */
00013
00014 #ifndef VARIABLE_H
00015 #define VARIABLE_H
00016
00017 #include "ModelElement.h"
00018 #include "ElementManager.h"
00019 #include "Plugin.h"
00020
00090 class Variable : public ModelElement {
00091 public:
00092     Variable(Model* model, std::string name = "");
00093     virtual ~Variable() = default;
00094 public:
00095     virtual std::string show();
00096 public: //static
00097     static PluginInformation* GetPluginInformation();
00098     static ModelElement* LoadInstance(Model* model, std::map<std::string,
00099                                         std::string>* fields);
00099 public:
00100     double value();
00101     void setValue(double value);
00102     double value(std::string index);
00103     void setValue(std::string index, double value);
00104     double initialValue();
00105     void setInitialValue(double value);
00106     double initialValue(std::string index);
00107     void setInitialValue(std::string index, double value);
00108     List<unsigned int>* dimensionSizes() const;
00109
00110 protected:
00111     virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00112     virtual std::map<std::string, std::string>* _saveInstance();
00113     virtual bool _check(std::string* errorMessage);
00114     virtual void _initBetweenReplications();
00115
00116 private:
00117     List<unsigned int>* _dimensionSizes = new
00118         List<unsigned int>();
00118     std::map<std::string, double>* _values = new std::map<std::string, double>();
00119     std::map<std::string, double>* _initialValues = new std::map<std::string, double>();
00120 };
00121
00122 #endif /* VARIABLE_H */
00123

```

7.545 Write.cpp File Reference

```

#include "Write.h"
#include "Model.h"
#include <fstream>

```

7.546 Write.cpp

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Write.cpp
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:06
00012 */
00013
00014 #include "Write.h"
00015 #include "Model.h"
00016
00017 #include <fstream>
00018
00019 Write::Write(Model* model, std::string name) : ModelComponent(model,
00020     Util::TypeOf<Write>(), name) {
00021 }
00022 std::string Write::show() {
00023     return ModelComponent::show() + "";
00024 }
00025
00026 ModelComponent* Write::LoadInstance(Model* model,
00027     std::map<std::string, std::string>* fields) {
00028     Write* newComponent = new Write(model);
00029     try {
00030         newComponent->_loadInstance(fields);
00031     } catch (const std::exception& e) {
00032     }
00033     return newComponent;
00034 }
00035
00036 List<WriteElement*>* Write::writeElements() const {
00037     return _writeElements;
00038 }
00039
00040 void Write::setFilename(std::string _filename) {
00041     this->_filename = _filename;
00042 }
00043
00044 std::string Write::filename() const {
00045     return _filename;
00046 }
00047
00048 void Write::setWriteToType(WriteToType _writeToType) {
00049     this->_writeToType = _writeToType;
00050 }
00051
00052 Write::WriteToType Write::writeToType() const {
00053     return _writeToType;
00054 }
00055
00056 void Write::_execute(Entity* entity) {
00057     WriteElement* msgElem;
00058     std::list<WriteElement*>* msgs = this->_writeElements->list();
00059     std::string message = "";
00060     for (std::list<WriteElement*>::iterator it = msgs->begin(); it != msgs->end(); it++) {
00061         msgElem = (*it);
00062         if (msgElem->isExpression) {
00063             message += std::to_string(_parentModel->parseExpression(msgElem->
00064                 text));
00065         } else {
00066             message += msgElem->text;
00067         }
00068         if (msgElem->newline) {
00069             if (this->_writeToType == Write::WriteToType::SCREEN) { //\todo: Write To
00070                 FILE not implemented
00071                 _parentModel->tracer()->trace(
00072                     Util::TraceLevel::report, message);
00073             } else if (this->_writeToType == Write::WriteToType::FILE) {
00074                 // open file
00075                 std::ofstream savefile;
00076                 savefile.open(_filename, std::ofstream::app);
00077                 savefile << message << std::endl;
00078                 savefile.close();
00079             }
00080             message = "";
00081         }
00082     }
00083 }
```

```

00080
00081     this->_parentModel->sendEntityToComponent(entity, this->
00082         nextComponents()->frontConnection(), 0.0);
00083
00084     bool Write::_loadInstance(std::map<std::string, std::string>* fields) {
00085         bool res = ModelComponent::_loadInstance(fields);
00086         if (res) {
00087             //...
00088         }
00089         return res;
00090     }
00091
00092     void Write::_initBetweenReplications() {
00093         try {
00094             std::ofstream savefile;
00095             savefile.open(_filename, std::ofstream::app);
00096             savefile << "# Replication number " << _parentModel->simulation()->
00097                 currentReplicationNumber() << "/" << _parentModel->
00098                     infos()->numberOfReplications() << std::endl;
00099             savefile.close();
00100         } catch (...) {
00101     }
00102
00103     std::map<std::string, std::string>* Write::_saveInstance() {
00104         std::map<std::string, std::string>* fields = ModelComponent::_saveInstance
00105             ();
00106         //...
00107         return fields;
00108     }
00109     bool Write::_check(std::string* errorMessage) {
00110         bool resultAll = true;
00111         WriteElement* msgElem;
00112         unsigned short i = 0;
00113         std::list<WriteElement*>* msgs = this->_writeElements->list();
00114         for (std::list<WriteElement*>::iterator it = msgs->begin(); it != msgs->end(); it++) {
00115             msgElem = (*it);
00116             i++;
00117             if (msgElem->isExpression) {
00118                 resultAll &= _parentModel->checkExpression(msgElem->
00119                     text, "writeExpression"+std::to_string(i), errorMessage);
00120             }
00121             // when cheking the model (before simulating it), remove the file if exists
00122             std::remove(_filename.c_str());
00123         }
00124         return resultAll;
00125     }
00126     PluginInformation* Write::GetPluginInformation() {
00127         PluginInformation* info = new PluginInformation(Util::TypeOf<Write>()
00128             , &Write::LoadInstance);
00129         // ...
00130         return info;
00131     }
00132

