

Laboratory for Advanced Software Systems  
University of Luxembourg

Group 8



*iCrash* :  
A Crisis Management Case Study  
**MESSIR** Analysis Document  
- v 1.4 -  
(Report type: Default)

Wednesday 30<sup>th</sup> November, 2016 - 16:53

# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
1.1	Overview	13
1.2	Purpose and recipients of the document	13
1.3	Application Domain	13
1.4	Definitions, acronyms and abbreviations	13
1.5	Document structure	14
<b>2</b>	<b>General Description</b>	<b>15</b>
2.1	Domain Stakeholders	15
2.1.1	Communication Company	15
2.1.2	Humans	16
2.1.3	Coordinators	16
2.1.4	Police	16
2.1.5	Administrator	17
2.1.6	Creator	17
2.1.7	Activator	17
2.2	System's Actors	19
2.3	Use Cases Model	19
2.3.1	Use Cases	19
2.3.2	Use Case Instance(s)	31
<b>3</b>	<b>Environment Model</b>	<b>35</b>
3.1	Local view 01	35
3.2	Local view 02	35
3.3	Local view 03	35
3.4	Local view 04	36
3.5	Local view 05	36
3.6	Local view 08	36
3.7	Global view 01	36
3.8	Actors and Interfaces Descriptions	39
3.8.1	actActivator Actor	39
3.8.2	actAdministrator Actor	39
3.8.3	actAuthenticated Actor	40
3.8.4	actComCompany Actor	40
3.8.5	actCoordinator Actor	40
3.8.6	actMsrCreator Actor	41
3.8.7	actPolice Actor	41
<b>4</b>	<b>Concept Model</b>	<b>43</b>
4.1	PrimaryTypes-Classes	43

4.1.1	Local view 01 . . . . .	43
4.1.2	Local view 02 . . . . .	43
4.1.3	Local view 03 . . . . .	43
4.1.4	Local view 04 . . . . .	43
4.1.5	Global view 01 . . . . .	43
4.2	PrimaryTypes-Datatypes . . . . .	43
4.2.1	Local view 06 . . . . .	43
4.2.2	Global view 01 . . . . .	47
4.3	SecondaryTypes-Datatypes . . . . .	47
4.3.1	Local view 01 . . . . .	47
4.4	Concept Model Types Descriptions . . . . .	47
4.4.1	Primary types - Class types descriptions . . . . .	47
4.4.2	Primary types - Datatypes types descriptions . . . . .	50
4.4.3	Primary types - Association types descriptions . . . . .	51
4.4.4	Primary types - Aggregation types descriptions . . . . .	53
4.4.5	Secondary types - Class types descriptions . . . . .	53
4.4.6	Secondary types - Datatypes types descriptions . . . . .	53
4.4.7	Secondary types - Association types descriptions . . . . .	53
4.4.8	Secondary types - Aggregation types descriptions . . . . .	54
4.4.9	Secondary types - Composition types descriptions . . . . .	54
<b>5</b>	<b>Operation Model . . . . .</b>	<b>55</b>
5.1	Environment - Out Interface Operation Scheme for actActivator . . . . .	55
5.1.1	Operation Model for oeSetClock . . . . .	55
5.1.2	Operation Model for oeSollicitateCrisisHandling . . . . .	57
5.2	Environment - Out Interface Operation Scheme for actAdministrator . . . . .	60
5.2.1	Operation Model for oeAddCoordinator . . . . .	60
5.2.2	Operation Model for oeAddPolice . . . . .	64
5.2.3	Operation Model for oeDeleteCoordinator . . . . .	66
5.2.4	Operation Model for oeDeletePolice . . . . .	69
5.3	Environment - Out Interface Operation Scheme for actAuthenticated . . . . .	70
5.3.1	Operation Model for oeLogin . . . . .	70
5.3.2	Operation Model for oeLogout . . . . .	73
5.4	Environment - Out Interface Operation Scheme for actComCompany . . . . .	75
5.4.1	Operation Model for oeAlert . . . . .	75
5.5	Environment - Out Interface Operation Scheme for actCoordinator . . . . .	83
5.5.1	Operation Model for oeCloseCrisis . . . . .	83
5.5.2	Operation Model for oeGetAlertsSet . . . . .	86
5.5.3	Operation Model for oeGetCrisisSet . . . . .	87
5.5.4	Operation Model for oeInvalidateAlert . . . . .	89
5.5.5	Operation Model for oeReportOnCrisis . . . . .	91
5.5.6	Operation Model for oeSetCrisisHandler . . . . .	93
5.5.7	Operation Model for oeSetCrisisStatus . . . . .	96
5.5.8	Operation Model for oeSetCrisisType . . . . .	98
5.5.9	Operation Model for oeValidateAlert . . . . .	100
5.6	Environment - Out Interface Operation Scheme for actMsrCreator . . . . .	102
5.6.1	Operation Model for oeCreateSystemAndEnvironment . . . . .	102
5.7	Environment - Out Interface Operation Scheme for actPolice . . . . .	106
5.7.1	Operation Model for oeCloseCrisis . . . . .	106

5.7.2	Operation Model for oeGetCrisisSet . . . . .	107
5.7.3	Operation Model for oeReportOnCrisis . . . . .	107
5.7.4	Operation Model for oeSetCrisisHandler . . . . .	108
5.7.5	Operation Model for oeSetCrisisStatus . . . . .	108
5.8	Environment - Actor Operation Scheme for actMsrCreator . . . . .	109
5.8.1	Operation Model for init . . . . .	109
5.9	Primary Types - Operation Schemes for Class ctAdministrator . . . . .	109
5.9.1	Operation Model for init . . . . .	109
5.10	Primary Types - Operation Schemes for Class ctAlert . . . . .	112
5.10.1	Operation Model for init . . . . .	112
5.10.2	Operation Model for isSentToCoordinator . . . . .	113
5.11	Primary Types - Operation Schemes for Class ctAuthenticated . . . . .	114
5.11.1	Operation Model for init . . . . .	114
5.12	Primary Types - Operation Schemes for Class ctCoordinator . . . . .	115
5.12.1	Operation Model for init . . . . .	115
5.13	Primary Types - Operation Schemes for Class ctCrisis . . . . .	116
5.13.1	Operation Model for init . . . . .	116
5.13.2	Operation Model for handlingDelayPassed . . . . .	118
5.13.3	Operation Model for maxHandlingDelayPassed . . . . .	119
5.13.4	Operation Model for isSentToCoordinator . . . . .	121
5.13.5	Operation Model for isSentToPolice . . . . .	122
5.13.6	Operation Model for isAllocatedIfPossible . . . . .	122
5.14	Primary Types - Operation Schemes for Class ctHuman . . . . .	124
5.14.1	Operation Model for init . . . . .	124
5.14.2	Operation Model for isAcknowledged . . . . .	126
5.15	Primary Types - Operation Schemes for Class ctPolice . . . . .	126
5.15.1	Operation Model for init . . . . .	126
5.16	Primary Types - Operation Schemes for Class ctState . . . . .	127
5.16.1	Operation Model for init . . . . .	127
5.17	Primary Types - Operation Schemes for Datatype dtAlertID . . . . .	129
5.17.1	Operation Model for is . . . . .	129
5.18	Primary Types - Operation Schemes for Datatype dtComment . . . . .	130
5.18.1	Operation Model for is . . . . .	130
5.19	Primary Types - Operation Schemes for Datatype dtCoordinatorID . . . . .	132
5.19.1	Operation Model for is . . . . .	132
5.20	Primary Types - Operation Schemes for Datatype dtCrisisID . . . . .	133
5.20.1	Operation Model for is . . . . .	133
5.21	Primary Types - Operation Schemes for Datatype dtGPSLocation . . . . .	134
5.21.1	Operation Model for is . . . . .	134
5.21.2	Operation Model for isNearTo . . . . .	135
5.22	Primary Types - Operation Schemes for Datatype dtLatitude . . . . .	137
5.22.1	Operation Model for is . . . . .	137
5.23	Primary Types - Operation Schemes for Datatype dtLogin . . . . .	138
5.23.1	Operation Model for is . . . . .	138
5.24	Primary Types - Operation Schemes for Datatype dtLongitude . . . . .	139
5.24.1	Operation Model for is . . . . .	139
5.25	Primary Types - Operation Schemes for Datatype dtPassword . . . . .	140
5.25.1	Operation Model for is . . . . .	140
5.26	Primary Types - Operation Schemes for Datatype dtPhoneNumber . . . . .	142

<b>CONTENTS</b>	<b>5</b>
-----------------	----------

5.26.1    Operation Model for is . . . . .	142
5.27    Primary Types - Operation Schemes for Datatype dtPoliceID . . . . .	143
5.27.1    Operation Model for is . . . . .	143
5.28    Primary Types - Operation Schemes for Enumeration etAlertStatus . . . . .	143
5.28.1    Operation Model for is . . . . .	143
5.29    Primary Types - Operation Schemes for Enumeration etCrisisStatus . . . . .	145
5.29.1    Operation Model for is . . . . .	145
5.30    Primary Types - Operation Schemes for Enumeration etCrisisType . . . . .	146
5.30.1    Operation Model for is . . . . .	146
5.31    Primary Types - Operation Schemes for Enumeration etHumanKind . . . . .	147
5.31.1    Operation Model for is . . . . .	147
5.32    Secondary Types - Operation Schemes for Classes . . . . .	148
5.33    Secondary Types - Operation Schemes for Datatype dtSMS . . . . .	148
5.33.1    Operation Model for is . . . . .	148
5.34    Secondary Types - Operation Schemes for Enumerations . . . . .	149
<b>6    Test Model(s) . . . . .</b>	<b>151</b>
6.1    Test Model for testcase01 . . . . .	151
6.1.1    Test Steps Specification . . . . .	151
6.1.2    Test Case Instance - instance01 . . . . .	173
6.1.3    Test Case Instance - instance01Part01 . . . . .	174
6.1.4    Test Case Instance - instance01Part02 . . . . .	174
<b>7    Additional Constraints . . . . .</b>	<b>177</b>
7.1    Quality Constraints . . . . .	177
7.1.1    Functional suitability . . . . .	177
7.1.2    Performance efficiency . . . . .	177
7.1.3    Compatibility . . . . .	178
7.1.4    Usability . . . . .	178
7.1.5    Reliability . . . . .	179
7.1.6    Security . . . . .	180
7.1.7    Maintainability . . . . .	180
7.1.8    Portability . . . . .	181
7.2    Other Constraints . . . . .	182
<b>A    Specification project lu.uni.lassy.excalibur.examples.icrash . . . . .</b>	<b>183</b>
A.1    Use Cases Model . . . . .	184
A.1.1    Use Cases . . . . .	184
<b>B    Messir Specification Files Listing . . . . .</b>	<b>185</b>
B.1    File /src-gen/messir-spec/.views.msr . . . . .	185
B.2    File /src-gen/messir-spec/operations/concepts/secondarytypes-datatYPES/dtSMS.msr	185
B.3    File /src-gen/messir-spec/operations.../environment-actActivator-oeSetClock.msr .	186
B.4    File /src-gen.../environment-actActivator-oeSollicitateCrisisHandling.msr . . . . .	186
B.5    File /src-gen/messir-spec.../environment-actAdministrator-oeAddCoordinator.msr .	187
B.6    File /src-gen/messir-spec.../environment-actAdministrator-oeAddPolice.msr . . . . .	188
B.7    File /src-gen.../environment-actAdministrator-oeDeleteCoordinator.msr . . . . .	189
B.8    File /src-gen/messir-spec.../environment-actAdministrator-oeDeletePolice.msr . . . . .	190
B.9    File /src-gen/messir-spec/operations.../environment-actAuthenticated.msr . . . . .	191
B.10   File /src-gen/messir-spec/operations/environment/environment-actComCompany.msr	193

B.11	File /src-gen/messir-spec.../environment-actCoordinator-oeCloseCrisis.msr . . . . .	195
B.12	File /src-gen/messir-spec.../environment-actCoordinator-oeGetAlertsSet.msr . . . . .	196
B.13	File /src-gen/messir-spec.../environment-actCoordinator-oeGetCrisisSet.msr . . . . .	196
B.14	File /src-gen/messir-spec.../environment-actCoordinator-oeInvalidateAlert.msr . . . . .	196
B.15	File /src-gen/messir-spec.../environment-actCoordinator-oeReportOnCrisis.msr . . . . .	197
B.16	File /src-gen/messir-spec.../environment-actCoordinator-oeSetCrisisHandler.msr . . . . .	197
B.17	File /src-gen/messir-spec.../environment-actCoordinator-oeSetCrisisStatus.msr . . . . .	197
B.18	File /src-gen/messir-spec.../environment-actCoordinator-oeSetCrisisType.msr . . . . .	198
B.19	File /src-gen/messir-spec.../environment-actCoordinator-oeValidateAlert.msr . . . . .	198
B.20	File /src-gen/messir-spec/operations.../environment-actMsrCreator-init.msr . . . . .	199
B.21	File /src-gen.../environment-actMsrCreator-oeCreateSystemAndEnvironment.msr . . . . .	199
B.22	File /src-gen/messir-spec/operations.../environment-actPolice-oeCloseCrisis.msr . . . . .	200
B.23	File /src-gen/messir-spec.../environment-actPolice-oeGetCrisisSet.msr . . . . .	200
B.24	File /src-gen/messir-spec.../environment-actPolice-oeReportOnCrisis.msr . . . . .	201
B.25	File /src-gen/messir-spec.../environment-actPolice-oeSetCrisisHandler.msr . . . . .	201
B.26	File /src-gen/messir-spec.../environment-actPolice-oeSetCrisisStatus.msr . . . . .	201
B.27	File /src-gen/messir-spec/environment/environment.msr . . . . .	202
B.28	File /src-gen/messir-spec/concepts/primarytypes-associations.msr . . . . .	204
B.29	File /src-gen/messir-spec.../primarytypes-classes-ctAdministrator.msr . . . . .	205
B.30	File /src-gen/messir-spec/operations.../primarytypes-classes-ctAlert.msr . . . . .	206
B.31	File /src-gen/messir-spec.../primarytypes-classes-ctAuthenticated.msr . . . . .	207
B.32	File /src-gen/messir-spec/operations.../primarytypes-classes-ctCoordinator.msr . . . . .	207
B.33	File /src-gen/messir-spec/operations.../primarytypes-classes-ctCrisis.msr . . . . .	207
B.34	File /src-gen/messir-spec/operations.../primarytypes-classes-ctHuman.msr . . . . .	210
B.35	File /src-gen/messir-spec/operations.../primarytypes-classes-ctPolice.msr . . . . .	210
B.36	File /src-gen/messir-spec/operations.../primarytypes-classes-ctState.msr . . . . .	211
B.37	File /src-gen/messir-spec/concepts/primarytypes-classes.msr . . . . .	212
B.38	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtAlertID.msr . . . . .	214
B.39	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtComment.msr . . . . .	214
B.40	File /src-gen/messir-spec.../primarytypes-datatype-dtCoordinatorID.msr . . . . .	215
B.41	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtCrisisID.msr . . . . .	215
B.42	File /src-gen/messir-spec.../primarytypes-datatype-dtGPSLocation.msr . . . . .	215
B.43	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtLogin.msr . . . . .	217
B.44	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtPassword.msr . . . . .	217
B.45	File /src-gen/messir-spec.../primarytypes-datatype-dtPhoneNumber.msr . . . . .	218
B.46	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtPoliceID.msr . . . . .	218
B.47	File /src-gen/messir-spec.../primarytypes-datatype-dtAlertStatus.msr . . . . .	219
B.48	File /src-gen/messir-spec.../primarytypes-datatype-dtCrisisStatus.msr . . . . .	219
B.49	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtCrisisType.msr . . . . .	220
B.50	File /src-gen/messir-spec/operations.../primarytypes-datatype-dtHumanKind.msr . . . . .	220
B.51	File /src-gen/messir-spec/concepts/primarytypes-datatype.msr . . . . .	221
B.52	File /src-gen/messir-spec/concepts/secondarytypes-associations.msr . . . . .	222
B.53	File /src-gen/messir-spec/concepts/secondarytypes-classes.msr . . . . .	222
B.54	File /src-gen/messir-spec/concepts/secondarytypes-datatype.msr . . . . .	222
B.55	File /src-gen/messir-spec/usecases/subfunctions-usecases.msr . . . . .	222
B.56	File /src-gen/messir-spec/test/tc-testcase01.msr . . . . .	225
B.57	File /src-gen/messir-spec/test/tci-testcase01-instance01.msr . . . . .	233
B.58	File /src-gen/messir-spec/usecases/usecase-suDeployAndRun.msr . . . . .	243
B.59	File /src-gen/messir-spec/usecases/usecase-suGlobalCrisisHandling.msr . . . . .	249

B.60	File /src-gen/messir-spec/usecases/usecase-ugAdministrateTheSystem.msr . . . . .	250
B.61	File /src-gen/messir-spec/usecases/usecase-ugManageCrisis.msr . . . . .	251
B.62	File /src-gen/messir-spec/usecases/usecase-ugMonitor.msr . . . . .	251
B.63	File /src-gen/messir-spec/usecases/usecase-ugSecurelyUseSystem.msr . . . . .	252
B.64	File /src-gen.../usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr . . . . .	252
<b>C</b>	<b>Listing of the Prolog Files Referenced in the Operation Model Specification . . . . .</b>	<b>255</b>
C.1	File /src-gen/prolog-ref-spec/Operations.../outactActivator-oeSetClock.pl . . . . .	255
C.2	File /src-gen/prolog-ref-spec.../outactActivator-oeSollicitateCrisisHandling.pl . . . . .	256
C.3	File /src-gen/prolog-ref-spec.../outactAdministrator-oeAddCoordinator.pl . . . . .	258
C.4	File /src-gen/prolog-ref-spec.../outactAdministrator-oeDeleteCoordinator.pl . . . . .	259
C.5	File /src-gen/prolog-ref-spec/Operations.../outactAuthenticated-oeLogin.pl . . . . .	260
C.6	File /src-gen/prolog-ref-spec/Operations.../outactAuthenticated-oeLogout.pl . . . . .	262
C.7	File /src-gen/prolog-ref-spec/Operations.../outactComCompany-oeAlert.pl . . . . .	263
C.8	File /src-gen/prolog-ref-spec/Operations.../outactCoordinator-oeCloseCrisis.pl . . . . .	267
C.9	File /src-gen/prolog-ref-spec/Operations.../outactCoordinator-oeGetAlertsSet.pl . . . . .	268
C.10	File /src-gen/prolog-ref-spec/Operations.../outactCoordinator-oeGetCrisisSet.pl . . . . .	269
C.11	File /src-gen/prolog-ref-spec.../outactCoordinator-oeInvalidateAlert.pl . . . . .	270
C.12	File /src-gen/prolog-ref-spec.../outactCoordinator-oeReportOnCrisis.pl . . . . .	272
C.13	File /src-gen/prolog-ref-spec.../outactCoordinator-oeSetCrisisHandler.pl . . . . .	273
C.14	File /src-gen/prolog-ref-spec.../outactCoordinator-oeSetCrisisStatus.pl . . . . .	275
C.15	File /src-gen/prolog-ref-spec.../outactCoordinator-oeSetCrisisType.pl . . . . .	277
C.16	File /src-gen/prolog-ref-spec.../outactCoordinator-oeValidateAlert.pl . . . . .	278
C.17	File /src-gen.../outactMsrCreator-oeCreateSystemAndEnvironment.pl . . . . .	280
C.18	File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctAdministrator-init.pl . . . . .	282
C.19	File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesClasses-ctAlert-init.pl . . . . .	282
C.20	File /src-gen.../PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl . . . . .	283
C.21	File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctAuthenticated-init.pl . . . . .	283
C.22	File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctCoordinator-init.pl . . . . .	284
C.23	File /src-gen.../PrimaryTypesClasses-ctCrisis-handlingDelayPassed.pl . . . . .	284
C.24	File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctCrisis-init.pl . . . . .	285
C.25	File /src-gen.../PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl . . . . .	285
C.26	File /src-gen.../PrimaryTypesClasses-ctCrisis-isSentToCoordinator.pl . . . . .	286
C.27	File /src-gen.../PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl . . . . .	287
C.28	File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesClasses-ctHuman-init.pl . . . . .	288
C.29	File /src-gen/prolog-ref-spec.../PrimaryTypesClasses-ctHuman-isAcknowledged.pl . . . . .	288
C.30	File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesClasses-ctState-init.pl . . . . .	288
C.31	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtAlertID-is.pl . . . . .	289
C.32	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtComment-is.pl . . . . .	290
C.33	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtCoordinatorID-is.pl . . . . .	290
C.34	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtCrisisID-is.pl . . . . .	291
C.35	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtGPSLocation-is.pl . . . . .	291
C.36	File /src-gen.../PrimaryTypesDatatypes-dtGPSLocation-isNearTo.pl . . . . .	292
C.37	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtLatitude-is.pl . . . . .	293
C.38	File /src-gen/prolog-ref-spec/Operations.../PrimaryTypesDatatypes-dtLogin-is.pl . . . . .	293
C.39	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtLongitude-is.pl . . . . .	294
C.40	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtPassword-is.pl . . . . .	294
C.41	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-dtPhoneNumber-is.pl . . . . .	295
C.42	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etAlertStatus-is.pl . . . . .	295

C.43	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etCrisisStatus-is.pl . . . . .	296
C.44	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etCrisisType-is.pl . . . . .	296
C.45	File /src-gen/prolog-ref-spec.../PrimaryTypesDatatypes-etHumanKind-is.pl . . . . .	297
C.46	File /src-gen/prolog-ref-spec/Operations.../SecondaryTypesDatatypes-dtSMS-is.pl .	297
<b>Glossary</b>	. . . . .	<b>299</b>

# List of Figures

2.1	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-suDeployAndRun . . . . .	21
2.2	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-suGlobalCrisisHandling . .	23
2.3	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugAdministrateTheSystem	24
2.4	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugManageCrisis . . . . .	25
2.5	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugMonitor . . . . .	29
2.6	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-ugSecurelyUseSystem . . .	29
2.7	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeSetCrisisHandler . . . . .	30
2.8	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeSollicitateCrisisHandling	30
2.9	lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: uci-suDeployAndRun-uciSimpleAndComplete-Pa	
2.10	lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: uci-suDeployAndRun-uciSimpleAndComplete-Pa	
2.11	lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: uci-uciugSecurelyUseSystem . . . . .	34
3.1	Environment Model - Local View 01 - environment model local view - Part . . . . .	35
3.2	Environment Model - Local View 02 - environment model local view - Part . . . . .	35
3.3	Environment Model - Local View 03 - administrator actor environment mode . . . . .	36
3.4	Environment Model - Local View 04 - coordinator actor environment model . . . . .	37
3.5	Environment Model - Local View 05 - authenticated actor environment mode . . . . .	37
3.6	Environment Model - Local View 08 - police actor environment model view . . . . .	38
3.7	Environment Model - Global View 01 - em-gv-01 environment model global v . . . . .	39
4.1	Concept Model - PrimaryTypes-Classes local view 01 - Local view of all the primary types	44
4.2	Concept Model - PrimaryTypes-Classes local view 02 - local view of the ctState primary ty	45
4.3	Concept Model - PrimaryTypes-Classes local view 03 - local view of the ctAlert primary ty	45
4.4	Concept Model - PrimaryTypes-Classes local view 04 - local view of the ctCrisis primary t	45
4.5	Concept Model - PrimaryTypes-Classes global view 01 - Primary types class types global vi	46
4.6	Concept Model - PrimaryTypes-Datatypes local view 06 - local view of primary types datatype	47
4.7	Concept Model - PrimaryTypes-Datatypes global view 01 - global view of primary types dataty	52
4.8	Concept Model - SecondaryTypes-Datatypes local view 01 - Local view of the secondary types da	52
5.1	lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactActivator-oeSollicitateCrisis	
5.2	lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactComCompany-oeAlertv2	8
5.3	lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactComCompany-oeAlertv3	8
5.4	lu.uni.lassy.excalibur.examples.icrash Operation Scope: operation-scope-outactMsrCreator-oeCreateSystem	
6.1	lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: tci-testcase01-instance01-Part01	175
6.2	lu.uni.lassy.excalibur.examples.icrash Sequence Diagram: tci-testcase01-instance01-Part02	176
A.1	lu.uni.lassy.excalibur.examples.icrash Use Case Diagram: uc-oeCloseCrisis . . . . .	184

# Listings

5.1	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeSetClock</i> . . . . .	55
5.2	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeSetClock</i> . . . . .	56
5.3	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeSollicitateCrisisHandling</i> . .	57
5.4	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeSollicitateCrisisHandling</i> .	58
5.5	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeAddCoordinator</i> . . . . .	61
5.6	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeAddCoordinator</i> . . . . .	63
5.7	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeAddPolice</i> . . . . .	65
5.8	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeDeleteCoordinator</i> . . . . .	66
5.9	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeDeleteCoordinator</i> . . . .	67
5.10	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeDeletePolice</i> . . . . .	69
5.11	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeLogin</i> . . . . .	71
5.12	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeLogin</i> . . . . .	71
5.13	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeLogout</i> . . . . .	74
5.14	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeLogout</i> . . . . .	74
5.15	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeAlert</i> . . . . .	77
5.16	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeAlert</i> . . . . .	78
5.17	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeCloseCrisis</i> . . . . .	84
5.18	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeGetAlertsSet</i> . . . . .	86
5.19	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeGetCrisisSet</i> . . . . .	88
5.20	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeInvalidateAlert</i> . . . . .	90
5.21	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeReportOnCrisis</i> . . . . .	92
5.22	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeSetCrisisHandler</i> . . . . .	94
5.23	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeSetCrisisStatus</i> . . . . .	96
5.24	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeSetCrisisType</i> . . . . .	98
5.25	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeValidateAlert</i> . . . . .	100
5.26	<b>Messir</b> (MCL-oriented) specification of the operation <i>oeCreateSystemAndEnvironment</i> .	103
5.27	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>oeCreateSystemAndEnvironment</i> .	104
5.28	<b>Messir</b> (MCL-oriented) specification of the operation <i>init</i> . . . . .	110
5.29	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>init</i> . . . . .	110
5.30	<b>Messir</b> (MCL-oriented) specification of the operation <i>init</i> . . . . .	112
5.31	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>init</i> . . . . .	113
5.32	<b>Messir</b> (MCL-oriented) specification of the operation <i>isSentToCoordinator</i> . . . . .	113
5.33	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>isSentToCoordinator</i> . . .	114
5.34	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>init</i> . . . . .	115
5.35	<b>Messir</b> (MCL-oriented) specification of the operation <i>init</i> . . . . .	115
5.36	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>init</i> . . . . .	116
5.37	<b>Messir</b> (MCL-oriented) specification of the operation <i>init</i> . . . . .	117
5.38	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>init</i> . . . . .	117
5.39	<b>Messir</b> (MCL-oriented) specification of the operation <i>handlingDelayPassed</i> . . . . .	118
5.40	<b>Messir</b> (Prolog-oriented) implementation of the operation <i>handlingDelayPassed</i> . . .	119