```

7.547 Write.h File Reference

```
#include "ModelComponent.h"
```

Classes

- class [WriteElement](#)
- class [Write](#)

7.548 Write.h

```
00001 /*
00002 * To change this license header, choose License Headers in Project Properties.
00003 * To change this template file, choose Tools | Templates
00004 * and open the template in the editor.
00005 */
00006
00007 /*
00008 * File: Write.h
00009 * Author: rlcancian
00010 *
00011 * Created on 11 de Setembro de 2019, 13:06
00012 */
00013
00014 #ifndef WRITE_H
00015 #define WRITE_H
00016
00017 #include "ModelComponent.h"
00018
00019 class WriteElement {
00020     public:
00021
00022     WriteElement(std::string text, bool isExpression=false, bool
00023         newline=false) {
00023         this->text = text;
00024         this->isExpression = isExpression;
00025         this->newline = newline;
00026     }
00027     std::string text;
00028     bool isExpression;
00029     bool newline;
00030 };
00031
00032 class Write : public ModelComponent {
00033     public:
00034
00035     enum class WriteToType : int {
00036         SCREEN = 1, FILE = 2
00037     };
00038
00039     public: // constructors
00040         Write(Model* model, std::string name="");
00041         virtual ~Write() = default;
00042
00043     public: // virtual
00044         virtual std::string show();
00045
00046     public: // static
00047         static PluginInformation* GetPluginInformation();
00048         static ModelComponent* LoadInstance(Model* model, std::map<std::string, std::string>
00049             * fields);
00050
00051     public:
00052         List<WriteElement*>* writeElements() const;
00053         void setFilename(std::string _filename);
00054         std::string filename() const;
00055         void setWriteToType(WriteToType _writeToType);
00056         Write::WriteToType writeToType() const;
00057
00058     protected: // virtual
00059         virtual void _execute(Entity* entity);
00060         virtual void _initBetweenReplications();
00061         virtual bool _loadInstance(std::map<std::string, std::string>* fields);
00062         virtual std::map<std::string, std::string>* _saveInstance();
00063         virtual bool _check(std::string* errorMessage);
00064
00065     private: // methods
00066         //std::string _buildText();
00067     private: // attributes 1:1
00068         WriteToType _writeToType = Write::WriteToType::SCREEN;
00069         std::string _filename = "";
00070
00071     private: // attributes 1:n
00072         List<WriteElement*>* _writeElements = new
00073             List<WriteElement*>();
00074
00075 #endif /* WRITE_H */
00076
```