5.41	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>maxHandlingDelayPassed</i> .	120
5.42	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>maxHandlingDelayPassed</i> .	120
5.43	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>isSentToCoordinator</i> .	121
5.44	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>isSentToCoordinator</i> .	121
5.45	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>isSentToPolice</i> .	122
5.46	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>isAllocatedIfPossible</i> .	123
5.47	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>isAllocatedIfPossible</i> .	123
5.48	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>init</i> .	125
5.49	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>init</i> .	125
5.50	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>isAcknowledged</i> .	126
5.51	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>init</i> .	127
5.52	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>init</i> .	128
5.53	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>init</i> .	128
5.54	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	129
5.55	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	130
5.56	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	131
5.57	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	131
5.58	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	132
5.59	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	132
5.60	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	133
5.61	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	133
5.62	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	134
5.63	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	135
5.64	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>isNearTo</i> .	135
5.65	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>isNearTo</i> .	136
5.66	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	137
5.67	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	137
5.68	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	138
5.69	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	139
5.70	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	140
5.71	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	140
5.72	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	141
5.73	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	141
5.74	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	142
5.75	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	142
5.76	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	143
5.77	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	144
5.78	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	144
5.79	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	145
5.80	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	145
5.81	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	146
5.82	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	146
5.83	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	147
5.84	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	147
5.85	<b>Mess1P</b> (MCL-oriented) specification of the operation <i>is</i> .	148
5.86	<b>Mess1P</b> (Prolog-oriented) implementation of the operation <i>is</i> .	148
6.1	<b>Mess1P</b> (MCL-oriented) specification of the test step <i>testcase01-ts01oeCreateSystemAndEnvironment</i> .	154
6.2	<b>Mess1P</b> (Prolog-oriented) implementation of the test step <i>testcase01-ts01oeCreateSystemAndEnvironment</i> .	154
6.3	<b>Mess1P</b> (MCL-oriented) specification of the test step <i>testcase01-ts02oeSetClock</i> .	154

6.4	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts03oeLogin</i> . . . . .	155
6.5	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts04oeAddCoordinator</i> . . . . .	156
6.6	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts05oeLogout</i> . . . . .	157
6.7	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts06oeSetClock02</i> . . . . .	158
6.8	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts07oeAlert1</i> . . . . .	159
6.9	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts08oeSetClock03</i> . . . . .	160
6.10	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts09oeSollicitateCrisisHandling</i> . . . . .	161
6.11	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts10oeLogin02</i> . . . . .	162
6.12	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts11oeGetCrisisSet</i> . . . . .	163
6.13	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts12oeSetCrisisHandler</i> . . . . .	165
6.14	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts13oeSetClock04</i> . . . . .	166
6.15	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts14oeValidateAlert</i> . . . . .	167
6.16	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts15oeAlert2</i> . . . . .	168
6.17	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts16oeSetClock05</i> . . . . .	170
6.18	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts17oeSetCrisisStatus</i> . . . . .	171
6.19	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts18oeReportOnCrisis</i> . . . . .	172
6.20	<b>Messir</b> (MCL-oriented) specification of the test step <i>testcase01-ts19oeCloseCrisis</i> . . . . .	173
B.1	Messir Spec. file .views.msr. . . . .	185
B.2	Messir Spec. file dtSMS.msr. . . . .	185
B.3	Messir Spec. file environment-actActivator-oeSetClock.msr. . . . .	186
B.4	Messir Spec. file environment-actActivator-oeSollicitateCrisisHandling.msr. . . . .	186
B.5	Messir Spec. file environment-actAdministrator-oeAddCoordinator.msr. . . . .	187
B.6	Messir Spec. file environment-actAdministrator-oeAddPolice.msr. . . . .	188
B.7	Messir Spec. file environment-actAdministrator-oeDeleteCoordinator.msr. . . . .	189
B.8	Messir Spec. file environment-actAdministrator-oeDeletePolice.msr. . . . .	190
B.9	Messir Spec. file environment-actAuthenticated.msr. . . . .	191
B.10	Messir Spec. file environment-actComCompany.msr. . . . .	193
B.11	Messir Spec. file environment-actCoordinator-oeCloseCrisis.msr. . . . .	195
B.12	Messir Spec. file environment-actCoordinator-oeGetAlertsSet.msr. . . . .	196
B.13	Messir Spec. file environment-actCoordinator-oeGetCrisisSet.msr. . . . .	196
B.14	Messir Spec. file environment-actCoordinator-oeInvalidateAlert.msr. . . . .	196
B.15	Messir Spec. file environment-actCoordinator-oeReportOnCrisis.msr. . . . .	197
B.16	Messir Spec. file environment-actCoordinator-oeSetCrisisHandler.msr. . . . .	197
B.17	Messir Spec. file environment-actCoordinator-oeSetCrisisStatus.msr. . . . .	198
B.18	Messir Spec. file environment-actCoordinator-oeSetCrisisType.msr. . . . .	198
B.19	Messir Spec. file environment-actCoordinator-oeValidateAlert.msr. . . . .	198
B.20	Messir Spec. file environment-actMsrCreator-init.msr. . . . .	199
B.21	Messir Spec. file environment-actMsrCreator-oeCreateSystemAndEnvironment.msr. . . . .	199
B.22	Messir Spec. file environment-actPolice-oeCloseCrisis.msr. . . . .	200
B.23	Messir Spec. file environment-actPolice-oeGetCrisisSet.msr. . . . .	200
B.24	Messir Spec. file environment-actPolice-oeReportOnCrisis.msr. . . . .	201
B.25	Messir Spec. file environment-actPolice-oeSetCrisisHandler.msr. . . . .	201
B.26	Messir Spec. file environment-actPolice-oeSetCrisisStatus.msr. . . . .	202
B.27	Messir Spec. file environment.msr. . . . .	202
B.28	Messir Spec. file primarytypes-associations.msr. . . . .	204
B.29	Messir Spec. file primarytypes-classes-ctAdministrator.msr. . . . .	205
B.30	Messir Spec. file primarytypes-classes-ctAlert.msr. . . . .	206
B.31	Messir Spec. file primarytypes-classes-ctAuthenticated.msr. . . . .	207
B.32	Messir Spec. file primarytypes-classes-ctCoordinator.msr. . . . .	207

B.33	Messir Spec. file primarytypes-classes-ctCrisis.msr.	207
B.34	Messir Spec. file primarytypes-classes-ctHuman.msr.	210
B.35	Messir Spec. file primarytypes-classes-ctPolice.msr.	210
B.36	Messir Spec. file primarytypes-classes-ctState.msr.	211
B.37	Messir Spec. file primarytypes-classes.msr.	212
B.38	Messir Spec. file primarytypes-datatypes-dtAlertID.msr.	214
B.39	Messir Spec. file primarytypes-datatypes-dtComment.msr.	214
B.40	Messir Spec. file primarytypes-datatypes-dtCoordinatorID.msr.	215
B.41	Messir Spec. file primarytypes-datatypes-dtCrisisID.msr.	215
B.42	Messir Spec. file primarytypes-datatypes-dtGPSLocation.msr.	215
B.43	Messir Spec. file primarytypes-datatypes-dtLogin.msr.	217
B.44	Messir Spec. file primarytypes-datatypes-dtPassword.msr.	217
B.45	Messir Spec. file primarytypes-datatypes-dtPhoneNumber.msr.	218
B.46	Messir Spec. file primarytypes-datatypes-dtPoliceID.msr.	218
B.47	Messir Spec. file primarytypes-datatypes-etAlertStatus.msr.	219
B.48	Messir Spec. file primarytypes-datatypes-etCrisisStatus.msr.	219
B.49	Messir Spec. file primarytypes-datatypes-etCrisisType.msr.	220
B.50	Messir Spec. file primarytypes-datatypes-etHumanKind.msr.	220
B.51	Messir Spec. file primarytypes-datatypes.msr.	221
B.52	Messir Spec. file secondarytypes-associations.msr.	222
B.53	Messir Spec. file secondarytypes-classes.msr.	222
B.54	Messir Spec. file secondarytypes-datatypes.msr.	222
B.55	Messir Spec. file subfunctions-usecases.msr.	223
B.56	Messir Spec. file tc-testcase01.msr.	225
B.57	Messir Spec. file tci-testcase01-instance01.msr.	233
B.58	Messir Spec. file usecase-suDeployAndRun.msr.	243
B.59	Messir Spec. file usecase-suGlobalCrisisHandling.msr.	249
B.60	Messir Spec. file usecase-ugAdministrateTheSystem.msr.	250
B.61	Messir Spec. file usecase-ugManageCrisis.msr.	251
B.62	Messir Spec. file usecase-ugMonitor.msr.	251
B.63	Messir Spec. file usecase-ugSecurelyUseSystem.msr.	252
B.64	Messir Spec. file usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr.	252
C.1	Prolog file outactActivator-oeSetClock.pl.	255
C.2	Prolog file outactActivator-oeSollicitateCrisisHandling.pl.	256
C.3	Prolog file outactAdministrator-oeAddCoordinator.pl.	258
C.4	Prolog file outactAdministrator-oeDeleteCoordinator.pl.	259
C.5	Prolog file outactAuthenticated-oeLogin.pl.	260
C.6	Prolog file outactAuthenticated-oeLogout.pl.	262
C.7	Prolog file outactComCompany-oeAlert.pl.	263
C.8	Prolog file outactCoordinator-oeCloseCrisis.pl.	267
C.9	Prolog file outactCoordinator-oeGetAlertsSet.pl.	268
C.10	Prolog file outactCoordinator-oeGetCrisisSet.pl.	269
C.11	Prolog file outactCoordinator-oeInvalidateAlert.pl.	270
C.12	Prolog file outactCoordinator-oeReportOnCrisis.pl.	272
C.13	Prolog file outactCoordinator-oeSetCrisisHandler.pl.	273
C.14	Prolog file outactCoordinator-oeSetCrisisStatus.pl.	275
C.15	Prolog file outactCoordinator-oeSetCrisisType.pl.	277
C.16	Prolog file outactCoordinator-oeValidateAlert.pl.	278
C.17	Prolog file outactMsrCreator-oeCreateSystemAndEnvironment.pl.	280

C.18 Prolog file PrimaryTypesClasses-ctAdministrator-init.pl . . . . .	282
C.19 Prolog file PrimaryTypesClasses-ctAlert-init.pl . . . . .	282
C.20 Prolog file PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl . . . . .	283
C.21 Prolog file PrimaryTypesClasses-ctAuthenticated-init.pl . . . . .	283
C.22 Prolog file PrimaryTypesClasses-ctCoordinator-init.pl . . . . .	284
C.23 Prolog file PrimaryTypesClasses-ctCrisis-handlingDelayPassed.pl . . . . .	284
C.24 Prolog file PrimaryTypesClasses-ctCrisis-init.pl . . . . .	285
C.25 Prolog file PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl . . . . .	285
C.26 Prolog file PrimaryTypesClasses-ctCrisis-isSentToCoordinator.pl . . . . .	286
C.27 Prolog file PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl . . . . .	287
C.28 Prolog file PrimaryTypesClasses-ctHuman-init.pl . . . . .	288
C.29 Prolog file PrimaryTypesClasses-ctHuman-isAcknowledged.pl . . . . .	288
C.30 Prolog file PrimaryTypesClasses-ctState-init.pl . . . . .	288
C.31 Prolog file PrimaryTypesDatatypes-dtAlertID-is.pl . . . . .	289
C.32 Prolog file PrimaryTypesDatatypes-dtComment-is.pl . . . . .	290
C.33 Prolog file PrimaryTypesDatatypes-dtCoordinatorID-is.pl . . . . .	290
C.34 Prolog file PrimaryTypesDatatypes-dtCrisisID-is.pl . . . . .	291
C.35 Prolog file PrimaryTypesDatatypes-dtGPSLocation-is.pl . . . . .	291
C.36 Prolog file PrimaryTypesDatatypes-dtGPSLocation-isNearTo.pl . . . . .	292
C.37 Prolog file PrimaryTypesDatatypes-dtLatitude-is.pl . . . . .	293
C.38 Prolog file PrimaryTypesDatatypes-dtLogin-is.pl . . . . .	293
C.39 Prolog file PrimaryTypesDatatypes-dtLongitude-is.pl . . . . .	294
C.40 Prolog file PrimaryTypesDatatypes-dtPassword-is.pl . . . . .	294
C.41 Prolog file PrimaryTypesDatatypes-dtPhoneNumber-is.pl . . . . .	295
C.42 Prolog file PrimaryTypesDatatypes-etAlertStatus-is.pl . . . . .	295
C.43 Prolog file PrimaryTypesDatatypes-etCrisisStatus-is.pl . . . . .	296
C.44 Prolog file PrimaryTypesDatatypes-etCrisisType-is.pl . . . . .	296
C.45 Prolog file PrimaryTypesDatatypes-etHumanKind-is.pl . . . . .	297
C.46 Prolog file SecondaryTypesDatatypes-dtSMS-is.pl . . . . .	297

# Chapter 1

## Introduction

### 1.1 Overview

*iCrash* is a simple system dedicated to any person who wants to inform of a car crash crisis situation in order to allow for crisis handling. At anytime and anywhere, anyone can be the witness or victim of a car crash and might be in a situation allowing for alerting this crisis. The *iCrash* system has for objectives to support crisis declaration and secure administration and crisis handling by the *iCrash* professional users.

### 1.2 Purpose and recipients of the document

This document is an analysis document complying with the **Messip** methodology [1]. Its intent is to provide an example of a precise specification of the functional properties of the *iCrash* system.

The recipients of this document are:

- the *iCrash* system's buyer company (ABC): this document is used as a contractual document jointly with any other document considered as useful (as requirement elicitation document, ...) in order to have a higher degree of precision in requirement description. It is also used as a basis document for the *iCrash* system validation using specification based testing.
- the *iCrash* system development company (ADC) is expected to use this document as the basis for development (mainly design, implementation, maintenance). It is also used for verification and validation using test plans defined using the analysis models described in this document and according to the **Messip** methodology.

### 1.3 Application Domain

The *iCrash* system belongs to the Crisis Management Systems Domain. It is a system dedicated to crisis professional and non professional end users. It has to be considered as an autonomous and external service for the society. It is not an institutional system certified and guaranteed by any governmental entity and thus, must be used with caution.

### 1.4 Definitions, acronyms and abbreviations

N.A.

## 1.5 Document structure

The document structure is designed to be coherent with the **Messip** methodology [1]. Section 2 provides a general description of the system purpose, its users, its environment and some general non functional requirements. A more detailed description of the non functional requirements, if any, are provided in section ?? . The **system operation** triggered by events sent by the external **actors** belonging to the environment are described in Section 3. The **iCrash** concepts used to represent the any persistent or transient information is given in Section 4. The precise specification of the system operations in term of system's state changes, events sent together with the constraints on the allowed sequences of system operations are described in Section 5.

# Chapter 2

## General Description

In the context of the **Messip** method, the information provided in this section is intended to present the system for which the **Messip** analysis is provided. The content of this section is made accordingly to the requirements elicitation document that might have been done during the project but also adapted coherently in order to be an abstract introduction to the **Messip** analysis.

### 2.1 Domain Stakeholders

All stakeholders of the system are detailed in this section. After a brief description of a stakeholder, its objectives are first stated. Thereafter, the responsibilities of the stakeholder are detailed which help to achieve the stakeholder objectives to a certain degree. While the objectives characterize the general problems addressed by the *iCrash* system, the responsibilities describe concrete actions that are expected from a stakeholder. Some of these responsibilities can be traced looking at the use case described in Section A.1, and hence must be supported by the *iCrash* system. All stakeholders listed in this section have an interest in the system or are affected by the system in some way, but only a subset of the stakeholders are directly involved in the use cases described. Let us remind that use case diagrams or descriptions are not **Messip** analysis phase mandatory outputs. They are proposed as informal means to help understanding the semantics of the system specification made of the mandatory analysis models, which provide a complete executable specification.

#### 2.1.1 Communication Company

A Communication Company is a company that has the capacity to ensure communication of information between its customers and the *iCrash* system. The objectives of a Communication Company are:

- to be able to deliver any SMS sent by any human to the *iCrash* 's phone number.
- to be able to transmit SMS messages from the ABC company that owns the *iCrash* system to any human having an SMS compatible device accessible using a phone number.

In order to achieve these objectives, the responsibilities of a Communication Company are:

- ensure confidentiality and integrity of the information sent by a human to the *iCrash* system or from the system to a human.
- to be always available and reliable.

### 2.1.2 Humans

A human is any person who considers himself related to a car crash either as a witness, a victim or an anonymous person. The objectives of a human are:

- inform the *iCrash* system about the crisis situation he detected.
- be sure that the ABC company has been informed about the situation.
- to be informed about the situation of the crisis he is related to as a victim or witness.

In order to achieve these objectives, the responsibilities of a human are:

- to provide as much details as possible concerning the crisis to the ABC company.
- to declare a crisis only if the crisis is real.
- to have access to the SMS compatible communication device he used to communicate with the *iCrash* system.

### 2.1.3 Coordinators

A coordinator is an employee of the ABC company being responsible of handling one or several crises. The objectives of a coordinator are:

- to securely monitor the existing alerts and crisis.
- to securely manage alerts and crisis until their termination.

In order to achieve these objectives, the responsibilities of a coordinator are:

- to be capable to determine how an alert received should be considered.
- to be available to react to requests to handle alerts and crisis.
- to be autonomous in handling crisis and to report on its handling.
- to be able to decide when a crisis or an alert can be closed.
- to know its system identification information for secure usage of the system.

### 2.1.4 Police

A police is responsible of handling one or several crisis with type 'huge'. The objectives of a police are:

- to securely monitor the existing crisis.
- to securely manage crisis until their termination.

In order to achieve these objectives, the responsibilities of a police are:

- to be available to react to requests to handle crisis.
- to be autonomous in handling crisis and to report on its handling.
- to be able to decide when a crisis can be closed.
- to know its system identification information for secure usage of the system.

### 2.1.5 Administrator

An administrator is a employee of the ABC company being responsible of administrating the *iCrash* system. The objectives of an administrator are:

- to add or delete coordinator actors from the system and its environment.
- to add or delete police actors from the system and its environment.

In order to achieve these objectives, the responsibilities of a coordinator are:

- know the company employees that can be coordinators and that have access to the system.
- to know its system identification information for secure usage of the system.
- to know the security policy of the ABC company.
- to communicate the coordinators their identification information for secure system usage.

In order to achieve these objectives, the responsibilities of a police are:

- know the company employees that can be coordinators and that have access to the system.
- to know its system identification information for secure usage of the system.
- to know the security policy of the ABC company.
- to communicate the polices their identification information for secure system usage.

### 2.1.6 Creator

Any system has a **Creator** stakeholder which is a technician who is installing the *iCrash* system on the targeted deployment infrastructure.

The objectives of a **Creator** are:

- to install the *iCrash* system
- to define the values for the initial system's state
- to define the values for the initial system's environment
- to ensure the integration of the *iCrash* system with its initial environment

In order to achieve these objectives, the responsibilities of a **Creator** are:

- provide the necessary data to the *iCrash* system for its initialization.

### 2.1.7 Activator

An **activator** is a logical representation of the active part the *iCrash* system. It represents an implicit stakeholder belonging to the system's environment that interacts with the *iCrash* system autonomously without the need of a external entity. It is usually used for representing time triggered functionalities.

The objectives of a **activator** are:

- to communicate the current time to the system

- to notify the administrator that some crisis are still pending for a too long time.

In order to achieve these objectives, the responsibilities of a **activator** are:

- to know the current universal time
- to send the messages to the system according to the time constraints specifically defined for it.

## 2.2 System's Actors

The objective of this section is not to provide the full requirement elicitation document in this section but to reuse a part of this document to provide an informal introduction to the **Messip** specification of the system under development. The use case model is made of a use case diagrams modelling abstractly and informally the actors and their use cases together with a set of use cases descriptions. In addition, those diagrams and description tables are adapted to the **Messip** specification since actor and messages names together with parameters are partly adapted to be consistent with the specification identifiers (see [1] for more details).

Among all the stakeholders presented in the previous section, we can determine five types of direct actors<sup>1</sup>:

- `actComCompany`: for the Communication Company stakeholder.
- `actAdministrator`: for the Administrator stakeholder.
- `actCoordinator`: for the Coordinators stakeholders.
- `actPolice`: for the Polices stakeholders.
- `actActivator`: for the Activator stakeholder.
- `actMsrCreator`: for the Creator stakeholder.

In addition to those system actors, we can add five other types of actors related to the system's ones. Those five actors are grouped into two categories:

- *Indirect actors*
  - *Witness*: for any human that is a witness of a car crash
  - *Victim*: for any human that is a victim of a car crash
  - *Anonymous*: for any human that want to inform about a car crash while staying anonymous.
- *Abstract actors*
  - `actHuman`: represent abstractly any kind of human being actor wanting to communicate with the ABC system in the context of a car crash.
  - `actAuthenticated`: for the logical Activator stakeholder.

## 2.3 Use Cases Model

This section contains the use cases elicited during the requirements elicitation phase. The use cases are textually described as suggested by the **Messip** method and inspired by the standard Cokburn template [2].

### 2.3.1 Use Cases

#### 2.3.1.1 summary-suDeployAndRun

The goal is to install the iCrash system on its infrastructure and to exploit its capacities related to the secure administration and efficient handling of car crash situations depending on alerts received.

---

<sup>1</sup>The naming conventions in **Messip** propose to start each type name by lowercase letters indicating the meta model type used (i.e. act for actors, ct for class type, ....). In addition to ease the reading it makes the translational semantics into Prolog code more straightforward.

<b>USE-CASE DESCRIPTION</b>	
<i>Name</i>	suDeployAndRun
<i>Scope</i>	system
<i>Level</i>	summary
<b>Primary actor(s)</b>	
1	actAdministrator[active]
<b>Secondary actor(s)</b>	
1	actMsrCreator[active]
2	actCoordinator[active, multiple]
3	actPolice[active, multiple]
4	actActivator[proactive]
5	actComCompany[active]
<b>Goal(s) description</b>	
The goal is to install the iCrash system on its infrastructure and to exploit its capacities related to the secure administration and efficient handling of car crash situations depending on alerts received.	
<b>Reuse</b>	
1	<u>oeCreateSystemAndEnvironment [1..1]</u>
2	<u>ugAdministateTheSystem [1...*]</u>
3	<u>suGlobalCrisisHandling [1...*]</u>
4	<u>oeSetClock [1...*]</u>
5	<u>oeSollicitateCrisisHandling [0...*]</u>
6	<u>oeAlert [1...*]</u>
<b>Protocol condition(s)</b>	
1	the iCrash system has never been deployed and used
<b>Pre-condition(s)</b>	
1	none
<b>Main post-condition(s)</b>	
1	the iCrash system has been created and has handled the crisis situations for which it received alerts through the communication company.
<b>Main Steps</b>	
a	the actor <code>actMsrCreator</code> executes the <u>oeCreateSystemAndEnvironment</u> use case
b	the actor <code>actAdministrator</code> executes the <u>ugAdministateTheSystem</u> use case
c	the actor <code>actComCompany</code> executes the <u>oeAlert</u> use case
d	the actor <code>actActivator</code> executes the <u>oeSetClock</u> use case
e	the actor <code>actActivator</code> executes the <u>oeSollicitateCrisisHandling</u> use case
f	the actor <code>actCoordinator</code> executes the <u>suGlobalCrisisHandling</u> use case
g	the actor <code>actPolice</code> executes the <u>suGlobalCrisisHandling</u> use case
<b>Steps Ordering Constraints</b>	
1	step (a) must be always the first step.
2	step (f) can be executed by different actCoordinator actors.
3	step (h) can be executed by different actPolice actors.
4	if (e) then previously (d).

Figure 2.1 shows the use case diagram for the suDeployAndRun summary use case

Figure 2.1: suDeployAndRun summary use case

### 2.3.1.2 summary-suGlobalCrisisHandling

the actCoordinator's goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis. the actPolice's goal is to monitor the crisis received in order to act as necessary to handle the crisis.

USE-CASE DESCRIPTION	
Name	suGlobalCrisisHandling
Scope	system
Level	summary
<b>Primary actor(s)</b>	
1	actCoordinator[active]
2	actPolice[active]
<b>Goal(s) description</b>	
the actCoordinator's goal is to monitor the alerts received and the corresponding crisis in order to act as necessary to handle the crisis. the actPolice's goal is to monitor the crisis received in order to act as necessary to handle the crisis.	
<b>Reuse</b>	
1	<u>ugSecurelyUseSystem</u> [1..*]
2	<u>ugMonitor</u> [1..*]
3	<u>ugManageCrisis</u> [1..*]
<b>Protocol condition(s)</b>	
1	the iCrash system has been deployed
2	the coordinator actor or the police involved in the use case has been declared by the actor actAdministrator
<b>Pre-condition(s)</b>	
1	none
<b>Main post-condition(s)</b>	
1	modifications have been made by the coordinator or by the police on existing alerts or crisis OR the coordinator or the police requested an updated status on existing alerts or crisis.
<b>Main Steps</b>	
a	the actor <code>actCoordinator</code> executes the <u>ugSecurelyUseSystem</u> use case
b	the actor <code>actCoordinator</code> executes the <u>ugMonitor</u> use case
c	the actor <code>actCoordinator</code> executes the <u>ugManageCrisis</u> use case
d	the actor <code>actPolice</code> executes the <u>ugSecurelyUseSystem</u> use case
e	the actor <code>actPolice</code> executes the <u>ugMonitor</u> use case
f	the actor <code>actPolice</code> executes the <u>ugManageCrisis</u> use case
<b>Steps Ordering Constraints</b>	
1	steps (a) (b) and (c) executions are interleaved (steps (b) and (c) have their protocol constrained by steps of (a)). steps (d) (e) and (f) executions are interleaved (steps (e) and (f) have their protocol constrained by steps of (d)).
2	steps (a) (b) and (c) can be executed multiple times. steps (d) (e) and (f) can be executed multiple times.

Figure 2.2 shows the use case diagram for the suGlobalCrisisHandling user goal use case

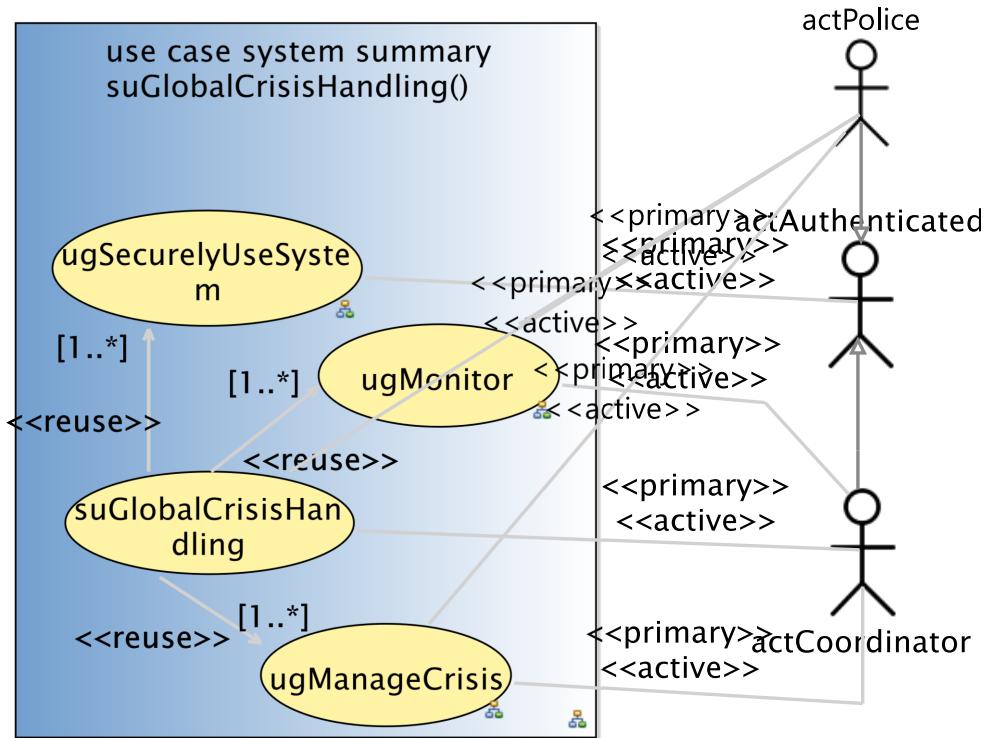


Figure 2.2: suGlobalCrisisHandling user goal use case

### 2.3.1.3 usergoal-ugAdministateTheSystem

the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators and polices that will be granted the responsibility to handle alerts and crisis.

USE-CASE DESCRIPTION	
Name	ugAdministateTheSystem
Scope	system
Level	usergoal
<i>Primary actor(s)</i>	
1	actAdministrator[active]
<i>Goal(s) description</i>	
the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators and polices that will be granted the responsibility to handle alerts and crisis.	
<i>Reuse</i>	
1	<u>ugSecurelyUseSystem</u> [1..*]
2	<u>oeAddCoordinator</u> [1..*]
3	<u>oeDeleteCoordinator</u> [0..*]
4	<u>oeAddPolice</u> [1..*]
5	<u>oeDeletePolice</u> [0..*]
<i>Protocol condition(s)</i>	
1	the iCrash system has been deployed

*continues in next page ...*

**... Use-Case Description table continuation**

<b>Pre-condition(s)</b>	
1	none
<b>Main post-condition(s)</b>	
1	modifications have been made to the system and its environment concerning existing or new coordinators and polices.
<b>Main Steps</b>	
a	the actor <code>actAdministrator</code> executes the <code>ugSecurelyUseSystem</code> use case
b	the actor <code>actAdministrator</code> executes the <code>oeAddCoordinator</code> use case
c	the actor <code>actAdministrator</code> executes the <code>oeDeleteCoordinator</code> use case
d	the actor <code>actAdministrator</code> executes the <code>oeAddPolice</code> use case
e	the actor <code>actAdministrator</code> executes the <code>oeDeletePolice</code> use case
<b>Steps Ordering Constraints</b>	
1	steps (a) (b) (c) (d) and (f) executions are interleaved (steps (b) (c) (d) and (f) have their protocol constrained by steps of (a)).
2	steps (a) (b) (c) (d) and (f) can be executed multiple times.

Figure 2.3 shows the use case diagram for the `ugAdministateTheSystem` user goal use case

Figure 2.3: `ugAdministateTheSystem` user goal use case

#### 2.3.1.4 usergoal-ugManageCrisis

The goal is to do an action that makes the handling of a crisis or an alert progress.

USE-CASE DESCRIPTION	
Name	<code>ugManageCrisis</code>
Scope	system
Level	usergoal
<b>Primary actor(s)</b>	
1	<code>actCoordinator[active]</code>
2	<code>actPolice[active]</code>
<b>Goal(s) description</b>	
The goal is to do an action that makes the handling of a crisis or an alert progress.	
<b>Reuse</b>	
1	<code>oeValidateAlert [0...*]</code>
2	<code>oeSetCrisisStatus [0...*]</code>
3	<code>oeSetCrisisHandler [0...*]</code>
4	<code>oeReportOnCrisis [0...*]</code>
5	<code>oeCloseCrisis [0...*]</code>
6	<code>oeInvalidateAlert [0...*]</code>

*continues in next page ...*

**... Use-Case Description table continuation**

<b>Protocol condition(s)</b>	
1	the iCrash system has been deployed
<b>Pre-condition(s)</b>	
1	none
<b>Main post-condition(s)</b>	
1	there exist one alert or one crisis whose related information has been changed.
<b>Main Steps</b>	
a	the actor <code>actCoordinator</code> executes the <code>oeValidateAlert</code> use case
b	the actor <code>actCoordinator</code> executes the <code>oeSetCrisisStatus</code> use case
c	the actor <code>actCoordinator</code> executes the <code>oeSetCrisisType</code> use case
d	the actor <code>actCoordinator</code> executes the <code>oeSetCrisisHandler</code> use case
e	the actor <code>actCoordinator</code> executes the <code>oeReportOnCrisis</code> use case
f	the actor <code>actCoordinator</code> executes the <code>oeCloseCrisis</code> use case
g	the actor <code>actCoordinator</code> executes the <code>oeInvalidateAlert</code> use case
h	the actor <code>actPolice</code> executes the <code>oeSetCrisisStatus</code> use case
i	the actor <code>actPolice</code> executes the <code>oeSetCrisisHandler</code> use case
j	the actor <code>actPolice</code> executes the <code>oeReportOnCrisis</code> use case
k	the actor <code>actPolice</code> executes the <code>oeCloseCrisis</code> use case
<b>Steps Ordering Constraints</b>	
1	managing a crisis is doing one of the indicated use cases.

Figure 2.4 shows the use case diagram for the ugManageCrisis user goal use case

Figure 2.4: ugManageCrisis user goal use case

**2.3.1.5 usergoal-ugMonitor**

the `actCoordinator`'s goal is to get the detailed list of existing crisis or alerts to decide on next actions to undertake. the `actPolice`'s goal is to get the detailed list of existing crisis to decide on next actions to undertake.

USE-CASE DESCRIPTION	
<i>Name</i>	ugMonitor
<i>Scope</i>	system
<i>Level</i>	usergoal
<b>Primary actor(s)</b>	
1	<code>actCoordinator[active]</code>
2	<code>actPolice[active]</code>
<b>Goal(s) description</b>	

*continues in next page ...*

**... Use-Case Description table continuation**

the actCoordinator's goal is to get the detailed list of existing crisis or alerts to decide on next actions to undertake. the actPolice's goal is to get the detailed list of existing crisis to decide on next actions to undertake.

***Reuse***

- 1      oeGetCrisisSet [0..\*]
- 2      oeGetAlertsSet [0..\*]

***Protocol condition(s)***

- 1      the iCrash system has been deployed

***Pre-condition(s)***

- 1      none

***Main post-condition(s)***

- 1      none

***Main Steps***

- a      the actor `actCoordinator` executes the oeGetAlertsSet use case
- b      the actor `actCoordinator` executes the oeGetCrisisSet use case
- c      the actor `actPolice` executes the oeGetCrisisSet use case

Figure 2.5 shows the use case diagram for the ugMonitor user goal use case

**2.3.1.6 usergoal-ugSecurelyUseSystem**

the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators and polices that will be granted the responsibility to handle alerts and crisis.

USE-CASE DESCRIPTION	
Name	ugSecurelyUseSystem
Scope	system
Level	usergoal
<b><i>Primary actor(s)</i></b>	
1	<code>actAuthenticated[active]</code>
<b><i>Goal(s) description</i></b>	
the actAdministrator's goal is to follow an identification procedure to be allowed to add or delete the necessary crisis coordinators and polices that will be granted the responsibility to handle alerts and crisis.	
<b><i>Reuse</i></b>	
1	<u>oeLogin [1..1]</u>
2	<u>oeLogout [1..1]</u>
<b><i>Protocol condition(s)</i></b>	
1	the iCrash system has been deployed
<b><i>Pre-condition(s)</i></b>	
1	none
<b><i>Main post-condition(s)</i></b>	
1	the <code>actAuthenticated</code> is known by the system not to be logged.
<b><i>Main Steps</i></b>	
a	the actor <code>actAuthenticated</code> executes the <u>oeLogin</u> use case

***continues in next page ...***

**... Use-Case Description table continuation**

b	the actor <code>actAuthenticated</code> executes the <code>oeLogout</code> use case
<b>Steps Ordering Constraints</b>	
1	step (a) must always precede step (b).

Figure 2.6 shows the use case diagram for the `ugSecurelyUseSystem` user goal use case

**2.3.1.7 subfunction-oeSetCrisisHandler**

goal is to declare himself as been the handler of a crisis having the specified id.

USE-CASE DESCRIPTION	
<i>Name</i>	<code>oeSetCrisisHandler</code>
<i>Scope</i>	system
<i>Level</i>	subfunction
<b>Parameters</b>	
AdtCrisisID: <code>dtCrisisID</code> 1	
<b>Primary actor(s)</b>	
1	<code>actCoordinator[active]</code>
<b>Secondary actor(s)</b>	
1	<code>actCoordinator[passive]</code>
2	<code>actComCompany[passive, multiple]</code>
<b>Goal(s) description</b>	
goal is to declare himself as been the handler of a crisis having the specified id.	
<b>Protocol condition(s)</b>	
1	
<b>Pre-condition(s)</b>	
1	
<b>Main post-condition(s)</b>	
1	
<b>Additional Information</b>	
none	

Figure 2.7 shows the use case diagram for the `oeSetCrisisHandler` subfunction use case

**2.3.1.8 subfunction-oeSollicitateCrisisHandling**

the `actActivator`'s goal is to decrease the number of unhandled crisis.

USE-CASE DESCRIPTION	
<i>Name</i>	<code>oeSollicitateCrisisHandling</code>
<i>Scope</i>	system
<i>Level</i>	subfunction
<b>Primary actor(s)</b>	
1	<code>actActivator[proactive]</code>

*continues in next page ...*

**... Use-Case Description table continuation**

<b>Secondary actor(s)</b>
1 actCoordinator[passive, multiple]
2 actPolice[passive, multiple]
3 actAdministrator[passive]
<b>Goal(s) description</b>
the actActivator's goal is to decrease the number of unhandled crisis.
<b>Protocol condition(s)</b>
1 the iCrash system has been deployed.
2 there exist some crisis still pending and for which no solicitation has been sent to the administrator, the coordinators and the polices for more than a predefined maximum delay.
<b>Pre-condition(s)</b>
1 none
<b>Main post-condition(s)</b>
1 a simple text message ieMessage('There are alerts not treated since more than the defined delay. Please REACT !') is sent to the system administrator, all the coordinators and all the polices of the environment for each crisis that is known to be not handled and for which no solicitation has been sent to the administrator, the coordinators and the polices for more than a predefined maximum delay.'
2 the reminder period for the concerned crisis is initialized.

Figure 2.8 shows the use case diagram for the oeSollicitateCrisisHandling subfunction use case

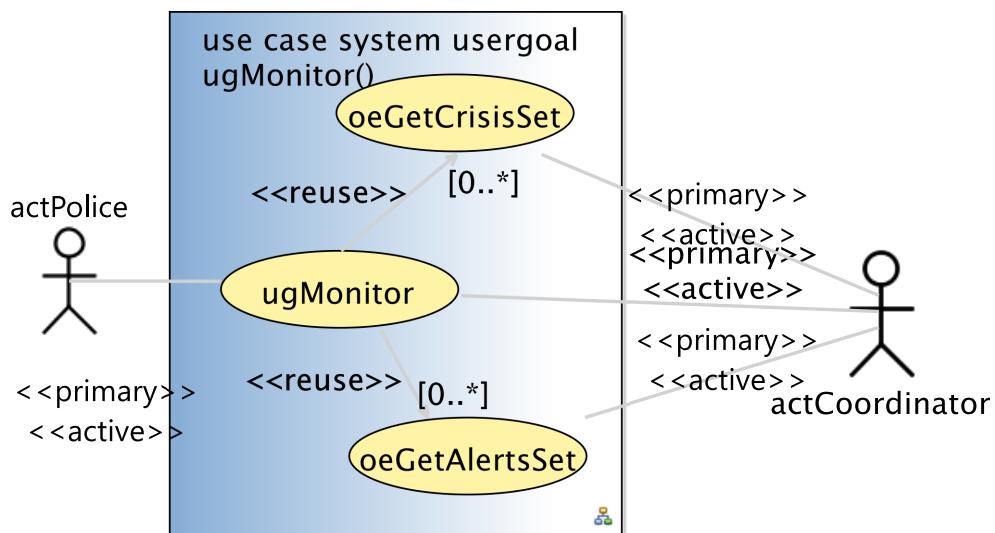


Figure 2.5: ugMonitor user goal use case

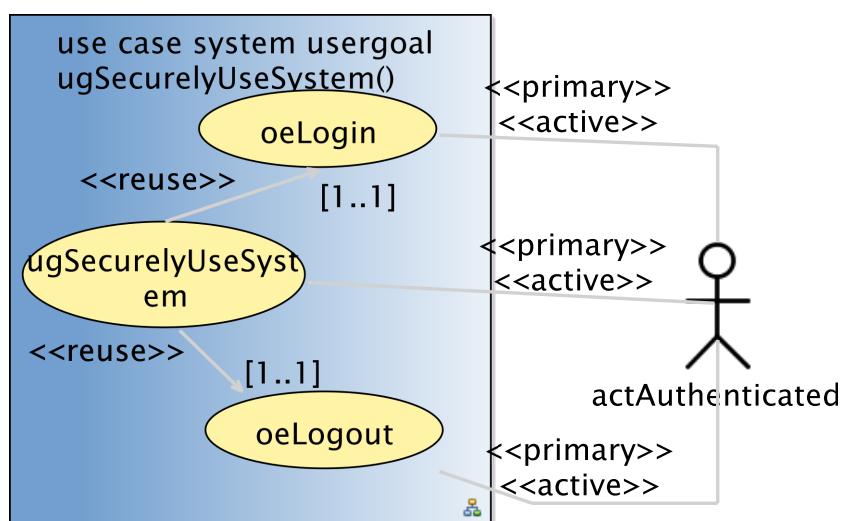


Figure 2.6: ugSecurelyUseSystem user goal use case

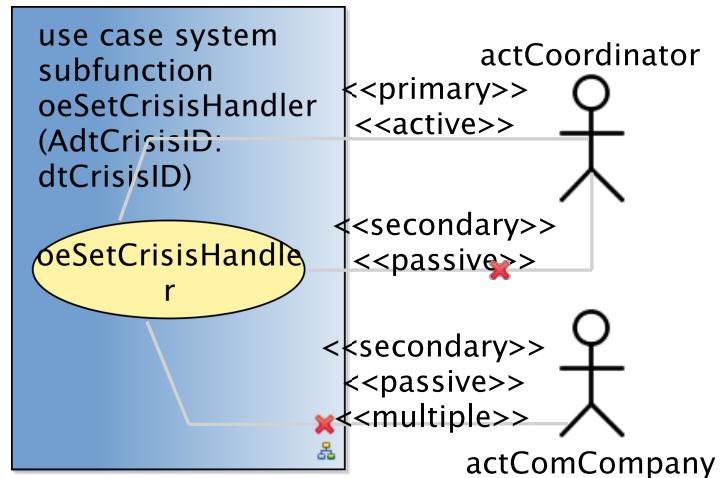


Figure 2.7: oeSetCrisisHandler subfunction use case

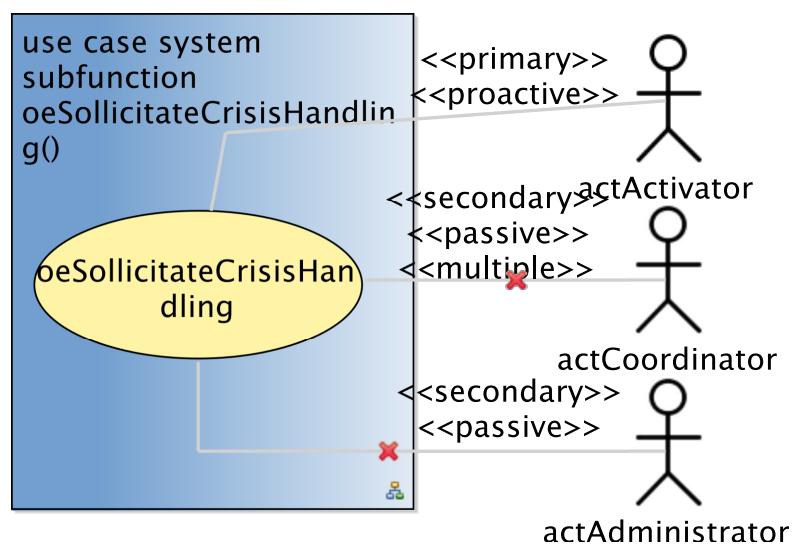


Figure 2.8: oeSollicitateCrisisHandling subfunction use case

### 2.3.2 Use Case Instance(s)

#### 2.3.2.1 Use-Case Instance - uciSimpleAndCompletePart01:suDeployAndRun

First part of a use case instance for the summary use case `suDeployAndRun` illustrating a simple and complete interaction scenario primarily handled by an administrator in a concrete situation.

SUMMARY USE-CASE INSTANCE	
<i>Instantiated Use Case</i>	<code>suDeployAndRun</code>
<i>Instance ID</i>	<code>uciSimpleAndCompletePart01</code>
<i>Remarks</i>	<ul style="list-style-type: none"> <li>a shows the system initialization and the first administrative tasks by the administrator.</li> <li>b The unique and always existing <code>actMsrCreator</code> actor instance (named here <code>theCreator</code>) requests the initialization of the system and its environment (made of one administrator identified here by <code>bill</code>), one activator actor (identified by <code>theClock</code>) and indicating that the number of communication company actor instances for the system's environment is 4 (one of them is identified here by <code>tango</code>)</li> <li>c the administrator logs in to initialize a coordinator</li> <li>d an alert is received. Time is going on without having the coordinator handling the alert which let's the proactive actor trigger the automatic solicitation of crisis handling.</li> <li>e this first part stops before the coordinator logs in the system.</li> </ul>

Figure 2.9 shows the sequence diagram representing the first part of a simple and complete use case instance for the summary use case `suDeployAndRun`.

#### 2.3.2.2 Use-Case Instance - uciSimpleAndCompletePart02:suDeployAndRun

Second part of a simple and complete use case instance for the summary use case `suDeployAndRun` illustrating a simple and complete interaction scenario primarily handled by an administrator in a concrete situation.

SUMMARY USE-CASE INSTANCE	
<i>Instantiated Use Case</i>	<code>suDeployAndRun</code>
<i>Instance ID</i>	<code>uciSimpleAndCompletePart02</code>
<i>Remarks</i>	<ul style="list-style-type: none"> <li>a starts when the coordinator logs in the system until the full handling of all the existing crisis.</li> <li>b shows an instantiated case of handling of a crisis by a coordinator until its closure after reporting.</li> </ul>

Figure 2.10 shows the sequence diagram representing the second part of a simple and complete use case instance for the summary use case `suDeployAndRun`.

#### 2.3.2.3 Use-Case Instance - uciugSecurelyUseSystem:ugSecurelyUseSystem

shows the use case diagram for the `uciugSecurelyUseSystem` user goal use case

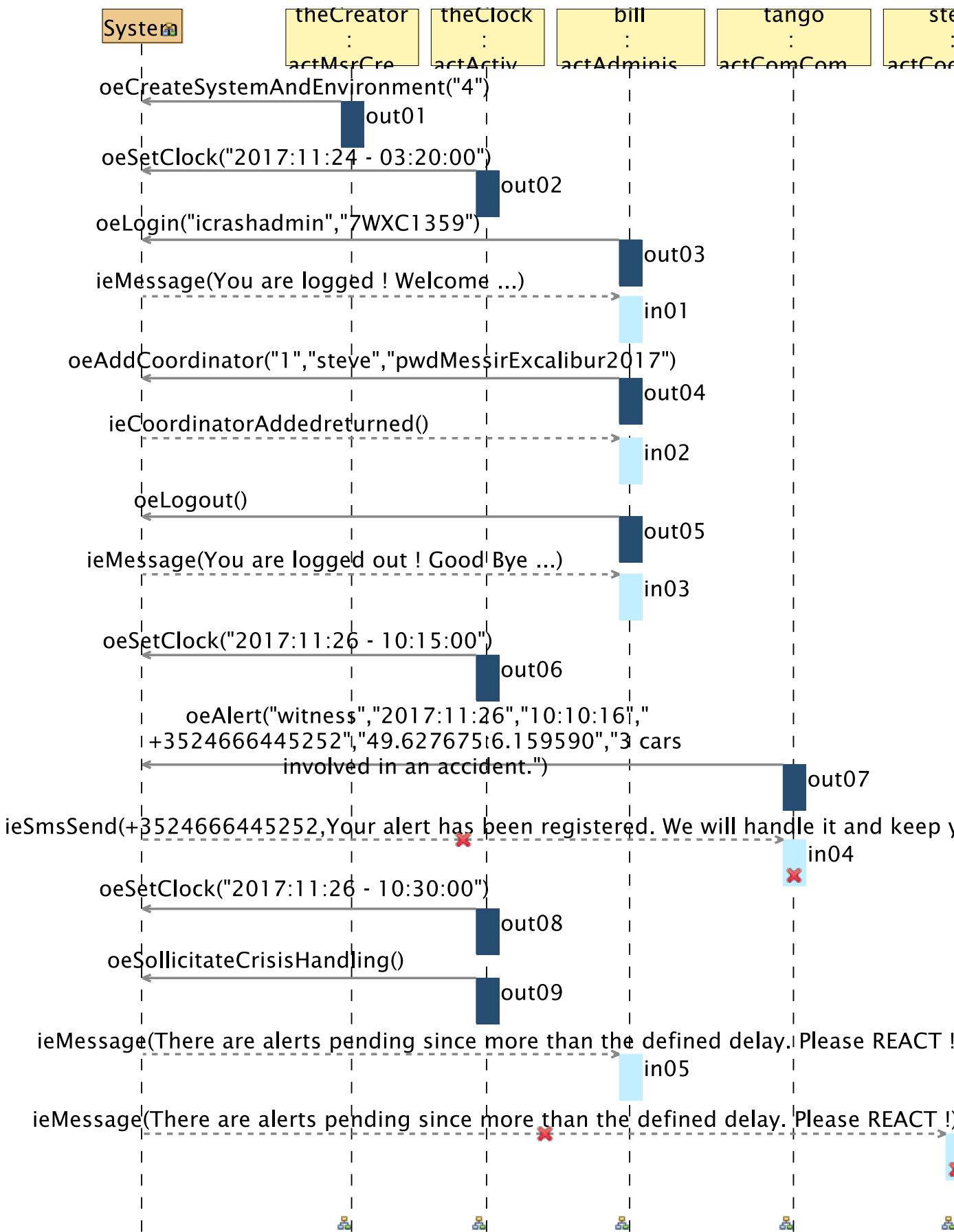


Figure 2.9: uci-suDeployAndRun-uciSimpleAndComplete-Part01

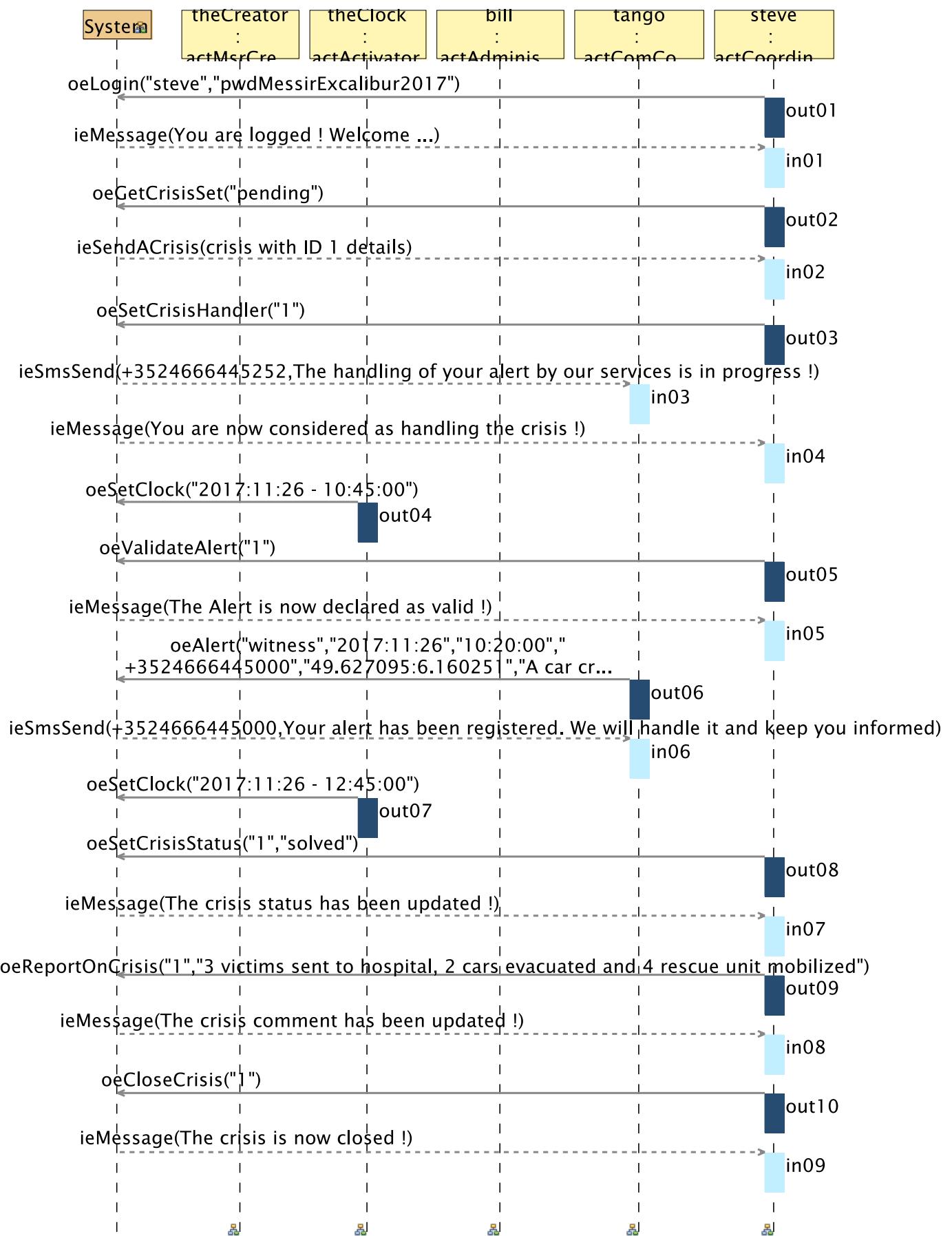


Figure 2.10: uci-suDeployAndRun-uciSimpleAndComplete-Part02 use case instance sequence diagram

USERGOAL USE-CASE INSTANCE
<i>Instantiated Use Case</i> ugSecurelyUseSystem
<i>Instance ID</i> uciugSecurelyUseSystem

Figure 2.11 shows the use case diagram for the uciugSecurelyUseSystem user goal use case

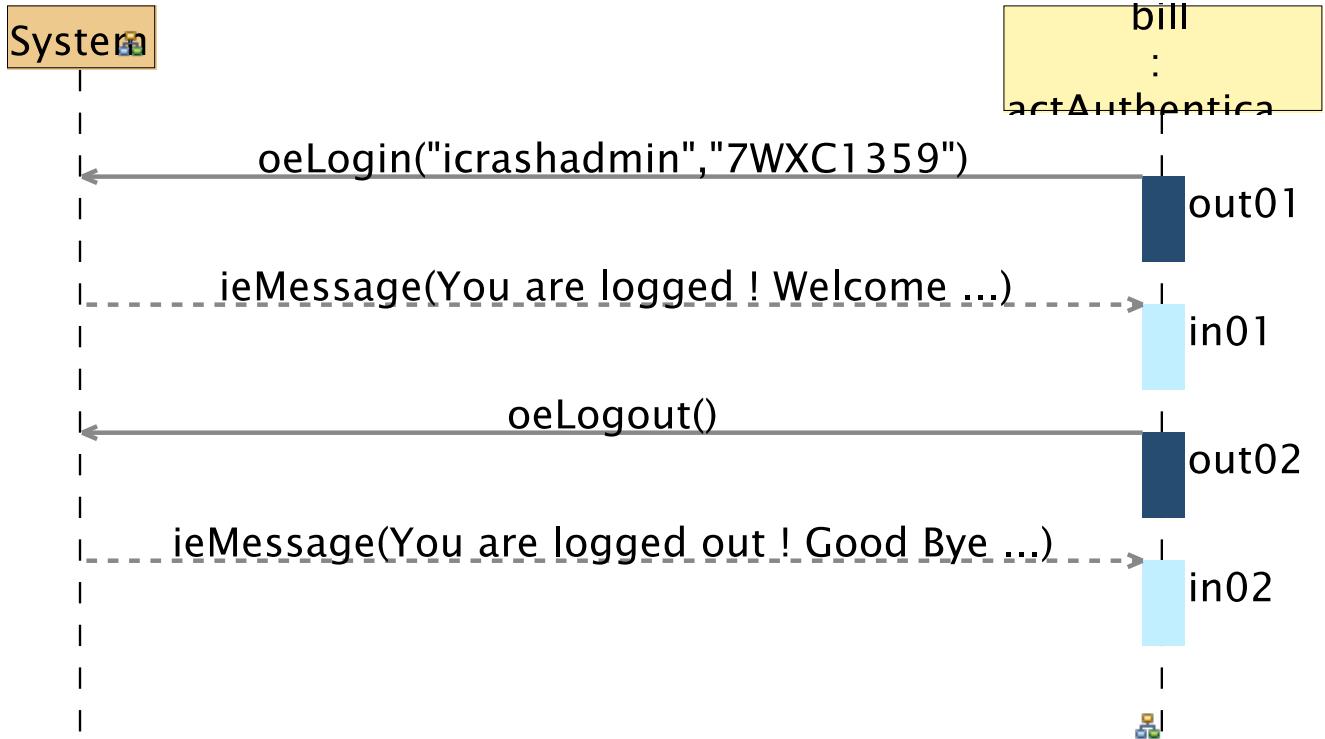


Figure 2.11: uciugSecurelyUseSystem user goal use case

# Chapter 3

## Environment Model

We provide below the view(s) defined for the **Messip** environment model (cf. [1]) of the system.

### 3.1 Local view 01

Figure 3.1 shows the local view giving the second part of the environment model of the system in term of its state class, actors with their input and output interfaces and all related associations.

Figure 3.1: Environment Model - Local View 01. environment model local view - Part 1.

### 3.2 Local view 02

Figure 3.2 shows the local view giving the second part the environment model of the system in term of its state class, actors with their input and output interfaces and all related associations.

Figure 3.2: Environment Model - Local View 02. environment model local view - Part 2.

### 3.3 Local view 03

Figure 3.3 shows the local view for the administrator actor and interfaces

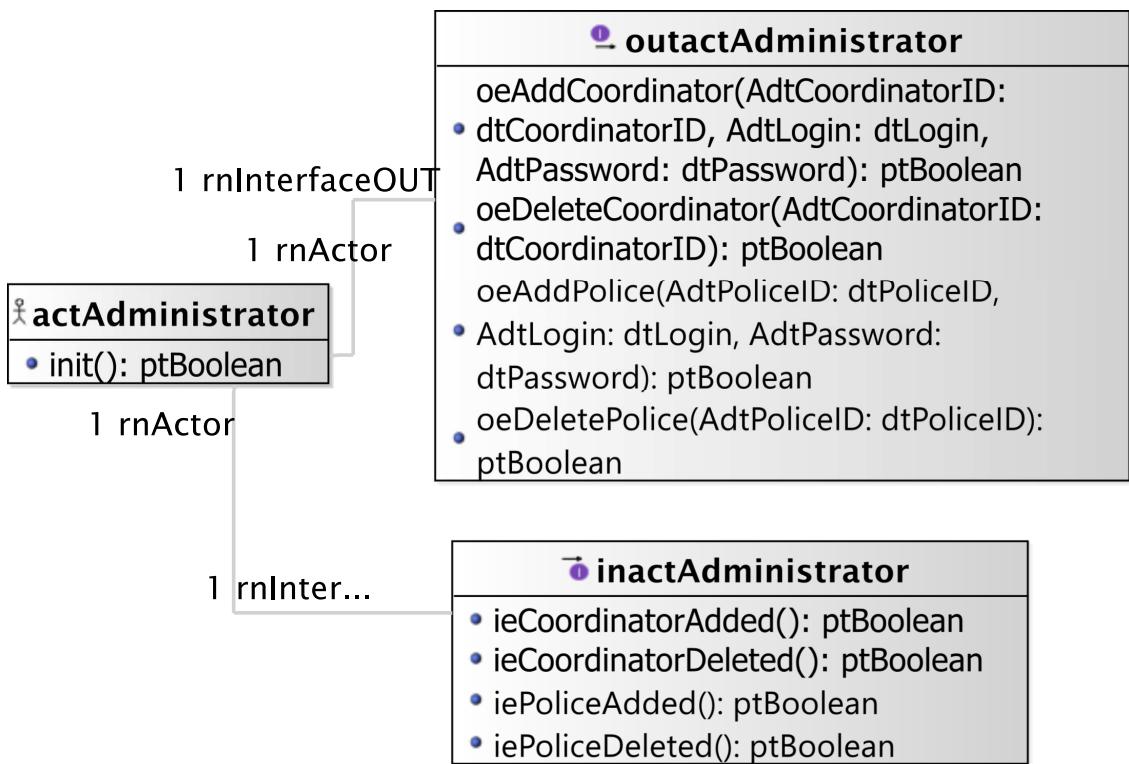


Figure 3.3: Environment Model - Local View 03. administrator actor environment model view.

### 3.4 Local view 04

Figure 3.4 shows the local view for the coordinator actor and interfaces

### 3.5 Local view 05

Figure 3.5 shows the local view for the authenticated actor and interfaces

### 3.6 Local view 08

Figure 3.6 shows the local view for the police actor and interfaces

### 3.7 Global view 01

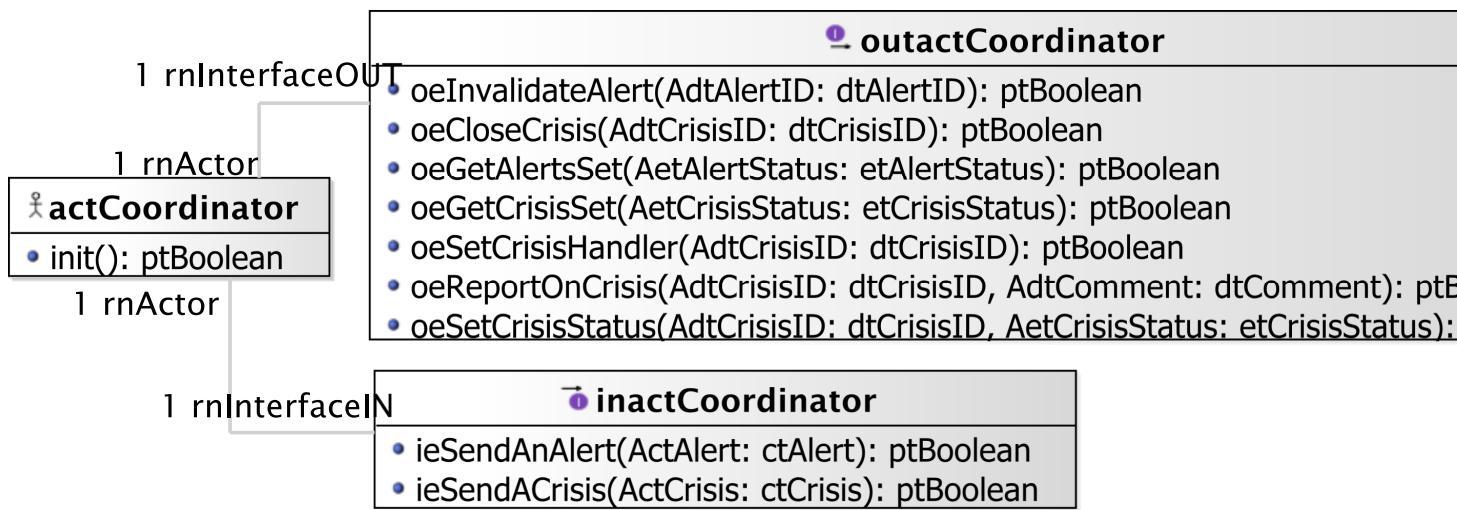


Figure 3.4: Environment Model - Local View 04. coordinator actor environment model view.

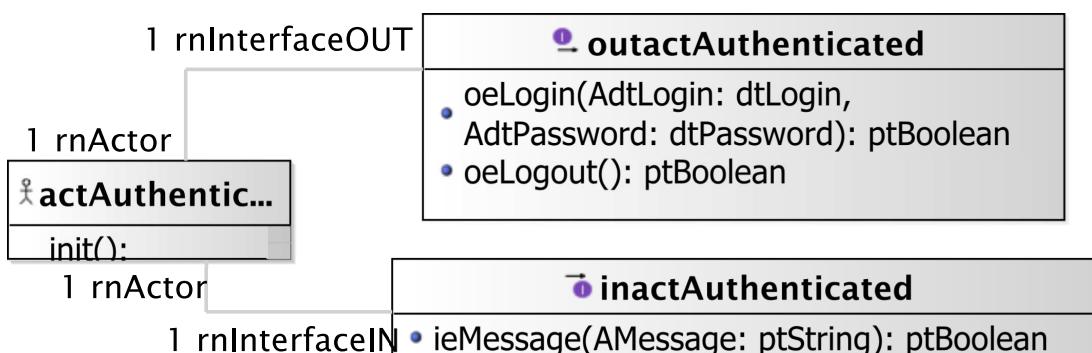


Figure 3.5: Environment Model - Local View 05. authenticated actor environment model local view.

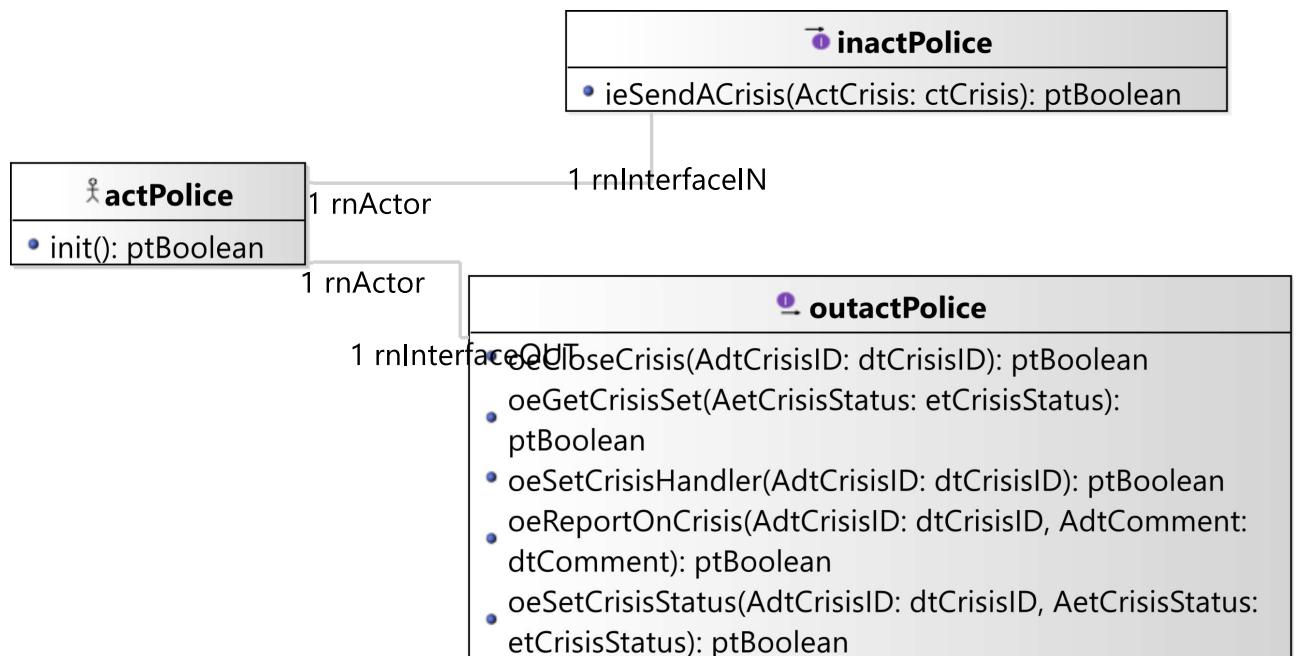


Figure 3.6: Environment Model - Local View 08. police actor environment model view.

Figure 3.7 shows a global view for all actors with their relationships with ctState

Figure 3.7: Environment Model - Global View 01. em-gv-01 environment model global view.

## 3.8 Actors and Interfaces Descriptions

We provide for the given views the description of the actors together with their associated input and output interface descriptions.

### 3.8.1 **actActivator** Actor

ACTOR
<i>actActivator</i>
represents a logical actor for time automatic message sending based on system's or environment status.
<i>Operations</i>
<i>init():ptBoolean</i>
<i>OutputInterfaces</i>
OUT 1 <b>[proactive]</b> <i>oeSollicitateCrisisHandling():ptBoolean</i> used to avoid crisis to stay too long in an not handled status.
OUT 2 <b>[proactive]</b> <i>oeSetClock(AcurrentClock:dtDateAndTime):ptBoolean</i> used to update the system's time

### 3.8.2 **actAdministrator** Actor

ACTOR
<i>actAdministrator</i>
represents an actor responsible of administration tasks for the <i>iCrash</i> system.
<i>Extends</i>
icrash.environment.actAuthenticated
<i>Operations</i>
<i>init():ptBoolean</i>
<i>OutputInterfaces</i>
OUT 1 <i>oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID,</i> <i>AdtLogin:dtLogin, AdtPassword:dtPassword):ptBoolean</i> sent to add a new coordinator in the system's post state and environment's post state.
OUT 2 <i>oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID):ptBoolean</i> <i>continues in next page ...</i>

**... Actor table continuation**

	sent to delete an existing coordinator in the system's post state and environment's post state.
<b><i>InputInterfaces</i></b>	
IN 1	<b>ieCoordinatorAdded():ptBoolean</b> its reception confirms the creation of the requested coordinator.
IN 2	<b>ieCoordinatorDeleted():ptBoolean</b> its reception confirms the deletion of the requested coordinator.

**3.8.3 actAuthenticated Actor**

<b>ACTOR</b>	
<b><i>actAuthenticated</i></b>	
abstract actor providing reusable input and output interfaces for actors that need to authenticate themselves.	
<b><i>Operations</i></b>	
<b>init():ptBoolean</b>	
<b><i>OutputInterfaces</i></b>	
OUT 1	<b>oeLogin(AdtLogin:dtLogin, AdtPassword:dtPassword):ptBoolean</b> sent to request authorization to request access secured system operations.
OUT 2	<b>oeLogout():ptBoolean</b> sent to end the secured access to specific system operations.
<b><i>InputInterfaces</i></b>	
IN 1	<b>ieMessage(AMessage:ptString):ptBoolean</b> allows for receiving general textual messages.

**3.8.4 actComCompany Actor**

<b>ACTOR</b>	
<b><i>actComCompany</i></b>	
represents the communication company stakeholder ensuring the input/ouput of textual messages with humans having communicaiton devices.	
<b><i>Operations</i></b>	
<b>init():ptBoolean</b>	
<b><i>OutputInterfaces</i></b>	
OUT 1	<b>oeAlert(AetHumanKind:etHumanKind, AdtDate:dtDate, AdtTime:dtTime, AdtPhoneNumber:dtPhoneNumber, AdtGPSLocation:dtGPSLocation, AdtComment:dtComment):ptBoolean</b> sent to alert of a potential crisis situation.
<b><i>InputInterfaces</i></b>	
IN 1	<b>ieSmsSend(AdtPhoneNumber:dtPhoneNumber, AdtSMS:dtSMS):ptBoolean</b> allows for receiving textual messages to be dispatched to the communication company customers having the provided phone number.

**3.8.5 actCoordinator Actor**

ACTOR
<i>actCoordinator</i>
represents actor responsible of handling one or several crisis for the <i>iCrash</i> system.
<i>Extends</i>
icrash.environment.actAuthenticated
<i>Operations</i>
<b><i>init():ptBoolean</i></b>
<i>OutputInterfaces</i>
OUT 1 <b><i>oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean</i></b> sent to indicate that an alert should be considered as closed.
OUT 2 <b><i>oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean</i></b> sent to indicate that a crisis should be considered as closed.
OUT 3 <b><i>oeGetAlertsSet(AetAlertStatus:etAlertStatus):ptBoolean</i></b> sent to request all the ctAlert instances having a specific status.
OUT 4 <b><i>oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean</i></b> sent to request all the ctCrisis instances having a specific status.
OUT 5 <b><i>oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean</i></b> sent to declare himself as been the handler of a crisis having the specified id.
OUT 6 <b><i>oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment):ptBoolean</i></b> sent to update the textual information available for a specific handled crisis.
OUT 7 <b><i>oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:etCrisisStatus):ptBoolean</i></b> sent to define the handling status of a specific crisis.
OUT 8 <b><i>oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:etCrisisType):ptBoolean</i></b> sent to define the gravity type of a specific crisis.
OUT 9 <b><i>oeValidateAlert(AdtAlertID:dtAlertID):ptBoolean</i></b> sent to indicate that a specific alert is not a fake.
<i>InputInterfaces</i>
IN 1 <b><i>ieSendAnAlert(ActAlert:ctAlert):ptBoolean</i></b> allows for receiving a requested ctAlert instance.
IN 2 <b><i>ieSendACrisis(ActCrisis:ctCrisis):ptBoolean</i></b> allows for receiving a requested ctCrisis instance.

### 3.8.6 **actMsrCreator** Actor

ACTOR
<i>actMsrCreator</i>
Represents the creator stakeholder in charge of state and environment initialization.
<i>Operations</i>
<b><i>init():ptBoolean</i></b>
<i>OutputInterfaces</i>
OUT 1 <b><i>oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger):ptBoolean</i></b> sent to request the initialization of the system's class instances and the environment actors instances.

### 3.8.7 **actPolice** Actor

<b>ACTOR</b>
<i>actPolice</i>
represents actor responsible of handling one or several crisis with huge type for the <i>iCrash</i> system.
<i>Extends</i>
icrash.environment.actAuthenticated
<i>Operations</i>
<b><i>init():ptBoolean</i></b>
<i>OutputInterfaces</i>
OUT 1 <b>oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean</b> sent to indicate that a crisis should be considered as closed.
OUT 2 <b>oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean</b> sent to request all the ctCrisis instances having a specific status.
OUT 3 <b>oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean</b> sent to declare himself as been the handler of a crisis having the specified id.
OUT 4 <b>oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment):ptBoolean</b> sent to update the textual information available for a specific handled crisis.
OUT 5 <b>oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:etCrisisStatus):ptBoolean</b> sent to define the handling status of a specific crisis.
<i>InputInterfaces</i>
IN 1 <b>ieSendACrisis(ActCrisis:ctCrisis):ptBoolean</b>

# Chapter 4

## Concept Model

### 4.1 PrimaryTypes-Classes

#### 4.1.1 Local view 01

Figure 4.1 shows the local view on all the primary types class types.

#### 4.1.2 Local view 02

Figure 4.2 shows the local view of the ctState primary type class type.

#### 4.1.3 Local view 03

Figure 4.3 shows the local view of the ctAlert primary type class type.

#### 4.1.4 Local view 04

Figure 4.4 shows the local view of the ctCrisis primary type class type.

#### 4.1.5 Global view 01

Figure 4.5 shows the global view on primary types class types showing the association(s) types with the actor classes of the environment model.

### 4.2 PrimaryTypes-Datatypes

#### 4.2.1 Local view 06

Figure 4.6 shows a local view on the GPSLocation primary types datatype types.

Figure 4.1: Concept Model - PrimaryTypes-Classes local view 01. Local view of all the primary types class types .

ctState	
④	nextValueForAlertID : dtInteger
④	nextValueForCrisisID : dtInteger
④	clock : dtDateAndTime
④	crisisReminderPeriod : dtSecond
④	maxCrisisReminderPeriod : dtSecond
④	vpLastReminder : dtDateAndTime
④	vpStarted : ptBoolean
	init(AnextValueForAlertID: dtInteger, AnextValueForCrisisID: dtInteger, Aclock:

Figure 4.2: Concept Model - PrimaryTypes-Classes local view 02. local view of the ctState primary type.

ctAlert	
④	id : dtAlertID
④	status : etAlertStatus
④	location : dtGPSLocation
④	instant : dtDateAndTime
④	comment : dtComment
	init(Aid: dtAlertID, Astatus: etAlertStatus, <del>Alocation: dtGPSLocation</del> , <del>Ainstant:</del>

Figure 4.3: Concept Model - PrimaryTypes-Classes local view 03. local view of the ctAlert primary type.

ctCrisis	
④	id : dtCrisisID
④	type : etCrisisType
④	status : etCrisisStatus
④	location : dtGPSLocation
④	instant : dtDateAndTime
④	comment : dtComment
	init(Aid: dtCrisisID, Atype: etCrisisType, Astatus: • etCrisisStatus, Alocation: dtGPSLocation, Ainstant: dtDateAndTime, Acomment: dtComment): ptBoolean

Figure 4.4: Concept Model - PrimaryTypes-Classes local view 04. local view of the ctCrisis primary type.

Figure 4.5: Concept Model - PrimaryTypes-Classes global view 01. Primary types class types global view - cm-pt-ct-gv-01 .

<b>dtGPSLocation</b>	
•	<b>latitude : dtLatitude</b>
•	<b>longitude : dtLongitude</b>
•	<b>is(): ptBoolean</b>
•	<b>isNearTo(AGPSLocation: dtGPSLocation): ptBoolean</b>

Figure 4.6: Concept Model - PrimaryTypes-Datatypes local view 06. local view of primary types datatype types - cm-pt-dt-lv-02-dtGPSLocation.

#### 4.2.2 Global view 01

Figure 4.7 shows a global view on the *iCrash* primary types datatype types.

### 4.3 SecondaryTypes-Datatypes

#### 4.3.1 Local view 01

Figure 4.8 shows the local view of the secondary types datatype types.

### 4.4 Concept Model Types Descriptions

This section provides the textual descriptions of all the types defined in the concept model and that can be part of the graphical views provided.

#### 4.4.1 Primary types - Class types descriptions

The table below is providing comments on the graphical views given for the class types of the primary types. Type logical operations are precisely specified in the operation model.

CLASSES	
<i>ctAdministrator</i>	
used to characterize internally the entity that is responsible of administrating the <i>iCrash</i> system.	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
operation	<b>init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean</b>
	used to initialize the current object as a new instance of the ctAdministrator type.
<i>ctAlert</i>	
Used to model crisis alerts sent by any human having communication capability using communication companies belonging to the system's environment	
attribute	<b>comment: dtComment</b>
	a textual description providing unstructured information on the alert.
attribute	<b>id: dtAlertID</b>
	the alert unique identification information.
attribute	<b>instant: dtDateAndTime</b>
	the date and time at which the alert notification has been sent.

***continues in next page ...***

**... Classes table continuation**

attribute	<b>location:</b> <code>dtGPSLocation</code>	the position of the alert provided by the space-based satellite navigation system used by the human using the communication company to inform the <i>iCrash</i> system of a crisis.
attribute	<b>status:</b> <code>etAlertStatus</code>	the alert validation status
operation		<code>init(Aid:dtAlertID, Astatus:etAlertStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment):ptBoolean</code> used to initialize the current object as a new instance of the ctAlert type.
operation	<b>isSentToCoordinator(AactCoordinator:actCoordinator):ptBoolean</b>	used to provide a given coordinator with current alert information.
<b>ctAuthenticated</b>		
	used to model system's representation about actors that need to authenticate to access some specific functionalities.	
attribute	<b>login:</b> <code>dtLogin</code>	an identifier for authentication.
attribute	<b>pwd:</b> <code>dtPassword</code>	a key for authentication.
attribute	<b>vpIsLogged:</b> <code>ptBoolean</code>	used to determine the access status.
operation	<b>init(Alogin:dtLogin, Apwd:dtPassword):ptBoolean</b>	used to initialize the current object as a new instance of the ctAuthenticated type.
<b>ctCoordinator</b>		
	used to model system's representation about the actors that have the responsibility to handle alerts and crisis.	
extends	icrash.concepts.primarytypes.classes.ctAuthenticated	
attribute	<b>id:</b> <code>dtCoordinatorID</code>	a unique identification information.
operation	<b>init(Aid:dtCoordinatorID, Alogin:dtLogin, Apwd:dtPassword):ptBoolean</b>	used to initialize the current object as a new instance of the ctCoordinator type.
<b>ctCrisis</b>		
	Used to model crisis that are inferred from the reception of at least one alert message. Crisis are entities that are handled by the <i>iCrash</i> system.	
attribute	<b>comment:</b> <code>dtComment</code>	a textual description providing unstructured information on the crisis handling.
attribute	<b>id:</b> <code>dtCrisisID</code>	the crisis unique identification information.
attribute	<b>instant:</b> <code>dtDateAndTime</code>	the date and time at which the first related alert notification has been sent.
attribute	<b>location:</b> <code>dtGPSLocation</code>	the position of the crisis equal to the one of the first alert received and associated to the crisis.
attribute	<b>status:</b> <code>etCrisisStatus</code>	the crisis handling status.
attribute	<b>type:</b> <code>etCrisisType</code>	an indication of the gravity of the crisis.

*continues in next page ...*

**... Classes table continuation**

operation	<b>handlingDelayPassed():ptBoolean</b> used to determine if the crisis stood too longly in a pending status since last reminder.
operation	<b>init(Aid:dtCrisisID, Atype:etCrisisType, Astatus:etCrisisStatus, Alocation:dtGPSLocation, Ainstant:dtDateAndTime, Acomment:dtComment):ptBoolean</b> used to initialize the current object as a new instance of the ctAlert type.
operation	<b>isAllocatedIfPossible():ptBoolean</b> used to allocate a crisis to a coordinator if any or to alert the administrator of crisis waiting to be handled.
operation	<b>isSentToCoordinator(AactCoordinator:actCoordinator):ptBoolean</b> used to provide a given coordinator with current crisis information.
operation	<b>maxHandlingDelayPassed():ptBoolean</b> used to determine if the crisis stood too longly in a pending status since its creation.
<b>ctHuman</b>	
	used to model system's representation about the indirect actors that has alerted of potential crisis.
attribute	<b>id: dtPhoneNumber</b> the number of the communication device used to send an alert to <i>iCrash</i> system.
attribute	<b>kind: etHumanKind</b> role with respect to the alert notified.
operation	<b>init(Aid:dtPhoneNumber, Akind:etHumanKind):ptBoolean</b> init: used to initialize the current object as a new instance of the ctHuman type.
operation	<b>isAcknowledged():ptBoolean</b>
<b>ctPolice</b>	
	used to model system's representation about the actors that have the responsibility to handle huge crisis.
extends	icrash.concepts.primarytypes.classes.ctAuthenticated
attribute	<b>id: dtPoliceID</b> a unique identification information.
operation	<b>init(Aid:dtPoliceID, Alogin:dtLogin, Apwd:dtPassword):ptBoolean</b> used to initialize the current object as a new instance of the ctPolice type.
<b>ctState</b>	
	used to model the system. Each system specified using <b>Mess1P</b> must include a ctState class for which there is only one instance at any state of the abstract machine after creation.
attribute	<b>clock: dtDateAndTime</b> used to represent the system local time.
attribute	<b>crisisReminderPeriod: dtSecond</b> used to define the delay between two reminders after which a reminder must be sent to the administrator and to the known coordinators to encourage them to handle the crisis.
attribute	<b>maxCrisisReminderPeriod: dtSecond</b> used to define the maximum delay after which the crisis is randomly allocated to a coordinator if any or an alert message is sent to the administrator in order to encourage him to add coordinators.
attribute	<b>nextValueForAlertID: dtInteger</b> nextValueForAlertID: dtInteger: used to associate each alert declared with a unique identification value.
attribute	<b>nextValueForCrisisID: dtInteger</b>

*continues in next page ...*

**... Classes table continuation**

attribute	used to associate each crisis declared with a unique identification value.
	<b>vpLastReminder: dtDateAndTime</b> date and time of the last reminder.
attribute	<b>vpStarted: ptBoolean</b>
operation	used to avoid reacting to an actor message if the system is not started (i.e. oeCreateSystemAndEnvironment not executed). <b>init(AnextValueForAlertID:dtInteger, AnextValueForCrisisID:dtInteger, Aclock:dtDateAndTime, AcrisisReminderPeriod:dtSecond, AmaxCrisisReminderPeriod:dtSecond, AvpLastReminder:dtDateAndTime, AvpStarted:ptBoolean):ptBoolean</b> used to initialize the current object as a new instance of the ctState type.

**4.4.2 Primary types - Datatypes types descriptions**

The table below is providing comments on the graphical views given for the datatype types of the primary types.

DATATYPES	
<b>dtAlertID</b>	A string used to identify alerts.
operation	<b>is():ptBoolean</b> used to determine which strings are considered as valid alert identifiers.
<b>dtComment</b>	a datatype made of a string value used to receive, store and send textual information about crisis and alerts.
operation	<b>is():ptBoolean</b> used to determine which strings are considered as valid comments.
<b>dtCoordinatorID</b>	A string used to identify coordinators.
operation	<b>is():ptBoolean</b> used to determine which strings are considered as valid coordinators identifiers.
<b>dtCrisisID</b>	A string used to identify crisis.
operation	<b>is():ptBoolean</b> used to determine which strings are considered as valid crisis identifiers.
<b>dtGPSLocation</b>	used to define coordinates of geographical positions on earth. It is defined a couple made of a latitude and a longitude.
attribute	<b>latitude: dtLatitude</b> for the latitude part of the coordinate.
attribute	<b>longitude: dtLongitude</b> for the longitude part of the coordinate.
operation	<b>is():ptBoolean</b> used to determine which couples are considered as valid dtGPSLocation values.
operation	<b>isNearTo(AGPSLocation:dtGPSLocation):ptBoolean</b> used to determine if locations are considered enough close to be treated as equivalent in the application domain context.
<b>dtLatitude</b>	

*continues in next page ...*

**... Datatypes table continuation**

used to define a latitude value of a geographical positions on earth.
operation <b>is():ptBoolean</b> used to determine which strings are considered as valid dtLatitude.
<b>dtLogin</b> a login string used to authentify an <i>iCrash</i> user
operation <b>is():ptBoolean</b> used to determine which strings are considered as valid dtLogin.
<b>dtLongitude</b> used to define a longitude value of a geographical positions on earth.
operation <b>is():ptBoolean</b> used to determine which strings are considered as valid dtLongitude.
<b>dtPassword</b> a password string used to authentify an <i>iCrash</i> user
operation <b>is():ptBoolean</b> used to determine which strings are considered as valid dtPassword.
<b>dtPhoneNumber</b> a string used to store the phone number from the human declaring the crisis or the alert.
operation <b>is():ptBoolean</b> used to determine which strings are considered as valid dtPhoneNumber.
<b>dtPoliceID</b> A string used to identify polices.
operation <b>is():ptBoolean</b> used to determine which strings are considered as valid polices identifiers.

**ENUMERATIONS**

<b>etAlertStatus</b> this type is used to indicate the different validation status of an alert.
operation <b>is():ptBoolean</b> used to determine which litteral belongs to the enumeration.
<b>etCrisisStatus</b> this type is used to indicate the different handling status of a crisis.
operation <b>is():ptBoolean</b> used to determine which litteral belongs to the enumeration.
<b>etCrisisType</b> this type is used to indicate the different types of a crisis.
operation <b>is():ptBoolean</b> used to determine which litteral belongs to the enumeration.
<b>etHumanKind</b> this type is used to indicate the kind of human that informs about a car crash crisis.
operation <b>is():ptBoolean</b> used to determine which litteral belongs to the enumeration.

**4.4.3 Primary types - Association types descriptions**

The table below is providing comments on the association types of the primary types.

Figure 4.7: Concept Model - PrimaryTypes-Datatypes global view 01. global view of primary types datatype types - cm-pt-dt-gv-01 .

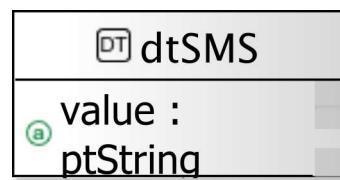


Figure 4.8: Concept Model - SecondaryTypes-Datatypes local view 01. Local view of the secondary types datatype types.

<b>UNDIRECTED ASSOCIATIONS</b>	
<b><i>assctAlertctCrisis</i></b>	a crisis is related to one or more alerts as the alerts judged to concern all the same crisis due to their location. An alert alerts exactly one crisis.
<b><i>assctAlertctHuman</i></b>	alerts are notified by human through the communication company. We need to keep an internal representation of those human to allow for communication of alert handling.
<b><i>assctAuthenticatedactAuthenticated</i></b>	mainly used to determine if the login request of an authenticated actor can be granted based on the given credentials and the registered ones.
<b><i>assctCoordinatoractCoordinator</i></b>	frequent messages must be sent to coordinator especially in relation to crisis they handle.
<b><i>assctCrisisctCoordinator</i></b>	at any point in time we need to know if a coordinator is handling existing crisis or not.
<b><i>assctCrisisctPolice</i></b>	at any point in time we need to know if a police is handling existing crisis or not.
<b><i>assctHumanactComCompany</i></b>	in order to communicate with humans who informed about potential crisis, we need to record the communication company to use to send them messages.
<b><i>assctPoliceactPolice</i></b>	frequent messages must be sent to police especially in relation to crisis they handle.

#### 4.4.4 Primary types - Aggregation types descriptions

There are no aggregation types for the primary types.

##### 4.4.4.1 Primary types - Composition types descriptions

There are no composition types for the primary types.

#### 4.4.5 Secondary types - Class types descriptions

There are no elements in this category in the system analysed.

#### 4.4.6 Secondary types - Datatypes types descriptions

The table below is providing comments on the graphical views given for the datatype types of the secondary types.

<b>DATATYPES</b>	
<b><i>dtSMS</i></b>	a datatype made of a string value used to send textual information to human mobile devices.
attribute	<b>value: ptString</b> the textual information.
operation	<b>is():ptBoolean</b> used to determine which strings are considered as valid comments.

#### 4.4.7 Secondary types - Association types descriptions

There are no association types for the secondary types.

**4.4.8 Secondary types - Aggregation types descriptions**

There are no aggregation types for the secondary types.

**4.4.9 Secondary types - Composition types descriptions**

There are no composition types for the secondary types.

# Chapter 5

## Operation Model

This section contains the operation schemes of each operation defined in either an actor, its output interface, in a primary or secondary type (class, datatype or enumeration types). The **Messip** OCL code listing is joined to the comment table.

### 5.1 Environment - Out Interface Operation Scheme for actActivator

#### 5.1.1 Operation Model for oeSetClock

The `oeSetClock` operation has the following properties:

OPERATION	
<b><i>oeSetClock[proactive]</i></b>	
An active message used to statically set the date and time information in the system's state.	
<b>Parameters</b>	
1	<b>AcurrentClock: dtDateAndTime</b> the date and time to be considered as the actual one.
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is supposed to be created and initialized and the provided date and time value is greater than the one known by the system.
<b>Pre-Condition (functional)</b>	
PreF 1	none
<b>Post-Condition (functional)</b>	
PostF 1	the ctState instance post-state is updated to have its clock attribute equal to the given date and time.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.1 provides the **Messip** (MCL-oriented) specification of the operation.

```
1
2 /* Pre Protocol:*/
3 preP{let TheSystem: ctState in
```

```

4  let AvpStarted: ptBoolean in
5
6 /* PreP01 */
7 self.rnActor.rnSystem = TheSystem
8 and self.rnActor.rnSystem.vpStarted = AvpStarted
9 and AvpStarted = true
10 and TheSystem.clock.lt(AcurrentClock)}
11
12 /* Pre Functional:*/
13 preF{true}
14
15 /* Post Functional:*/
16 postF{let TheSystem: ctState in
17 self.rnActor.rnSystem = TheSystem
18
19 /* PostF01 */
20 and TheSystem@post.clock = AcurrentClock}
21
22 /* Post Protocol:*/
23 postP{ true}

```

Listing 5.1: **Messir** (MCL-oriented) specification of the operation *oeSetClock*.

The listing 5.2 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%-----%
6msrop(outactActivator,
7    oeSetClock,
8    [preProtocol,Self,
9     AcurrentClock
10    ],
11    []):-!
12/* Pre Protocol:*/
13/* PreP01 */
14 msrVar(ctState,TheSystem),
15 msrVar(ptBoolean,AvpStarted),
16
17 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
18
19 msrNav([Self],[rnActor,rnSystem,vpStarted],[AvpStarted]),
20 AvpStarted = [ptBoolean,true],
21
22 msrNav([TheSystem],
23         [clock,lt,[AcurrentClock]],
24         [[ptBoolean,true]])
25 .
26
27msrop(outactActivator,
28    oeSetClock,
29    [preFunctional,Self,
30     AcurrentClock
31    ],
32    []):-!
33/* Pre Functional:*/
34/* PreF01 */
35true.
36
37msrop(outactActivator,
38    oeSetClock,
39    [post,Self,
40     AcurrentClock
41    ],

```

```

42      []):-  

43  

44 msrVar(ctState,TheSystem),  

45  

46 /* Post Functional:*/  

47  

48 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  

49  

50 /* PostF01 */  

51 msrNav([TheSystem],  

52         [msmAtPost,clock],  

53         [AcurrentClock]),  

54  

55 /* Post Protocol:*/  

56 /* PostP01 */  

57 true  

58 .

```

Listing 5.2: **Messip** (Prolog-oriented) implementation of the operation *oeSetClock*.

### 5.1.2 Operation Model for oeSollicitateCrisisHandling

The *oeSollicitateCrisisHandling* operation has the following properties:

OPERATION
<i>oeSollicitateCrisisHandling[proactive]</i>
A proactive message (message of a pro-active actor with no parameter triggered automatically if the pre protocol condition is true) used to avoid crisis to stay too long in an not handled status.
<i>Return type</i>
ptBoolean
<i>Pre-Condition (protocol)</i>
PreP 1 the system is started
PreP 2 there exist some crisis that are in pending status and for which the duration between the current ctState clock information and the last reminder is greater than the crisis reminder period duration.
<i>Pre-Condition (functional)</i>
PreF 1 none
<i>Post-Condition (functional)</i>
PostF 1 if there exist coordinators and crisis who stood in a not handled status more than the maximum allowed time then those crisis are randomly allocated to the existing coordinators.
PostF 2 for all other crisis who stood too longly in a not handled status but not more than the maximum delay allowed then a reminder message is sent to the administrator and all coordinator actors of the environment to sollicitate handling of those crisis.
<i>Post-Condition (protocol)</i>
PostP 1 the value of the last reminder known by the system at post state is the system's clock value.

The listing 5.3 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  

2 /* Pre Protocol:*/  

3 prep{let TheSystem: ctState in  

4 let AvpStarted: ptBoolean in  

5 let ColctCrisisToHandle:

```

```

6      Bag(ctCrisis) in
7
8  self.rnActor.rnSystem = TheSystem
9
10 /* PreP01 */
11 and TheSystem.vpStarted
12
13 /* PreP02 */
14 and TheSystem.rnctCrisis->select(handlingDelayPassed())
15     = ColctCrisisToHandle
16 and ColctCrisisToHandle->size().geq(1)
17
18 /* Pre Functional:*/
19 preF{true}
20
21 /* Post Functional:*/
22 postF{let TheSystem: ctState in
23 let AMesssageForCrisisHandlers: dtComment in
24 let ColctCrisisToAllocateIfPossible:Bag(ctCrisis) in
25
26 self.rnActor.rnSystem = TheSystem
27 /* PostF01 */
28 and TheSystem.rnctCrisis->select(maxHandlingDelayPassed())
29     = ColctCrisisToAllocateIfPossible
30 and ColctCrisisToAllocateIfPossible->forAll(isAllocatedIfPossible())
31
32 /* PostF02 */
33 and TheSystem.rnctCrisis->select(handlingDelayPassed())
34     = ColctCrisisToHandle
35
36 and ColctCrisisToHandle->msrColSubtract(ColctCrisisToAllocateIfPossible)
37     = ColctCrisisToRemind
38
39 and if (ColctCrisisToRemind->size().geq(1))
40     then (AMesssageForCrisisHandlers.value
41             ='There are alerts pending since more than the defined delay. Please REACT !'
42             and TheSystem.rnactAdministrator.
43             rnInterfaceIN^ieMessage(AMesssageForCrisisHandlers)
44             and TheSystem.rnactCoordinator
45             ->forAll(rnInterfaceIN^ieMessage(AMesssageForCrisisHandlers)))
46
47 else true
48 endif}
49
50 /* Post Protocol:*/
51 postP{ let TheSystem: ctState in
52 let TheClock: dtDateAndTime in
53
54 self.rnActor.rnSystem = TheSystem
55 and TheSystem.clock = TheClock
56 and TheSystem@post.vpLastReminder = TheClock}

```

Listing 5.3: **Messsir** (MCL-oriented) specification of the operation *oeSollicitateCrisisHandling*.

The listing 5.4 provides the **Messsir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6
7msrop(outactActivator,
8    oeSollicitateCrisisHandling,
9    [preProtocol,Self
10    ],

```

```

11      []):-  

12 /* Pre Protocol:- */  

13 msrVar(ctState,TheSystem),  

14 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  

15  

16 msrVarCol(ctCrisis,_,ColctCrisisToHandle),  

17  

18 /* PreP01 */  

19 msrNav([TheSystem],  

20        [vpStarted],  

21        [[ptBoolean,true]]),  

22  

23 /* PreP02 */  

24 msrNav([TheSystem],  

25        [rnctCrisis,msrSelect,  

26         handlingDelayPassed,[]]  

27        ],  

28        ColctCrisisToHandle),  

29  

30 msrNav(ColctCrisisToHandle,  

31        [msrSize,geq,[[ptInteger,1]]],  

32        [[ptBoolean,true]]))  

33.  

34  

35 msrop(outactActivator,  

36        oeSolicitCrisisHandling,  

37        [preFunctional,Self  

38        ],  

39        []):-  

40 /* Pre Functional:- */  

41 /* PreF01 */  

42 true.  

43  

44 msrop(outactActivator,  

45        oeSolicitCrisisHandling,  

46        [post,Self  

47        ],  

48        []):-  

49  

50 msrVar(ctState,TheSystem),  

51 msrVar(dtComment,AMessageForCrisisHandlers),  

52 msrVar(dtDateAndTime, TheClock),  

53 msrVarCol(ctCrisis,_,ColctCrisisToAllocateIfPossible),  

54  

55 /* Post Functional:- */  

56 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  

57  

58 /* PostF01 */  

59 msrNav([TheSystem],  

60        [rnctCrisis,msrSelect,  

61         maxHandlingDelayPassed,[]]  

62        ],  

63        ColctCrisisToAllocateIfPossible),  

64  

65 msrNav(ColctCrisisToAllocateIfPossible,  

66        [msrForAll,isAllocatedIfPossible,[],  

67        [[ptBoolean,true]]),  

68  

69 /* PostF02 */  

70 msrNav([TheSystem],  

71        [rnctCrisis,msrSelect,  

72         handlingDelayPassed,[]]  

73        ],  

74        ColctCrisisToHandle),  

75  

76 msrNav(ColctCrisisToHandle,  

77        [msrColSubtract,[ColctCrisisToAllocateIfPossible]  

78        ],  

79        ColctCrisisToRemind),  

80

```

```

81  (msrNav(ColctCrisisToRemind,
82      [msrSize,geq,[[ptInteger,1]]],
83      [[ptBoolean,true]]))
84 -> (msrNav([AMessageForCrisisHandlers],
85     [value],
86     [[ptString,'There are alerts pending since more than the defined delay. Please REACT !']]),
87
88  msrNav([TheSystem],
89      [rnactAdministrator,rnInterfaceIN,
90      ieMessage,[AMessageForCrisisHandlers]
91      ],
92      [[ptBoolean,true]]),
93
94  msrNav([TheSystem],
95      [rnactCoordinator,msrForAll,rnInterfaceIN,
96      ieMessage,[AMessageForCrisisHandlers]
97      ],
98      [[ptBoolean,true]]))
99  )
100 ; true
101 ),
102
103 /* Post Protocol:*/
104 /* PostP01 */
105 msrNav([TheSystem],
106     [clock],
107     [TheClock]),
108
109 msrNav([TheSystem],
110     [msmAtPost,vpLastReminder],
111     [TheClock])
112 .

```

Listing 5.4: **Messip** (Prolog-oriented) implementation of the operation *oeSollicitateCrisisHandling*.

Figure 5.1 shows concept model elements in the scope of the *oeSollicitateCrisisHandling* operation

## 5.2 Environment - Out Interface Operation Scheme for actAdministrator

### 5.2.1 Operation Model for *oeAddCoordinator*

The *oeAddCoordinator* operation has the following properties:

<b>OPERATION</b>	
<i>oeAddCoordinator</i>	
sent to add a new coordinator in the system's post state and environment's post state.	
<i>Parameters</i>	
1	<b>AdtCoordinatorID: dtCoordinatorID</b> used to initialize the id field
2	<b>AdtLogin: dtLogin</b> used to initialize the login field
3	<b>AdtPassword: dtPassword</b> used to initialize the password field
<i>Return type</i>	
ptBoolean	
<i>Pre-Condition (protocol)</i>	

*continues in next page ...*

*... Operation table continuation*

PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	it is supposed that there cannot exist a ctCoordinator instance with the same <code>id</code> attribute as the one the administrator wants to delete.
<b>Post-Condition (functional)</b>	
PostF 1	the environment has a new instance of coordinator actor allowing for input/output message communication with the system.
PostF 2	the system's state has a new instance of ctCoordinator initialized with the given values.
PostF 3	the new actor instance and ctCoordinator instance are related.
PostF 4	the new actor instance and ctCoordinator instance are related according to the authenticated association.
PostF 5	the administrator actor is informed about the satisfaction of its request.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.5 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Pre Protocol:*/
2  prep{let TheSystem: ctState in
3    let TheActor:actAdministrator in
4
5    self.rnActor.rnSystem = TheSystem
6    and self.rnActor = TheActor
7
8
9  /* PreP01 */
10 and TheSystem.vpStarted = true
11 /* PreP02 */
12 and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Pre Functional:*/
15 pref{let TheSystem: ctState in
16  let TheActor:actAdministrator in
17  let ColctCoordinators:Bag(ctCoordinator) in
18
19  self.rnActor.rnSystem = TheSystem
20  and self.rnActor = TheActor
21 /* PreF01 */
22  and TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
23  = ColctCoordinators
24  and ColctCoordinators->isEmpty() = true}
25
26 /* Post Functional:*/
27 postF{let TheSystem: ctState in
28  let TheactCoordinator:actCoordinator in
29  let ThectCoordinator:ctCoordinator in
30  self.rnActor.rnSystem = TheSystem
31  and self.rnActor = TheActor
32 /* PostF01 */
33  TheactCoordinator.init()
34 /* PostF02 */
35  and ThectCoordinator.init(AdtCoordinatorID,AdtLogin,AdtPassword)
36
37 /* PostF03 */
38  and TheactCoordinator@post.rnctCoordinator = ThectCoordinator

```



Figure 5.1: oeSollicitateCrisisHandling operation scope

## 5.2. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTADMINISTRATOR65

```

39  /* PostF04 */
40  and ThectCoordinator@post.rnactAuthenticated = TheactCoordinator
41
42
43  /* PostF05 */
44  and TheActor.rnInterfaceIN^ieCoordinatorAdded()
45
46  /* Post Protocol:*/
47  postP{ true}

```

Listing 5.5: **Messir** (MCL-oriented) specification of the operation *oeAddCoordinator*.

The listing 5.6 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactAdministrator,
7    oeAddCoordinator,
8    [preProtocol,Self,
9     AdtCoordinatorID,
10    AdtLogin,
11    AdtPassword
12   ],
13  []):-!
14/* Pre Protocol:*/
15 msrVar(ctState,TheSystem),
16 msrVar(actAdministrator,TheActor),
17 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
18 msrNav([Self],[rnActor],[TheActor]),
19
20/* PreP01 */
21 msrNav([TheSystem],
22     [vpStarted],
23     [[ptBoolean,true]]),
24
25/* PreP02 */
26 msrNav([TheActor],
27     [rnctAuthenticated,vpIsLogged],
28     [[ptBoolean,true]])
29
30 .
31
32msrop(outactAdministrator,
33    oeAddCoordinator,
34    [preFunctional,Self,
35     AdtCoordinatorID,
36     AdtLogin,
37     AdtPassword
38    ],
39  []):-!
40/* Pre Functional:*/
41 msrVar(ctState,TheSystem),
42 msrVar(actAdministrator,TheActor),
43 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
44 msrNav([Self],[rnActor],[TheActor]),
45/* PreF01 */
46 msrNav([TheSystem],
47     [rnctCoordinator,
48      msrSelect,id,eq,[AdtCoordinatorID]],
49     ColctCoordinators),
50 msrNav(ColctCoordinators,
51     [msrIsEmpty],
52     [[ptBoolean,true]])

```

```

53 .
54
55msrop(outactAdministrator,
56    oeAddCoordinator,
57    [post,Self,
58     AdtCoordinatorID,
59     AdtLogin,
60     AdtPassword
61     ],
62     []):-!
63
64 /* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actAdministrator,TheActor),
67 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
68 msrNav([Self],[rnActor],[TheActor]),
69
70 msrVar(actCoordinator,TheactCoordinator),
71 msrVar(ctCoordinator,ThectCoordinator),
72
73 /* PostF01 */
74 msrNav([TheactCoordinator],
75     [init,[],  
[[ptBoolean,true]]),
76
77
78 /* PostF02 */
79 msrNav([ThectCoordinator],
80     [init,[AdtCoordinatorID,AdtLogin,AdtPassword]],  
[[ptBoolean,true]]),
81
82
83 /* PostF03 */
84 msrNav([TheactCoordinator],
85     [msmAtPost,rnctCoordinator],  
[ThectCoordinator]),
86
87
88 /* PostF04 */
89 msrNav([ThectCoordinator],
90     [msmAtPost,rnactAuthenticated],  
[TheactCoordinator]),
91
92
93 /* PostF05 */
94 msrNav([TheActor],
95     [rnInterfaceIN,  
ieCoordinatorAdded,[],  
[[ptBoolean,true]]),
96
97
98 /* Post Protocol:*/
99 /* PostP01 */
100 /* PostP01 */
101 true
102 .

```

Listing 5.6: **Messir** (Prolog-oriented) implementation of the operation *oeAddCoordinator*.

### 5.2.2 Operation Model for *oeAddPolice*

The *oeAddPolice* operation has the following properties:

<b>OPERATION</b>	
<b>oeAddPolice</b>	
sent to add a new police in the system's post state and environment's post state.	
<b>Parameters</b>	
1	<b>AdtPoliceID: dtPoliceID</b> used to initialize the id field
2	<b>AdtLogin: dtLogin</b>

*continues in next page ...*

**... Operation table continuation**

3	used to initialize the login field <b>AdtPassword: dtPassword</b> used to initialize the password field
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	it is supposed that there cannot exist a ctPolice instance with the same id attribute as the one the administrator wants to delete.
<b>Post-Condition (functional)</b>	
PostF 1	the environment has a new instance of police actor allowing for input/output message communication with the system.
PostF 2	the system's state has a new instance of ctPolice initialized with the given values.
PostF 3	the new actor instance and ctPolice instance are related.
PostF 4	the new actor instance and ctPolice instance are related according to the authenticated association.
PostF 5	the administrator actor is informed about the satisfaction of its request.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.7 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Pre Protocol:*/
2  prep{let TheSystem: ctState in
3    let TheActor:actAdministrator in
4
5
6    self.rnActor.rnSystem = TheSystem
7    and self.rnActor = TheActor
8
9  /* PreP01 */
10 and TheSystem.vpStarted = true
11 /* Prep02 */
12 and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Pre Functional:*/
15 prep{let TheSystem: ctState in
16  let TheActor:actAdministrator in
17  let ColctPolices:Bag(ctPolice) in
18
19  self.rnActor.rnSystem = TheSystem
20  and self.rnActor = TheActor
21 /* Prep01 */
22  and TheSystem.rnctPolice->select(id.eq(AdtPoliceID))
23    = ColctPolices
24  and ColctPolices->isEmpty() = true}
25
26 /* Post Functional:*/
27 postF{let TheSystem: ctState in
28  let TheactPolice:actPolice in
29  let ThectPolice:ctPolice in

```

```

30  self.rnActor.rnSystem = TheSystem
31  and self.rnActor = TheActor
32  /* PostF01 */
33  TheactPolice.init()
34  /* PostF02 */
35  and ThectPolice.init(AdtPoliceID,AdtLogin,AdtPassword)
36
37  /* PostF03 */
38  and TheactPolice@post.rnctPolice = ThectPolice
39
40  /* PostF04 */
41  and ThectPolice@post.rnactAuthenticated = TheactPolice
42
43  /* PostF05 */
44  and TheActor.rnInterfaceIN^iePoliceAdded())
45
46  /* Post Protocol:*/
47  postP{ true}

```

Listing 5.7: **Messip** (MCL-oriented) specification of the operation *oeAddPolice*.

### 5.2.3 Operation Model for oeDeleteCoordinator

The *oeDeleteCoordinator* operation has the following properties:

<b>OPERATION</b>	
<i>oeDeleteCoordinator</i>	
sent to delete an existing coordinator in the system's post state and environment's post state.	
<b>Parameters</b>	
1	<b>AdtCoordinatorID: dtCoordinatorID</b> used for ctCoordinator instance retrieval
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
<b>Pre-Condition (functional)</b>	
Pref 1	it is supposed that there exist one ctCoordinator instance with the same id attribute than the one the administrator wants to create.
<b>Post-Condition (functional)</b>	
PostF 1	the ctCoordinator class instance having the required id do not belong anymore to the post state as well as is related actCoordinator actor instance.
PostF 2	the administrator actor is informed about the satisfaction of its request.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.8 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in

```

```

4  let TheActor:actAdministrator in
5
6  self.rnActor.rnSystem = TheSystem
7  and self.rnActor = TheActor
8
9  /* PreP01 */
10 and TheSystem.vpStarted = true
11 /* PreP02 */
12 and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Pre Functional:*/
15 preF{let TheSystem: ctState in
16 let TheActor:actAdministrator in
17
18 self.rnActor.rnSystem = TheSystem
19 and self.rnActor = TheActor
20 /* PreF01 */
21 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
22 = ColctCoordinators
23 and ColctCoordinators->size().eq(1)}
24
25 /* Post Functional:*/
26 postF{let TheSystem: ctState in
27 let TheActor:actAdministrator in
28 let ThectCoordinator:ctCoordinator in
29 self.rnActor.rnSystem = TheSystem
30 and self.rnActor = TheActor
31 /* PostF01 */
32 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
33 = ThectCoordinator
34 and ThectCoordinator.rnactCoordinator->forall(msrIsKilled)
35 and ThectCoordinator.msrIsKilled
36
37 /* PostF02 */
38 and TheActor.rnInterfaceIN^ieCoordinatorDeleted()
39
40 /* Post Protocol:*/
41 /* PostP01 */
42 and true}
43
44 /* Post Protocol:*/
45 postP{ true}

```

Listing 5.8: **Messip** (MCL-oriented) specification of the operation *oeDeleteCoordinator*.

The listing 5.9 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactAdministrator,
7 oeDeleteCoordinator,
8 [preProtocol,Self,
9 AdtCoordinatorID
10 ],
11 []):-!
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actAdministrator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],

```

```

20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]]),
26 .
27
28msrop(outactAdministrator,
29     oeDeleteCoordinator,
30     [preFunctional,Self,
31      AdtCoordinatorID
32      ],
33     []):-.
34 /* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actAdministrator,TheActor),
37 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
38 msrNav([Self],[rnActor],[TheActor]),
39
40 /* PreF01 */
41 msrNav([TheSystem],
42     [rnctCoordinator,
43      msrSelect,id,eq,[AdtCoordinatorID]],
44     ColctCoordinators),
45
46 msrNav(ColctCoordinators,
47     [msrSize,eq,[[ptInteger,1]]],
48     [[ptBoolean,true]]).
49
50msrop(outactAdministrator,
51     oeDeleteCoordinator,
52     [post,Self,
53      AdtCoordinatorID
54      ],
55     []):-.
56
57 /* Post Functional:*/
58 msrVar(ctState,TheSystem),
59 msrVar(actAdministrator,TheActor),
60 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
61 msrNav([Self],[rnActor],[TheActor]),
62
63 /* PostF01 */
64 msrNav([TheSystem],
65     [rnctCoordinator,
66      msrSelect,id,eq,[AdtCoordinatorID]],
67     [ThectCoordinator]),
68
69 msrNav([ThectCoordinator],
70     [rnactCoordinator,msrForAll,msrIsKilled,
71      [[ptBoolean,true]]),
72
73 msrNav([ThectCoordinator],
74     [msrIsKilled,
75      [[ptBoolean,true]]),
76
77 /* PostF02 */
78 msrNav([TheActor],
79     [rnInterfaceIN,
80      ieCoordinatorDeleted,[]],
81     []),
82     [[ptBoolean,true]]),
83
84 /* Post Protocol:*/
85 /* PostP01 */
86 true
87 .

```

Listing 5.9: **Messir** (Prolog-oriented) implementation of the operation *oeDeleteCoordinator*.

### 5.2.4 Operation Model for oeDeletePolice

The `oeDeletePolice` operation has the following properties:

OPERATION
<b><i>oeDeletePolice</i></b>
sent to delete an existing police in the system's post state and environment's post state.
<b>Parameters</b>
1 <b>AdtPoliceID: dtPoliceID</b> used for ctPolice instance retrieval
<b>Return type</b>
ptBoolean
<b>Pre-Condition (protocol)</b>
PreP 1    the system is started PreP 2    the actor logged previously and did not log out ! (i.e. the associated ctAdministrator instance is considered logged)
<b>Pre-Condition (functional)</b>
PreF 1    it is supposed that there exist one ctPolice instance with the same <code>id</code> attribute than the one the administrator wants to create.
<b>Post-Condition (functional)</b>
PostF 1    the ctPolice class instance having the required id do not belong anymore to the post state as well as is related actPolice actor instance. PostF 2    the administrator actor is informed about the satisfaction of its request.
<b>Post-Condition (protocol)</b>
PostP 1    none

The listing 5.10 provides the **Mess1P** (MCL-oriented) specification of the operation.

```

1
2  /* Pre Protocol:*/
3  preP{let TheSystem: ctState in
4    let TheActor:actAdministrator in
5
6    self.rnActor.rnSystem = TheSystem
7    and self.rnActor = TheActor
8
9  /* PreP01 */
10 and TheSystem.vpStarted = true
11 /* PreP02 */
12 and TheActor.rnctAuthenticated.vpIsLogged = true}
13
14 /* Pre Functional:*/
15 preF{let TheSystem: ctState in
16  let TheActor:actAdministrator in
17
18  self.rnActor.rnSystem = TheSystem
19  and self.rnActor = TheActor
20 /* PreF01 */
21 TheSystem.rnctPolice->select(id.eq(AdtPoliceID))
22 = ColctPolices
23 and ColctPolices->size().eq(1)}
24
25 /* Post Functional:*/
26 postF{let TheSystem: ctState in

```

```

27 let TheActor:actAdministrator in
28 let ThectPolice:ctPolice in
29 self.rnActor.rnSystem = TheSystem
30 and self.rnActor = TheActor
31 /* PostF01 */
32 TheSystem.rnctPolice->select(id.eq(AdtPoliceID))
33 = ThectPolice
34 and ThectPolice.rnactPolice->forAll(msrIsKilled)
35 and ThectPolice.msrIsKilled
36
37 /* PostF02 */
38 and TheActor.rnInterfaceIN^iePoliceDeleted()
39
40 /* Post Protocol:*/
41 /* PostP01 */
42 and true}
43
44 /* Post Protocol:*/
45 postP{ true}

```

Listing 5.10: **Messip** (MCL-oriented) specification of the operation *oeDeletePolice*.

## 5.3 Environment - Out Interface Operation Scheme for actAuthenticated

### 5.3.1 Operation Model for oeLogin

The *oeLogin* operation has the following properties:

OPERATION
<i>oeLogin</i>
sent to request authorization to request access secured system operations.
<b>Parameters</b>
1 <b>AdtLogin: dtLogin</b> first information used to determine accessibility rights for the actual actor. 2 <b>AdtPassword: dtPassword</b> second information used to determine accessibility rights for the actual actor.
<b>Return type</b>
ptBoolean
<b>Pre-Condition (protocol)</b>
PreP 1    the system is started PreP 2    the actor is not already logged in ! (i.e. the associated ctAuthenticated instance is not considered logged)
<b>Pre-Condition (functional)</b>
PreF 1    none
<b>Post-Condition (functional)</b>
PostF 1    if the login and password provided by the actor correspond to the ones that belong to the ctAuthenticated instance he is related to then a welcome message is sent to the actor (n.b. the logged status is changed as a post-protocol condition); else the actor is notified that he gave incorrect data and all the administrator actors existing in the environement are notified of an intrusion temptative.
<b>Post-Condition (protocol)</b>
PostP 1    if the authentication information is correct then the actor is known to be logged in ! (i.e. the associated ctAuthenticated instance with given login and password is considered logged)

The listing 5.11 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Pre Protocol:*/
2  preP{let TheSystem: ctState in
3    let TheActor:actAuthenticated in
4      self.rnActor.rnSystem = TheSystem
5      and self.rnActor = TheActor
6
7
8  /* PreP01 */
9  and TheSystem.vpStarted = true
10 /* Prep02 */
11 and TheActor.rnctAuthenticated.vpIsLogged = false}
12
13 /* Pre Functional:*/
14 preF{/* PreF01 */
15 true}
16
17 /* Post Functional:*/
18 postF{let TheSystem: ctState in
19   let TheactAuthenticated:actAuthenticated in
20
21   let AptStringMessageForTheactAuthenticated: ptString in
22   let AptStringMessageForTheactAdministrator:ptString in
23
24   self.rnActor.rnSystem = TheSystem
25   and self.rnActor = TheactAuthenticated
26
27   and /* PostF01 */
28     if (TheactAuthenticated.rnctAuthenticated.pwd
29       = AdtPassword
30       and TheactAuthenticated.rnctAuthenticated.login
31         = AdtLogin
32     )
33     then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
34       and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
35     )
36   else (AptStringMessageForTheactAuthenticated
37     .eq('Wrong identification information ! Please try again ...')
38     and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
39     and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
40     and TheSystem.rnactAdministrator
41       .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
42     )
43   endif}
44
45 /* Post Protocol:*/
46 postP{ let TheSystem: ctState in
47   let TheactAuthenticated:actAuthenticated in
48
49   self.rnActor.rnSystem = TheSystem
50   and self.rnActor = TheactAuthenticated
51 /* PostP01 */
52   if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
53     and TheactAuthenticated.rnctAuthenticated.login = AdtLogin
54   )
55   then (TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true)
56   else true
57 endif}

```

Listing 5.11: **Messip** (MCL-oriented) specification of the operation *oeLogin*.

The listing 5.12 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

2 /* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4 %-----%
5 %
6 msrop(outactAuthenticated,
7     oeLogin,
8     [preProtocol,Self,
9      AdtLogin,
10     AdtPassword
11     ],
12     []):-!
13 /* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actAuthenticated,TheactAuthenticated),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheactAuthenticated]),
18 .
19 /* PreP01 */
20 msrNav([TheSystem],
21     [vpStarted],
22     [[ptBoolean,true]]),
23 .
24 msrNav([TheactAuthenticated],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,false]]),
27 .
28 .
29 msrop(outactAuthenticated,
30     oeLogin,
31     [preFunctional,Self,
32      AdtLogin,
33      AdtPassword
34      ],
35     []):-!
36 /* Pre Functional:*/
37 /* PreF01 */
38 true
39 .
40 .
41 msrop(outactAuthenticated,
42     oeLogin,
43     [post,Self,
44      AdtLogin,
45      AdtPassword
46      ],
47     []):-!
48 .
49 msrVar(ctState,TheSystem),
50 msrVar(actAuthenticated,TheactAuthenticated),
51 .
52 msrVar(ptString,AptStringMessageForTheactAuthenticated),
53 msrVar(ptString,AptStringMessageForTheactAdministrator),
54 .
55 /* Post Functional:*/
56 .
57 msrNav([Self],[rnActor],[TheactAuthenticated]),
58 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
59 .
60 /* PostF01 */
61 .
62 ( (msrNav([TheactAuthenticated],
63     [rnctAuthenticated,pwd],
64     [AdtPassword]),
65     msrNav([TheactAuthenticated],
66     [rnctAuthenticated,login],
67     [AdtLogin])
68   )
69  -> ( msrNav([AptStringMessageForTheactAuthenticated],
70     [eq,[[ptString,'You are logged ! Welcome ...']]],
71     [[ptBoolean,true]])),

```

```

72   msrNav([TheactAuthenticated],
73     [rnInterfaceIN,
74      ieMessage,[AptStringMessageForTheactAuthenticated]],
75      [[ptBoolean,true]])
76 )
77 ; ( msrNav([AptStringMessageForTheactAuthenticated],
78   [eq,[[ptString,'Wrong identification information ! Please try again ...']]],,
79   [[ptBoolean,true]]),
80   msrNav([TheactAuthenticated],
81     [rnInterfaceIN,
82      ieMessage,[AptStringMessageForTheactAuthenticated]],
83      [[ptBoolean,true]]),
84
85   msrNav([AptStringMessageForTheactAdministrator],
86     [eq,[[ptString,'Intrusion tentative !']]],,
87     [[ptBoolean,true]]),
88   msrNav([TheSystem],
89     [rnactAdministrator,rnInterfaceIN,
90      ieMessage,[AptStringMessageForTheactAdministrator]],
91      [[ptBoolean,true]])
92 )
93 ),
94
95 /* Post Protocol:*/
96 /* PostP01 */
97 ( (msrNav([TheactAuthenticated],
98   [rnctAuthenticated,pwd],
99   [AdtPassword]),
100  msrNav([TheactAuthenticated],
101   [rnctAuthenticated,login],
102   [AdtLogin])
103 )
104 -> (msrNav([TheactAuthenticated],
105   [rnctAuthenticated,msmAtPost,vpIsLogged],
106   [[ptBoolean,true]])
107 )
108 ; true
109 )
110 .

```

Listing 5.12: **Messip** (Prolog-oriented) implementation of the operation *oeLogout*.

### 5.3.2 Operation Model for *oeLogout*

The *oeLogout* operation has the following properties:

OPERATION
<b><i>oeLogout</i></b>
sent to end the secured access to specific system operations.
<b><i>Return type</i></b>
ptBoolean
<b><i>Pre-Condition (protocol)</i></b>
PreP 1 the system is started
PreP 2 the actor is currently logged in ! (i.e. the associated ctAuthenticated instance is considered logged)
<b><i>Pre-Condition (functional)</i></b>
PreF 1
<b><i>Post-Condition (functional)</i></b>
PostF 1 a logout confirmation message is sent to the actor (n.b. the logged status is changed as a post-protocol condition)

*continues in next page ...*

**... Operation table continuation****Post-Condition (protocol)**

PostP 1	the actor is known to be logged out ! (i.e. the associated ctAuthenticated instance with given login and password is considered logged out)
---------	---

The listing 5.13 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Pre Protocol:*/
2  preP{let TheSystem: ctState in
3    let TheActor:actAdministrator in
4    self.rnActor.rnSystem = TheSystem
5    and self.rnActor = TheActor
6
7
8  /* PreP01 */
9  and TheSystem.vpStarted = true
10 /* PreP02 */
11 and TheActor.rnctAuthenticated.vpIsLogged = true}
12
13 /* Pre Functional:*/
14 pref{/* PreF01 */
15 true}
16
17 /* Post Functional:*/
18 postF{let TheSystem: ctState in
19   let TheactAuthenticated:actAuthenticated in
20   let AptStringMessageForTheactAuthenticated: ptString in
21
22   self.rnActor.rnSystem = TheSystem
23   and self.rnActor = TheactAuthenticated
24
25 /* PostF01 */
26 AptStringMessageForTheactAuthenticated.eq('You are logged out ! Good Bye ...')
27 and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)}
28
29 /* Post Protocol:*/
30 postP{ let TheSystem: ctState in
31   let TheactAuthenticated:actAuthenticated in
32
33   self.rnActor.rnSystem = TheSystem
34   and self.rnActor = TheactAuthenticated.asset
35 /* PostP01 */
36 TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = false}
```

Listing 5.13: **Messip** (MCL-oriented) specification of the operation *oeLogout*.

The listing 5.14 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5%-----%
6msrop(outactAuthenticated,
7  oeLogout,
8  [preProtocol,Self
9    ],
10   []),
11  []):-%
12 /* Pre Protocol:*/
```

## 5.4. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOMCOMPANY77

```

12 msrVar(ctState,TheSystem),
13 msrVar(actAuthenticated,TheActor),
14 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
15 msrNav([Self],[rnActor],[TheActor]),
16
17 /* PreP01 */
18 msrNav([TheSystem],
19     [vpStarted],
20     [[ptBoolean,true]]),
21
22 msrNav([TheActor],
23     [rnctAuthenticated,vpIsLogged],
24     [[ptBoolean,true]])
25 .
26
27 msrop(outactAuthenticated,
28     oeLogout,
29     [preFunctional,Self
30     ],
31     []):-!
32 /* Pre Functional:*/
33 /* PreF01 */
34 true
35 .
36
37 msrop(outactAuthenticated,
38     oeLogout,
39     [post,Self
40     ],
41     []):-!
42
43 msrVar(ctState,TheSystem),
44 msrVar(actAuthenticated,TheactAuthenticated),
45
46 msrVar(ptString,AptStringMessageForTheactAuthenticated),
47
48 /* Post Functional:*/
49 msrNav([Self],[rnActor],[TheactAuthenticated]),
50 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
51
52 /* PostF01 */
53 msrNav([AptStringMessageForTheactAuthenticated],
54     [eq,[[ptString,'You are logged out ! Good Bye ...']]],
55     [[ptBoolean,true]]),
56 msrNav([TheactAuthenticated],
57     [rnInterfaceIN,
58     ieMessage,[AptStringMessageForTheactAuthenticated]],
59     [[ptBoolean,true]]),
60
61 /* Post Protocol:*/
62 /* PostP01 */
63 msrNav([TheactAuthenticated],
64     [rnctAuthenticated,msmAtPost,vpIsLogged],
65     [[ptBoolean,false]])
66.

```

Listing 5.14: **Messir** (Prolog-oriented) implementation of the operation *oeLogout*.

## 5.4 Environment - Out Interface Operation Scheme for actComCompany

### 5.4.1 Operation Model for oeAlert

The *oeAlert* operation has the following properties:

<b>OPERATION</b>	
<b><i>oeAlert</i></b>	
Any human having a phone able to connect to the communication companies using the <i>iCrash</i> system can send his company an sms message with structured information in order to declare an alert.	
<b>Parameters</b>	
1	<b>AetHumanKind: etHumanKind</b> the kind of human informing of an alert.
2	<b>AdtDate: dtDate</b> the date of the alert
3	<b>AdtTime: dtTime</b> the time of the alert
4	<b>AdtPhoneNumber: dtPhoneNumber</b> the phone number of the human sending the alert SMS message
5	<b>AdtGPSLocation: dtGPSLocation</b> the GPS position of the phone at the date and time the message was sent.
6	<b>AdtComment: dtComment</b> a free text message sent by the human providing information on the alert that he wants to declare
<b><i>Return type</i></b>	
ptBoolean	
<b><i>Pre-Condition (protocol)</i></b>	
PreP 1	the system is supposed to be created and initialized.
<b><i>Pre-Condition (functional)</i></b>	
PreF 1	the date and time the alert is declared is supposed to be in the past with respect to the current time known by the system.
<b><i>Post-Condition (functional)</i></b>	
PostF 1	the ctState attribute for the next value for alert IDs is incremented by one at post.
PostF 2	a new alert instance exists in the post state with status pending, instant information (resp. GPS location and comment) based on date and time provided (resp. position and comment); and with alert ID being a string conversion of the dtInteger value available in the pre state in the ctState instance.
PostF 3	if there exist no already registered alert near to the alert currently declared then a new crisis is added in the post state and initialized with: its ID being the one provided by the ctState instance (which is incremented by one in the post state), its type considered as small, its status being pending, its declared time being the same than the alert and a default comment indicating that a report will come later on. else the crisis to which the new alert must be related to is the one related to any alert nearby in the pre-state.
PostF 4	the post state relates the new alert to the previously characterized crisis.
PostF 5	if there is no ctHuman instance having same phone number and same kind in the pre-state then a new one is added in the post-state with given phone number and kind and is associated to the communication company actor used to declare the alert. else the pre-state one is chosen
PostF 6	and this specified ctHuman is related to the new alert thus indicating he has signed the alert.
<b><i>Post-Condition (protocol)</i></b>	
PostP 1	none

The listing 5.15 provides the **Messir** (MCL-oriented) specification of the operation.

```

68      and AetCrisisStatus = pending
69      and ACrisisInstant= AAlertInstant
70      and ACrisisdtComment = 'no reporting yet defined'
71      and ActCrisis.init( AdtCrisisID,
72          AdtCrisisType,
73          AetCrisisStatus,
74          AdtGPSLocation,
75          ACrisisInstant,
76          ACrisisdtComment)
77      )
78  else (ColctAlertsNearBy.rnTheCrisis->msrAny(true) = ActCrisis)
79 endif
80
81 /* PostF04 */
82 and ActAlert@post.rnTheCrisis = ActCrisis
83
84 /* PostF05 */
85 and TheSystem.rnctHuman->select(id.eq(AdtPhoneNumber)) = HumanColl
86
87 and HumanColl->select(kind.etEq(AetHumanKind)) = HumanCol2
88 and if (HumanCol2->msrIsEmpty)
89   then (ActHuman.init(AdtPhoneNumber,AetHumanKind)
90       and ActHuman@post.rnactComCompany = TheactComCompany
91   )
92 else (HumanCol2->any(true) = ActHuman)
93 endif
94
95 and ActHuman.rnSignaled->msrIncluding(ActAlert) = ColAlerts
96
97 and ActHuman@post.rnSignaled = ColAlerts
98
99 /* PostF06 */
100 AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
101 and TheactComCompany.rnInterfaceIN^ieSmsSend(AdtPhoneNumber,AdtSMS)}
102
103 /* Post Protocol:*/
104 postP{ true}

```

Listing 5.15: **Messip** (MCL-oriented) specification of the operation *oeAlert*.

The listing 5.16 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6nico(A):-
7 trace,
8 write('here'),
9 write('\n').
10
11msrop(outactComCompany,
12 oeAlert,
13 [preProtocol,Self,
14 AetHumanKind,
15 AdtDate,
16 AdtTime,
17 AdtPhoneNumber,
18 AdtGPSLocation,
19 AdtComment
20 ],
21 []):-!
22 /* Pre Protocol:*/
23 msrVar(ctState,TheSystem),
24 msrNav([Self],[rnActor,rnSystem],[TheSystem]),

```

## 5.4. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOMCOMPANY81

```

25/* PreP01 */
26 msrNav([TheSystem],
27   [vpStarted],
28   [[ptBoolean,true]]))
29 .
30
31msrop(outactComCompany,
32 oeAlert,
33 [preFunctional,Self,
34 AetHumanKind,
35 AdtDate,
36 AdtTime,
37 AdtPhoneNumber,
38 AdtGPSLocation,
39 AdtComment
40 ],
41 []):-!
42/* Pre Functional:*/
43/* PreP01 */
44 msrVar(ctState,TheSystem),
45 msrNav([Self],
46   [msmAtPre,rnActor,rnSystem],
47   [TheSystem]),
48
49 ( msrNav([TheSystem],[clock,date,gt,[AdtDate]],[[ptBoolean,true]]))
50 ; (msrNav([TheSystem],[clock,date,eq,[AdtDate]],[[ptBoolean,true]]))
51 , msrNav([TheSystem],[clock,time,gt,[AdtTime]],[[ptBoolean,true]]))
52 )
53 )
54 .
55
56msrop(outactComCompany,
57 oeAlert,
58 [post,Self,
59 AetHumanKind,
60 AdtDate,
61 AdtTime,
62 AdtPhoneNumber,
63 AdtGPSLocation,
64 AdtComment
65 ],
66 []):-!
67
68 msrVar(ctState,TheSystem),
69 msrVar(ctHuman,ActHuman),
70 msrVar(actComCompany,TheactComCompany),
71 msrVar(ctAlert,ActAlert),
72 msrVar(dtDateAndTime,AAlertInstant),
73 msrVar(etAlertStatus,AetAlertStatus),
74% msrVar(ctAlert,ActAlertNearBy),
75 msrVar(ctCrisis,ActCrisis),
76 msrVar(dtCrisisID,AdtCrisisID),
77% msrVar(etCrisisType,AetCrisisType),
78 msrVar(etCrisisStatus,AetCrisisStatus),
79 msrVar(dtDateAndTime,ACrisisInstant),
80 msrVar(dtComment,ACrisisdtComment),
81% msrVar(ptString,AptStringMessage),
82 msrVar(dtSMS,AdtSMS),
83 msrVar(dtAlertID,AdtAlertID),
84
85% msrVar(ptInteger,TheNextptIntegerValue),
86% msrVar(ptInteger,UpdatedNextptIntegerValue),
87% msrVar(inactComCompany,TheComCompanyIN),
88% msrVar(dtComment,TheCommentStored),
89% msrVar(dtString,TheCommentStoreddtString),
90
91/* Post Functional:*/
92
93 msrNav([Self],[rnActor],[TheactComCompany]),
94 msrNav([Self],[rnActor,rnSystem],[TheSystem]),

```



## 5.4. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOMCOMPANY83

```

165          ACrisisdtComment]],  

166          [[ptBoolean,true]]))  

167  

168      )  

169 ; (  

170     msrNav(ColctAlertsNearBy,  

171         [rnTheCrisis,msrAny,msrTrue],  

172         [ActCrisis])  

173     ),  

174 ),  

175  

176 /* PostF04 */  

177  

178 msrNav([ActAlert],  

179     [msmAtPost,rnTheCrisis],  

180     [ActCrisis]),  

181  

182/* PostF05 */  

183  

184 msrNav([TheSystem],  

185     [rnctHuman,  

186         msrSelect,id,eq,[AdtPhoneNumber]],  

187     HumanColl),  

188  

189 msrNav(HumanColl,  

190     [msrSelect,kind,etEq,[AetHumanKind]],  

191     HumanCol2),  

192  

193 (msrNav(HumanCol2,[msrIsEmpty],[[ptBoolean,true]]))  

194 -> (msrNav([ActHuman],  

195     [init,[AdtPhoneNumber,AetHumanKind]],  

196     [[ptBoolean,true]]),  

197     msrNav([ActHuman],  

198     [msmAtPost,rnactComCompany],  

199     [TheactComCompany]))  

200 )  

201 ; msrNav(HumanCol2,  

202     [msrAny],  

203     [ActHuman])  

204 ),  

205  

206msrNav([ActHuman],  

207     [rnSignaled,msrIncluding,[ActAlert]],  

208     ColAlerts),  

209  

210msrNav([ActHuman],  

211     [msmAtPost,rnSignaled],  

212     ColAlerts),  

213  

214/* PostF06 */  

215msrNav([AdtSMS],  

216     [value],  

217     [[ptString,'Your alert has been registered. We will handle it and keep you informed']])),  

218msrNav([TheactComCompany],  

219     [rnInterfaceIN,  

220     ieSmsSend,[AdtPhoneNumber,  

221         AdtSMS]],[[ptBoolean,true]]),  

222  

223/*  

224  

225 */  

226  

227 /* Post Protocol:*/  

228/* PostP01 */  

229 true  

230 .

```

Listing 5.16: **Messir** (Prolog-oriented) implementation of the operation *oeAlert*.

Figure 5.2 shows concept model elements in the scope of the *oeAlert* operation

Figure 5.2: oeAlert operation scope

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR85

Figure 5.3 shows concept model elements in the scope of the oeAlert operation

Figure 5.3: oeAlert operation scope

## 5.5 Environment - Out Interface Operation Scheme for actCoordinator

### 5.5.1 Operation Model for oeCloseCrisis

The oeCloseCrisis operation has the following properties:

<b>OPERATION</b>	
<b><i>oeCloseCrisis</i></b>	
sent to indicate that a crisis should be considered as closed.	
<b>Parameters</b>	
1	<b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis to close
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	it is supposed that there exist one ctCrisis instance with the same <code>id</code> attribute value as the one provided by the coordinator actor who wants to close.
<b>Post-Condition (functional)</b>	
PostF 1	the ctCrisis class instance having the provided id is considered closed in the post state.
PostF 2	There is no handler declared in the system as associated to the crisis.
PostF 3	all the alert instances associated to this crisis do not belong any more to the system's post state.
PostF 4	the coordinator actor is informed about the satisfaction of its request.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.17 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeCloseCrisis,
8    [preProtocol,Self,
9     AdtCrisisID
10    ],
11    []):-!
12/* Pre Protocol*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23/* PreP02 */
24 msrNav([TheActor],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,true]])
27.
28

```

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR87

```

29msrop(outactCoordinator,
30    oeCloseCrisis,
31    [preFunctional,Self,
32     AdtCrisisID
33     ],
34     []):-.
35/* Pre Functional:*/
36 msrVar(ctState,TheSystem),
37 msrVar(actCoordinator,TheActor),
38
39 msrVar(dtCrisisID,AdtCrisisID),
40
41 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
42 msrNav([Self],[rnActor],[TheActor]),
43
44/* PreF01 */
45 msrNav([TheSystem],
46     [rnctCrisis,
47      msrSelect,
48      id,eq,[AdtCrisisID]
49     ],
50     ColCrisis),
51
52 msrNav(ColCrisis,
53     [msrSize,eq,[[ptInteger,1]]],
54     [[ptBoolean,true]])
55 .
56
57msrop(outactCoordinator,
58    oeCloseCrisis,
59    [post,Self,
60     AdtCrisisID
61     ],
62     []):-.
63
64/* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actCoordinator,TheActor),
67
68 msrVar(ctCrisis,TheCrisis),
69 msrVar(dtCrisisID,AdtCrisisID),
70
71 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
72 msrNav([Self],[rnActor],[TheActor]),
73
74/* PostF01 */
75 msrNav([TheSystem],
76     [rnctCrisis,
77      msrSelect,
78      id,eq,[AdtCrisisID]],
79     [TheCrisis]),
80
81 msrNav([TheCrisis],
82     [msmAtPost,status],
83     [[etCrisisStatus,closed]]),
84
85/* PostF02 */
86 msrNav([TheCrisis],
87     [msmAtPost,rnHandler],
88     []),
89
90 /* PostF03 */
91 msrNav([TheCrisis],
92     [rnAlerts,msrForAll,msrIsKilled],
93     [[ptBoolean,true]]),
94
95/* PostF04 */
96 msrNav([TheActor],
97     [rnInterfaceIN,
98      ieMessage,[[ptString,'The crisis is now closed !']]])
```

```

99      ],
100     [[ptBoolean,true]]),
101
102/* Post Protocol:*/
103/* PostP01 */
104 true
105 .

```

Listing 5.17: **Messip** (Prolog-oriented) implementation of the operation *oeCloseCrisis*.

### 5.5.2 Operation Model for oeGetAlertsSet

The `oeGetAlertsSet` operation has the following properties:

OPERATION	
<b><i>oeGetAlertsSet</i></b>	
sent to request all the <code>ctAlert</code> instances having a specific status.	
<b>Parameters</b>	
1	<b>AetAlertStatus: etAlertStatus</b> the criteria used to select the alerts to send back to the actor
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated <code>ctCoordinator</code> instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	none
<b>Post-Condition (functional)</b>	
PostF 1	the post state is the one obtained by satisfying the <code>isSentToCoordinator</code> predicate for each alert having the provided status and for the actor sending the message. (cf. specification of <code>isSentToCoordinator</code> predicate given for the <code>ctAlert</code> type.)
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.18 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7  oeGetAlertsSet,
8  [preProtocol,Self,
9   AetAlertStatus
10  ],
11  []):- 
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),

```

```

17
18/* PreP01 */
19 msrNav([TheSystem],
20   [vpStarted,
21    [[ptBoolean,true]]) ,
22
23 msrNav([TheActor],
24   [rnctAuthenticated,vpiIsLogged],
25   [[ptBoolean,true]])
26.
27
28msrop(outactCoordinator,
29  oeGetAlertsSet,
30  [preFunctional,Self,
31   AetAlertStatus
32   ],
33   []):-!
34/* Pre Functional:*/
35/* PreF01 */
36true
37 .
38
39msrop(outactCoordinator,
40  oeGetAlertsSet,
41  [post,Self,
42   AetAlertStatus
43   ],
44   []):-!
45
46/* Post Functional:*/
47 msrVar(ctState,TheSystem),
48 msrVar(actCoordinator,TheActor),
49 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
50 msrNav([Self],[rnActor],[TheActor]),
51
52/* PostF01 */
53 msrNav([TheSystem],
54   [rnctAlert,
55    msrSelect,
56    status,etEq,[AetAlertStatus]],
57   ColAlertSet),
58
59 msrNav(ColAlertSet,
60   [msrForAll,isSentToCoordinator,[TheActor]],
61   [[ptBoolean,true]]) ,
62
63 /* Post Protocol:*/
64/* PostP01 */
65 true
66 .

```

Listing 5.18: **Messir** (Prolog-oriented) implementation of the operation *oeGetAlertsSet*.

### 5.5.3 Operation Model for *oeGetCrisisSet*

The *oeGetCrisisSet* operation has the following properties:

<b>OPERATION</b>
<b>oeGetCrisisSet</b>
sent to request all the <i>ctCrisis</i> instances having a specific status.
<b>Parameters</b>
1 <b>AetCrisisStatus: etCrisisStatus</b> the status information used to determine the crisis to send back to the actor
<b>Return type</b>

*continues in next page...*

***... Operation table continuation***

ptBoolean
<b><i>Pre-Condition (protocol)</i></b>
PreP 1 the system is started
PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b><i>Pre-Condition (functional)</i></b>
PreF 1 none
<b><i>Post-Condition (functional)</i></b>
PostF 1 the post state is the one obtained by satisfying the <code>isSentToCoordinator</code> predicate for each crisis having the provided status and for the actor sending the message <code>ieSendACrisis</code> . (cf. specification of <code>isSentToCoordinator</code> predicate given for the <code>ctCrisis</code> type.)
<b><i>Post-Condition (protocol)</i></b>
PostP 1 none

The listing 5.19 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%-----%
6msrop(outactCoordinator,
7    oeGetCrisisSet,
8    [preProtocol,Self,
9     AetCrisisStatus
10    ],
11   []):-!
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17 .
18/* PreP01 */
19 msrNav([TheSystem],
20        [vpStarted],
21        [[ptBoolean,true]]),
22 .
23 msrNav([TheActor],
24        [rnctAuthenticated,vpIsLogged],
25        [[ptBoolean,true]]))
26 .
27 .
28msrop(outactCoordinator,
29    oeGetCrisisSet,
30    [preFunctional,Self,
31     AetCrisisStatus
32    ],
33   []):-!
34/* Pre Functional:*/
35/* PreF01 */
36true
37 .
38 .
39msrop(outactCoordinator,
40    oeGetCrisisSet,

```

```

41  [post,Self,
42   AetCrisisStatus
43   ],
44   []):-  

45
46 /* Post Functional:*/
47 msrVar(ctState,TheSystem),
48 msrVar(actCoordinator,TheActor),
49 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
50 msrNav([Self],[rnActor],[TheActor]),
51
52 /* PostF01 */
53 msrNav([TheSystem],
54   [rnctCrisis,
55    msrSelect,
56    status,etEq,[AetCrisisStatus]],
57   ColCrisisSet),
58
59 msrNav(ColCrisisSet,
60   [msrForAll,isSentToCoordinator,[TheActor]],
61   [[ptBoolean,true]]),
62
63 /* Post Protocol:*/
64 /* PostP01 */
65 true
66 .

```

Listing 5.19: **Messir** (Prolog-oriented) implementation of the operation *oeGetCrisisSet*.

#### 5.5.4 Operation Model for oeInvalidateAlert

The *oeInvalidateAlert* operation has the following properties:

OPERATION
<b><i>oeInvalidateAlert</i></b>
sent to indicate that an alert should be considered as closed.
<b><i>Parameters</i></b>
1 <b>AdtAlertID: dtAlertID</b> the identification information used to determine the alert to close
<b><i>Return type</i></b>
ptBoolean
<b><i>Pre-Condition (protocol)</i></b>
PreP 1    the system is started PreP 2    the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b><i>Pre-Condition (functional)</i></b>
PreF 1    it is supposed that there exist one ctAlert instance with the same <b>id</b> attribute value as the one provided by the coordinator actor who wants to close.
<b><i>Post-Condition (functional)</i></b>
PostF 1    the ctAlert class instance having the provided id is considered closed in the post state. PostF 2    the coordinator actor is informed about the satisfaction of its request.
<b><i>Post-Condition (protocol)</i></b>
PostP 1    none

The listing 5.20 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeInvalidateAlert,
8    [preProtocol,Self,
9     AdtAlertID
10    ],
11    []):-.
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17 .
18/* PreP01 */
19 msrNav([TheSystem],
20        [vpStarted],
21        [[ptBoolean,true]]),
22 .
23/* PreP02 */
24 msrNav([TheActor],
25        [rnctAuthenticated,vpIsLogged],
26        [[ptBoolean,true]]),
27 .
28
29msrop(outactCoordinator,
30    oeInvalidateAlert,
31    [preFunctional,Self,
32     AdtAlertID
33    ],
34    []):-.
35/* Pre Functional:*/
36 msrVar(ctState,TheSystem),
37 msrVar(actCoordinator,TheActor),
38 .
39 msrVar(dtAlertID,AdtAlertID),
40 .
41 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
42 msrNav([Self],[rnActor],[TheActor]),
43 .
44/* PreF01 */
45 msrNav([TheSystem],
46        [rnctAlert,
47         msrSelect,
48         id,eq,[AdtAlertID]
49        ],
50        ColAlert),
51 .
52 msrNav(ColAlert,
53        [msrSize,eq,[[ptInteger,1]]],
54        [[ptBoolean,true]]),
55 .
56
57msrop(outactCoordinator,
58    oeInvalidateAlert,
59    [post,Self,
60     AdtAlertID
61    ],
62    []):-.
63
64/* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actCoordinator,TheActor),
67 .

```

```

68 msrVar(ctAlert,TheAlert),
69 msrVar(dtAlertID,AdtAlertID),
70
71 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
72 msrNav([Self],[rnActor],[TheActor]),
73
74 /* PostF01 */
75 msrNav([TheSystem],
76     [rnctAlert,
77      msrSelect,
78      id,eq,[AdtAlertID]],
79     [TheAlert]),
80
81 msrNav([TheAlert],
82     [msmAtPost,status],
83     [[etAlertStatus,invalid]]),
84
85 /* PostF02 */
86 msrNav([TheActor],
87     [rnInterfaceIN,
88      ieMessage,[[ptString,'The alert is now declared as invalid !']],
89      ],
90     [[ptBoolean,true]]),
91
92 /* Post Protocol:*/
93 /* PostP01 */
94 true
95 .

```

Listing 5.20: **Messir** (Prolog-oriented) implementation of the operation *oeInvalidateAlert*.

### 5.5.5 Operation Model for oeReportOnCrisis

The *oeReportOnCrisis* operation has the following properties:

OPERATION
<b><i>oeReportOnCrisis</i></b>
sent to update the textual information available for a specific handled crisis.
<b>Parameters</b>
1 <b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis to report on
2 <b>AdtComment: dtComment</b> the textual information commenting the crisis
<b>Return type</b>
ptBoolean
<b>Pre-Condition (protocol)</b>
PreP 1    the system is started PreP 2    the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b>Pre-Condition (functional)</b>
PreF 1    it is supposed that there exist one crisis in the pre state having the given id.
<b>Post-Condition (functional)</b>
PostF 1    the comment attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
<b>Post-Condition (protocol)</b>
PostP 1    none

The listing 5.21 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeReportOnCrisis,
8    [preProtocol,Self,
9     AdtCrisisID,
10    AdtComment
11    ],
12    []):-!
13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actCoordinator,TheActor),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18
19/* PreP01 */
20 msrNav([TheSystem],
21        [vpStarted],
22        [[ptBoolean,true]]),
23
24 msrNav([TheActor],
25        [rnctAuthenticated,vpIsLogged],
26        [[ptBoolean,true]]))
27.
28
29msrop(outactCoordinator,
30    oeReportOnCrisis,
31    [preFunctional,Self,
32     AdtCrisisID,
33     AdtComment
34     ],
35    []):-!
36/* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39
40 msrVar(dtCrisisID,AdtCrisisID),
41
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44
45/* PreF01 */
46 msrNav([TheSystem],
47        [rnctCrisis,
48         msrSelect,
49         id,eq,[AdtCrisisID]
50        ],
51        ColCrisis),
52
53 msrNav(ColCrisis,
54        [msrSize,eq,[[ptInteger,1]]],
55        [[ptBoolean,true]]))
56.
57
58msrop(outactCoordinator,
59    oeReportOnCrisis,
60    [post,Self,
61     AdtCrisisID,
62     AdtComment
63     ],
64    []):-!
65
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),

```

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR95

```

68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(dtComment,AdtComment),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77 /* PostF01 */
78 msrNav([TheSystem],
79   [rnctCrisis,
80    msrSelect,
81    id,eq,[AdtCrisisID]],
82   [TheCrisis]),
83
84 msrNav([TheCrisis],
85   [msmAtPost,comment],
86   [AdtComment]),
87
88 msrNav([TheActor],
89   [rnInterfaceIN,
90    ieMessage,[[ptString,'The crisis comment has been updated !']],
91    ],
92   [[ptBoolean,true]]),
93
94 /* Post Protocol:*/
95 /* PostP01 */
96 true
97 .

```

Listing 5.21: **Messir** (Prolog-oriented) implementation of the operation *oeReportOnCrisis*.

### 5.5.6 Operation Model for oeSetCrisisHandler

The *oeSetCrisisHandler* operation has the following properties:

OPERATION
<b><i>oeSetCrisisHandler</i></b>
sent to declare himself as been the handler of a crisis having the specified id.
<b>Parameters</b>
1 <b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis
<b>Return type</b>
ptBoolean
<b>Pre-Condition (protocol)</b>
PreP 1    the system is started PreP 2    the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b>Pre-Condition (functional)</b>
PreF 1    there exist one crisis having the given id in the pre-state.
<b>Post-Condition (functional)</b>
PostF 1    the ctCrisis instance having the provided id is in handled status at poststate and is associated to the actor that sends the message (which himself is notified with a textual message as confirmation). PostF 2    All the alerts related to this crisis are sent to the actor such that he can decide how to handle them.

***continues in next page ...***

**... Operation table continuation**

- |         |  |
|---------|--|
| PostF 3 | if the crisis was already handled at pre-sate then the associated handler actor is notified about the change of handler for one of his crisis (n.b. it might be the same even if not relevant).  |
| PostF 4 | a message is sent to the communication company for any human related to an alert associated to the crisis. A human will receive as many messages as alerts he sent despite the fact that they might relate to the same crisis (i.e. one alert, one acknowledgement). |

***Post-Condition (protocol)***

- |         |      |
|---------|------|
| PostP 1 | none |
|---------|------|

The listing 5.22 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5%-----%
6msrop(outactCoordinator,
7    oeSetCrisisHandler,
8    [preProtocol,Self,
9     AdtCrisisID
10    ],
11    []):-!
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20    [vpStarted],
21    [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24    [rnctAuthenticated,vpIsLogged],
25    [[ptBoolean,true]]),
26.
27
28msrop(outactCoordinator,
29    oeSetCrisisHandler,
30    [preFunctional,Self,
31     AdtCrisisID
32    ],
33    []):-!
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtCrisisID,AdtCrisisID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45    [rnctCrisis,
46     msrSelect,
47     id,eq,[AdtCrisisID]
48    ],
49    ColCrisis),
50

```

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR97

```

51 msrNav(ColCrisis,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56msrop(outactCoordinator,
57 oeSetCrisisHandler,
58 [post,Self,
59 AdtCrisisID
60 ],
61 []):-  

62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66 msrVar(ctCoordinator,TheCoordinator),
67 msrVar(ctCoordinator,TheCurrentHandler),
68
69 msrVar(ctCrisis,TheCrisis),
70 msrVar(dtCrisisID,AdtCrisisID),
71
72 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
73 msrNav([Self],[rnActor],[TheActor]),
74
75 /* PostF01 */
76 msrNav([TheSystem],
77     [rnctCrisis,
78      msrSelect,
79      id,eq,[AdtCrisisID]],
80     [TheCrisis]),
81
82 msrNav([TheCrisis],
83     [msmAtPost,status],
84     [[etCrisisStatus,handled]]),
85
86 msrNav([TheActor],
87     [rnctCoordinator],
88     [TheCoordinator]),
89 msrNav([TheCrisis],
90     [msmAtPost,rnHandler],
91     [TheCoordinator]),
92
93 msrNav([TheActor],
94     [rnInterfaceIN,
95      ieMessage,[[ptString,'You are now considered as handling the crisis !']],
96      ],
97      [[ptBoolean,true]]),
98
99 /* PostF02 */
100 msrNav([TheCrisis],
101     [rnAlerts,msrForAll,isSentToCoordinator,[TheActor]],
102     [[ptBoolean,true]]),
103
104 /* PostF03 */
105 ( msrNav([TheCrisis],
106     [rnHandler,msrSize,eq,[[ptInteger,1]]],
107     [[ptBoolean,true]])
108 -> (msrNav([TheCrisis],
109     [rnHandler],
110     [TheCurrentHandler]),
111     msrNav([TheCurrentHandler],
112     [rnactCoordinator,rnInterfaceIN,
113      ieMessage,[[ptString,'One of the crisis you were handling is now handled by one of your
114      colleagues!']]
115      ],
116      [[ptBoolean,true]]))
117 ; true
118 ),
119

```

```

120  /* PostF04 */
121  msrNav([TheCrisis,
122    [rnAlerts,rnSignaler,msrForAll,isAcknowledged,[]],
123    [[ptBoolean,true]]),
124
125 /* Post Protocol:*/
126 /* PostP01 */
127 true
128 .

```

Listing 5.22: **Messip** (Prolog-oriented) implementation of the operation *oeSetCrisisHandler*.

### 5.5.7 Operation Model for *oeSetCrisisStatus*

The *oeSetCrisisStatus* operation has the following properties:

<b>OPERATION</b>	
<b><i>oeSetCrisisStatus</i></b>	
sent to define the handling status of a specific crisis.	
<b>Parameters</b>	
1 <b>AdtCrisisID: dtCrisisID</b>	the identification information used to determine the crisis
2 <b>AetCrisisStatus: etCrisisStatus</b>	the new status value
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
<b>Post-Condition (functional)</b>	
PostF 1	the crisis status attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.23 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7  oeSetCrisisStatus,
8  [preProtocol,Self,
9   AdtCrisisID,
10  AetCrisisStatus
11  ],
12  []):-
```

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR99

```

13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actCoordinator,TheActor),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18 .
19/* PreP01 */
20 msrNav([TheSystem],
21     [vpStarted],
22     [[ptBoolean,true]]),
23 .
24 msrNav([TheActor],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,true]]),
27 .
28 .
29 msrop(outactCoordinator,
30     oeSetCrisisStatus,
31     [preFunctional,Self,
32      AdtCrisisID,
33      AetCrisisStatus
34     ],
35     []):-.
36/* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39 .
40 msrVar(dtCrisisID,AdtCrisisID),
41 .
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44 .
45/* PreF01 */
46 msrNav([TheSystem],
47     [rnctCrisis,
48      msrSelect,
49      id,eq,[AdtCrisisID]
50     ],
51     ColCrisis),
52 .
53 msrNav(ColCrisis,
54     [msrSize,eq,[[ptInteger,1]]],
55     [[ptBoolean,true]]),
56 .
57 .
58 msrop(outactCoordinator,
59     oeSetCrisisStatus,
60     [post,Self,
61      AdtCrisisID,
62      AetCrisisStatus
63     ],
64     []):-.
65 .
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69 .
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisStatus,AetCrisisStatus),
73 .
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76 .
77/* PostF01 */
78 msrNav([TheSystem],
79     [rnctCrisis,
80      msrSelect,
81      id,eq,[AdtCrisisID]],
82     [TheCrisis]),
```

```

83
84 msrNav([TheCrisis],
85     [msmAtPost,status],
86     [AetCrisisStatus]),
87
88 msrNav([TheActor],
89     [rnInterfaceIN,
90     ieMessage,[[ptString,'The crisis status has been updated !']],
91     ],
92     [[ptBoolean,true]]),
93
94 /* Post Protocol:*/
95 /* PostP01 */
96 true
97 .

```

Listing 5.23: **Messir** (Prolog-oriented) implementation of the operation *oeSetCrisisStatus*.

### 5.5.8 Operation Model for *oeSetCrisisType*

The *oeSetCrisisType* operation has the following properties:

<b>OPERATION</b>	
<i>oeSetCrisisType</i>	
sent to define the gravity type of a specific crisis.	
<i>Parameters</i>	
1	<b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis
2	<b>AetCrisisType: etCrisisType</b> the new type value
<i>Return type</i>	
ptBoolean	
<i>Pre-Condition (protocol)</i>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<i>Pre-Condition (functional)</i>	
PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
<i>Post-Condition (functional)</i>	
PostF 1	the crisis type attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
<i>Post-Condition (protocol)</i>	
PostP 1	none

The listing 5.24 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,

```

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR101

```

7   oeSetCrisisType,
8   [preProtocol,Self,
9    AdtCrisisID,
10   AetCrisisType
11   ],
12  []):-.
13 /* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actCoordinator,TheActor),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18 .
19 /* PreP01 */
20 msrNav([TheSystem],
21   [vpStarted],
22   [[ptBoolean,true]]),
23 .
24 msrNav([TheActor],
25   [rnctAuthenticated,vpIsLogged],
26   [[ptBoolean,true]]),
27 .
28 .
29 msrop(outactCoordinator,
30   oeSetCrisisType,
31   [preFunctional,Self,
32    AdtCrisisID,
33    AetCrisisType
34    ],
35  []):-.
36 /* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39 .
40 msrVar(dtCrisisID,AdtCrisisID),
41 .
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44 .
45 /* PreP01 */
46 msrNav([TheSystem],
47   [rnctCrisis,
48    msrSelect,
49    id,eq,[AdtCrisisID]
50   ],
51   ColCrisis),
52 .
53 msrNav(ColCrisis,
54   [msrSize,eq,[[ptInteger,1]]],
55   [[ptBoolean,true]]),
56 .
57 .
58 msrop(outactCoordinator,
59   oeSetCrisisType,
60   [post,Self,
61    AdtCrisisID,
62    AetCrisisType
63    ],
64  []):-.
65 .
66 /* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69 .
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisType,AetCrisisType),
73 .
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76 
```

```

77 /* PostF01 */
78 msrNav([TheSystem],
79     [rnctCrisis,
80      msrSelect,
81      id,eq,[AdtCrisisID]],
82     [TheCrisis]),
83
84 msrNav([TheCrisis],
85     [msmAtPost,type],
86     [AetCrisisType]),
87
88 msrNav([TheActor],
89     [rnInterfaceIN,
90      ieMessage,[[ptString,'The crisis type has been updated !']],
91      ],
92     [[ptBoolean,true]]),
93
94 /* Post Protocol:*/
95 /* PostP01 */
96 true
97 .

```

Listing 5.24: **Messip** (Prolog-oriented) implementation of the operation *oeSetCrisisType*.

### 5.5.9 Operation Model for oeValidateAlert

The *oeValidateAlert* operation has the following properties:

<b>OPERATION</b>	
<b><i>oeValidateAlert</i></b>	
sent to indicate that a specific alert is not a fake.	
<b>Parameters</b>	
1	<b>AdtAlertID: dtAlertID</b> the identification information used to determine the alert instance
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctCoordinator instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	it is supposed that there exist one ctAlert instance with the same <code>id</code> attribute value as the one provided by the coordinator actor who wants to validate.
<b>Post-Condition (functional)</b>	
PostF 1	the ctAlert class instance having the provided id is considered as valid in the post state and the coordinator actor is informed about the satisfaction of its request.
<b>Post-Condition (protocol)</b>	
PostP 1	none

The listing 5.25 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */

```

## 5.5. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTCOORDINATOR103

```

3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%-----%
6msrop(outactCoordinator,
7    oeValidateAlert,
8    [preProtocol,Self,
9     AdtAlertID
10    ],
11    []):-%
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpiIsLogged],
25     [[ptBoolean,true]]))
26.
27
28msrop(outactCoordinator,
29    oeValidateAlert,
30    [preFunctional,Self,
31     AdtAlertID
32    ],
33    []):-%
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtAlertID,AdtAlertID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45     [rnctAlert,
46      msrSelect,
47      id,eq,[AdtAlertID]
48    ],
49    ColAlerts),
50
51 msrNav(ColAlerts,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56msrop(outactCoordinator,
57    oeValidateAlert,
58    [post,Self,
59     AdtAlertID
60    ],
61    []):-%
62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66
67 msrVar(ctAlert,TheAlert),
68 msrVar(dtAlertID,AdtAlertID),
69
70 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
71 msrNav([Self],[rnActor],[TheActor]),
72

```

```

73 /* PostF01 */
74 msrNav([TheSystem],
75     [rnctAlert,
76      msrSelect,
77      id,eq,[AdtAlertID]],
78     [TheAlert]),
79
80 msrNav([TheAlert],
81     [msmAtPost,status],
82     [[etAlertStatus,valid]]),
83
84 msrNav([TheActor],
85     [rnInterfaceIN,
86      ieMessage,[[ptString,'The Alert is now declared as valid !']],
87      ],
88     [[ptBoolean,true]]),
89
90 /* Post Protocol:*/
91 /* PostP01 */
92 true
93 .

```

Listing 5.25: **Messip** (Prolog-oriented) implementation of the operation *oeValidateAlert*.

## 5.6 Environment - Out Interface Operation Scheme for actMsrCreator

### 5.6.1 Operation Model for oeCreateSystemAndEnvironment

The *oeCreateSystemAndEnvironment* operation has the following properties:

<b>OPERATION</b>	
<i>oeCreateSystemAndEnvironment</i>	
sent to request the initialization of the system's class instances and the environment actors instances.	
<b>Parameters</b>	
1	<b>AqtyComCompanies: ptInteger</b> the quantity of communication companies to create in the environment
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	none
<b>Pre-Condition (functional)</b>	
PreF 1	none
<b>Post-Condition (functional)</b>	
PostF 1	the ctState instance is initialized with the integer 1 for the crisis and alert counters used for their identifications, a value for the clock corresponding to a default initial time (i.e. January 1st, 1970) the crisis reminder period is set to 300 seconds, the maximum crisis reminder period is fixed to 1200 seconds (i.e. 20 minutes), an initial value for the automatic reminder period equal to the current date and time and the system is considered in a started state. <b>Those predicates must be satisfied first since all the other depend on the existence of a ctState instance !</b>
PostF 2	the actMsrCreator actor instance is initiated (remember that since the oeCreateSystemAndEnvironment is a special event its role is to make consistent the post state thus creating the actor and its interfaces is required even though the sending of this message logically would need the actor and its interfaces to already exist ...).

**continues in next page ...**

**... Operation table continuation**

PostF 3	the environment for communication company actors, in the post state, is made of AqtyComCompanies instances allowing for receiving and sending messages to humans.
PostF 4	the environment for administrator actors, in the post state, is made of one instance.
PostF 5	the environment for activator actors, in the post state, is made of one instance allowing for automatic message sending based on current system's and environment state'.
PostF 6	the set of ctAdministrator instances at post is made of one instance initialized with 'icrashadmin' (resp. '7WXC1359') for login (resp. password) values.
PostF 7	the association between ctAdministrator and actAdministrator is made of one couple made of the conjointly specified instances.

<b>Post-Condition (protocol)</b>	
PostP 1	none is given since the only protocol variable to be modified in the post state is the one initialized with the ctState instance (i.e. vpStarted).

The listing 5.26 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Pre Protocol:*/
2  preP{true}
3
4
5  /* Pre Functional:*/
6  preF{true}
7
8  /* Post Functional:*/
9  postF{let TheSystem: ctState in
10 let AactMsrCreator: actMsrCreator in
11 let AactAdministrator: actAdministrator in
12 let AnextValueForAlertID: dtInteger in
13 let AnextValueForCrisisID: dtInteger in
14 let Aclock: dtDateAndTime in
15 let AcrisisReminderPeriod: dtSecond in
16 let AmaxCrisisReminderPeriod: dtSecond in
17 let AvpStarted: ptBoolean in
18
19  /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
20  AnextValueForAlertID.value.eq(1)
21  and AnextValueForCrisisID.value.eq(1)
22  and Aclock.date.year.value = 1970
23  and Aclock.date.month.value = 01
24  and Aclock.date.day.value = 01
25  and Aclock.time.hour.value = 00
26  and Aclock.time.minute.value = 00
27  and Aclock.time.second.value = 00
28
29  and AcrisisReminderPeriod.value.eq(300)
30  and AmaxCrisisReminderPeriod.value.eq(1200)
31  and AvpStarted = true
32  and TheSystem.init(AnextValueForAlertID,
33      AnextValueForCrisisID,
34      Aclock,
35      AcrisisReminderPeriod,
36      AmaxCrisisReminderPeriod,
37      Aclock,
38      AvpStarted
39  )
40  /* PostF02*/
41  and AactMsrCreator.init()
42  /* PostF03 */
43  and let AactComCompanyCol: Bag(actComCompany) in
44  AactComCompanyCol->size() = AqtyComCompanies

```

```

45 AactComCompanyCol-> forAll(init())
46 /* PostF04*/
47 and AactAdministrator.init()
48 /* PostF05*/
49 and let AactActivator:actActivator in
50 AactActivator.init()
51 /* PostF06 */
52 and let ActAdministrator:ctAdministrator in
53   let AdtLogin:dtLogin in
54     let AdtPassword:dtPassword in
55       AdtLogin.value.eq('icrashadmin')
56       and AdtPassword.value.eq('7WXC1359')
57       and ActAdministrator.init(AdtLogin,AdtPassword)
58 /* PostF07*/
59 and ActAdministrator@post.rnactAuthenticated = AactAdministrator}
60
61 /* Post Protocol:*/
62 postP{ true}

```

Listing 5.26: **Messip** (MCL-oriented) specification of the operation *oeCreateSystemAndEnvironment*.

The listing 5.27 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5/*
6*****
7MSRCreatorActor
8*****
9
10/*** createSystemAndEnvironment ***/
11
12msrop(outactMsrCreator,
13  oeCreateSystemAndEnvironment,
14  [preFunctional,_Self,_AqtyComCompanies],
15  []):-  

16  true.
17
18msrop(outactMsrCreator,
19  oeCreateSystemAndEnvironment,
20  [preProtocol,_Self,_AqtyComCompanies],
21  []):-  

22  true.
23
24msrop(outactMsrCreator,
25  oeCreateSystemAndEnvironment,
26  [post,_Self,AqtyComCompanies],
27  []):-  

28
29 msrVar(ctState,TheSystem),
30 msrVar(actMsrCreator,AactMsrCreator),
31 msrVar(actAdministrator,AactAdministrator),
32
33 msrVar(dtInteger, AnextValueForAlertID),
34 msrVar(dtInteger, AnextValueForCrisisID),
35 msrVar(dtDateAndTime, Aclock),
36 msrVar(dtSecond, AcrisisReminderPeriod),
37 msrVar(dtSecond, AmaxCrisisReminderPeriod),
38 msrVar(ptBoolean, AvpStarted),
39
40 /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
41 msrNav([AnextValueForAlertID],
42           [value,eq,[ptInteger,1]]),

```

## 5.6. ENVIRONMENT - OUT INTERFACE OPERATION SCHEME FOR ACTMSRCREATOR107

```
43      [[ptBoolean,true]]),
44
45 msrNav([AnextValueForCrisisID],
46     [value,eq,[[ptInteger,1]]],
47     [[ptBoolean,true]]),
48
49msrNav([Aclock],
50     [date,year,value],
51     [[ptInteger,1970]]),
52msrNav([Aclock],
53     [date,month,value],
54     [[ptInteger,01]]),
55msrNav([Aclock],
56     [date,day,value],
57     [[ptInteger,01]]),
58
59msrNav([Aclock],
60     [time,hour,value],
61     [[ptInteger,00]]),
62msrNav([Aclock],
63     [time,minute,value],
64     [[ptInteger,00]]),
65msrNav([Aclock],
66     [time,second,value],
67     [[ptInteger,00]]),
68
69 msrNav([AcrisisReminderPeriod],
70     [value,eq,[[ptInteger,300]]],
71     [[ptBoolean,true]]),
72
73 msrNav([AmaxCrisisReminderPeriod],
74     [value,eq,[[ptInteger,1200]]],
75     [[ptBoolean,true]]),
76
77 msrNav([AvpStarted],
78     [],
79     [[ptBoolean,true]]),
80
81 msrNav([TheSystem],
82     [init,[AnextValueForAlertID,
83         AnextValueForCrisisID,
84         Aclock,
85         AcrisisReminderPeriod,
86         AmaxCrisisReminderPeriod,
87         Aclock,
88         AvpStarted]
89     ],
90     [[ptBoolean,true]]),
91
92/* PostF02*/
93 msrNav([AactMsrCreator],
94     [init,[],[[ptBoolean,true]]),
95
96
97 /* PostF03 */
98 msrVarCol(actComCompany,AqtyComCompanies,AactComCompanyCol),
99
100 msrNav(AactComCompanyCol,
101     [msrForAll,init,[],[[ptBoolean,true]]),
102
103
104 /* PostF04*/
105 msrNav([AactAdministrator],
106     [init,[],[[ptBoolean,true]]),
107
108
109 /* PostF05*/
110 msrVar(actActivator,AactActivator),
111 msrNav([AactActivator],
112     [init,[]],
```

```

113     [[ptBoolean,true]]),
114
115 /* PostF06 */
116 msrVar(ctAdministrator,ActAdministrator),
117 msrVar(dtLogin,AdtLogin),
118 msrVar(dtPassword,AdtPassword),
119
120 msrNav([AdtLogin],
121     [value,eq,[[ptString,'icrashadmin']]],
122     [[ptBoolean,true]]),
123
124 msrNav([AdtPassword],
125     [value,eq,[[ptString,'7WXC1359']]],
126     [[ptBoolean,true]]),
127
128 msrNav([ActAdministrator],
129     [init,[AdtLogin,AdtPassword]],
130     [[ptBoolean,true]]),
131
132 /* PostF07*/
133 msrNav([ActAdministrator],
134     [msmAtPost,rnactAuthenticated],
135     [AactAdministrator]),
136
137 /* Post Protocol:*/
138 /* PostP01 */
139 true
140 .

```

Listing 5.27: **Messir** (Prolog-oriented) implementation of the operation `oeCreateSystemAndEnvironment`.

Figure 5.4 shows all the concept model elements in the scope of the `oeCreateSystemAndEnvironment` operation

## 5.7 Environment - Out Interface Operation Scheme for `actPolice`

### 5.7.1 Operation Model for `oeCloseCrisis`

The `oeCloseCrisis` operation has the following properties:

<b>OPERATION</b>
<b><i>oeCloseCrisis</i></b>
sent to indicate that a crisis should be considered as closed.
<b>Parameters</b>
1 <b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis to close
<b>Return type</b>
ptBoolean
<b>Pre-Condition (protocol)</b>
PreP 1    the system is started PreP 2    the actor logged previously and did not log out ! (i.e. the associated <code>ctPolice</code> instance is considered logged)
<b>Pre-Condition (functional)</b>
PreF 1    it is supposed that there exist one <code>ctCrisis</code> instance with the same <code>id</code> attribute value as the one provided by the police actor who wants to close.

***continues in next page ...***

***... Operation table continuation***

<b><i>Post-Condition (functional)</i></b>
PostF 1 the ctCrisis class instance having the provided id is considered closed in the post state.
<b><i>Post-Condition (protocol)</i></b>
PostP 1 none

**5.7.2 Operation Model for oeGetCrisisSet**

The `oeGetCrisisSet` operation has the following properties:

<b>OPERATION</b>
<b><i>oeGetCrisisSet</i></b>
sent to request all the ctCrisis instances having a specific status.
<b><i>Parameters</i></b>
1 <b>AetCrisisStatus: etCrisisStatus</b> the status information used to determine the crisis to send back to the actor
<b><i>Return type</i></b>
ptBoolean
<b><i>Pre-Condition (protocol)</i></b>
PreP 1 the system is started
PreP 2 the actor logged previously and did not log out ! (i.e. the associated ctPolice instance is considered logged)
<b><i>Pre-Condition (functional)</i></b>
PreF 1 none
<b><i>Post-Condition (functional)</i></b>
PostF 1 the post state is the one obtained by satisfying the <code>isSentToPolice</code> predicate for each crisis having the provided status and for the actor sending the message <code>ieSendACrisis</code> . (cf. specification of <code>isSentToPolice</code> predicate given for the <code>ctCrisis</code> type).
<b><i>Post-Condition (protocol)</i></b>
PostP 1 none

**5.7.3 Operation Model for oeReportOnCrisis**

The `oeReportOnCrisis` operation has the following properties:

<b>OPERATION</b>
<b><i>oeReportOnCrisis</i></b>
sent to update the textual information available for a specific handled crisis.
<b><i>Parameters</i></b>
1 <b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis to report on
2 <b>AdtComment: dtComment</b> the textual information commenting the crisis
<b><i>Return type</i></b>
ptBoolean
<b><i>Pre-Condition (protocol)</i></b>
PreP 1 the system is started

***continues in next page ...***

**... Operation table continuation**

PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctPolice instance is considered logged)
--------	--

**Pre-Condition (functional)**

PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
--------	--

**Post-Condition (functional)**

PostF 1	the comment attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
---------	--

**Post-Condition (protocol)**

PostP 1	none
---------	------

**5.7.4 Operation Model for oeSetCrisisHandler**

The `oeSetCrisisHandler` operation has the following properties:

OPERATION	
<b><i>oeSetCrisisHandler</i></b>	
sent to declare himself as been the handler of a crisis having the specified id.	
<b>Parameters</b>	
1	<b>AdtCrisisID: dtCrisisID</b> the identification information used to determine the crisis
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctPolice instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	there exist one crisis having the given id in the pre-state.
<b>Post-Condition (functional)</b>	
PostF 1	the ctCrisis instance having the provided id is in handled status at poststate and is associated to the actor that sends the message (which himself is notified with a textual message as confirmation).
PostF 2	if the crisis was already handled at pre-state then the associated handler actor is notified about the change of handler for one of his crisis (n.b. it might be the same even if not relevant).
PostF 3	a message is sent to the communication company for any human related to an alert associated to the crisis. A human will receive as many messages as alerts he sent despite the fact that they might relate to the same crisis (i.e. one alert, one acknowledgement).
<b>Post-Condition (protocol)</b>	
PostP 1	none

**5.7.5 Operation Model for oeSetCrisisStatus**

The `oeSetCrisisStatus` operation has the following properties:

OPERATION
<i>continues in next page ...</i>

*... Operation table continuation*

<b>oeSetCrisisStatus</b>	sent to define the handling status of a specific crisis.
<b>Parameters</b>	
1 <b>AdtCrisisID: dtCrisisID</b>	the identification information used to determine the crisis
2 <b>AetCrisisStatus: etCrisisStatus</b>	the new status value
<b>Return type</b>	
ptBoolean	
<b>Pre-Condition (protocol)</b>	
PreP 1	the system is started
PreP 2	the actor logged previously and did not log out ! (i.e. the associated ctPolice instance is considered logged)
<b>Pre-Condition (functional)</b>	
PreF 1	it is supposed that there exist one crisis in the pre state having the given id.
<b>Post-Condition (functional)</b>	
PostF 1	the crisis status attribute of the crisis instance having the given id is replaced by the given one and the requesting actor is notified of this update.
<b>Post-Condition (protocol)</b>	
PostP 1	none

## 5.8 Environment - Actor Operation Scheme for actMsrCreator

### 5.8.1 Operation Model for init

The `init` operation has the following properties:

<b>OPERATION</b>
<b>init</b>
used to create an instance of the actor together with its interface instances and update the associations with the <code>ctState</code> instance.
<b>Return type</b>

## 5.9 Primary Types - Operation Schemes for Class ctAdministrator

### 5.9.1 Operation Model for init

The `init` operation has the following properties:

<b>OPERATION</b>
<b>init</b>
used to initialize the current object as a new instance of the <code>ctAdministrator</code> type.
<b>Parameters</b>
1 <b>Alogin: dtLogin</b> used to initialize the login field

*continues in next page ...*

**... Operation table continuation**

2	<b>Apwd:</b> dtPassword used to initialize the password field
<b>Return type</b>	
ptBoolean	
<b>Post-Condition (functional)</b>	
PostF 1	true iff the system poststate includes the current object as a new ctAdministrator instance having its login and password attributes equal to the one provided as parameters and its vpIsLogged attribute equal to false.

The listing 5.28 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{if
4 (
5 let Self:ctAdministrator in
6 /* Post F01 */
7 Self.login = Alogin
8 //Self.login(Alogin)
9 and Self.pwd = Apwd
10 and Self.vpIsLogged = false
11
12 /* Post F02 */
13 and (Self.oclisNew and self = Self)
14 )
15 then (result = true)
16 else (result = false)
17 endif}

```

Listing 5.28: **Messip** (MCL-oriented) specification of the operation *init*.

The listing 5.29 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctAdministrator,init,[Self,
7          Alogin,
8          Apwd],
9          Result):-
10(
11msrVar(ctAdministrator,Self),
12
13/* Post F01 */
14msrNav([Self],[login],[Alogin]),
15msrNav([Self],[pwd],[Apwd]),
16msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),
17
18/* Post F02 */
19 msrNav([Self],[msrIsNew],[Self])
20)
21-> Result = [ptBoolean,true]
22; Result = [ptBoolean,false]

```

Figure 5.4: oeCreateSystemAndEnvironment operation scope

23.

Listing 5.29: **Messip** (Prolog-oriented) implementation of the operation *init*.

## 5.10 Primary Types - Operation Schemes for Class ctAlert

### 5.10.1 Operation Model for init

The *init* operation has the following properties:

<b>OPERATION</b>	
<i>init</i>	
used to initialize the current object as a new instance of the ctAlert type.	
<b>Parameters</b>	
1	<b>Aid:</b> dtAlertID used to initialize the id field
2	<b>Astatus:</b> etAlertStatus used to initialize the status field
3	<b>Alocation:</b> dtGPSLocation used to initialize the location field
4	<b>Ainstant:</b> dtDateAndTime used to initialize the instant field
5	<b>Acomment:</b> dtComment used to initialize the comment field
<b>Return type</b>	
ptBoolean	
<b>Post-Condition (functional)</b>	
PostF 1	true iff the system poststate includes the current object as a new ctAlert instance having its attributes equal to the ones provided as parameters.

The listing 5.30 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{if
3  (
4  /* Post F01 */
5  let Self:ctAlert in
6  Self.id = Aid
7  and Self.status = Astatus
8  and Self.location = Alocation
9  and Self.instant = Ainstant
10 and Self.comment = Acomment
11 /* Post F02 */
12 and (Self.oclIsNew and self = Self)
13 )
14 then (result = true)
15 else (result = false)
16 endif}

```

Listing 5.30: **Messip** (MCL-oriented) specification of the operation *init*.

The listing 5.31 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctAlert,init,[Self,
7      Aid,
8      Astatus,
9      Alocation,
10     Ainstant,
11     Acomment],
12   Result):-!
13
14/* Post F01 */
15(
16msrVar(ctAlert,Self),
17
18msrNav([Self],[id],[Aid]),
19msrNav([Self],[status],[Astatus]),
20msrNav([Self],[location],[Alocation]),
21msrNav([Self],[instant],[Ainstant]),
22msrNav([Self],[comment],[Acomment]),
23
24/* Post F02 */
25 msrNav([Self],[msrIsNew],[Self])
26)
27-> Result = [ptBoolean,true]
28; Result = [ptBoolean,false]
29.

```

Listing 5.31: **Messip** (Prolog-oriented) implementation of the operation *init*.

### 5.10.2 Operation Model for `isSentToCoordinator`

The `isSentToCoordinator` operation has the following properties:

OPERATION
<b><i>isSentToCoordinator</i></b>
used to provide a given coordinator with current alert information.
<b>Parameters</b>
1 <b>AactCoordinator: actCoordinator</b> the message destination
<b>Return type</b>
ptBoolean
<b>Post-Condition (functional)</b>
PostF 1    true iff the message ieSendAnAlert is sent to the input interface of the given coordinator actor with the current alert as parameter value.

The listing 5.32 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{if
4 (

```

```

5 /* Post F01 */
6 AactCoordinator.rnInterfaceIN.ieSendAnAlert(self)
7 )
8 then (result = true)
9 else (result = false)
10 endif

```

Listing 5.32: **Messir** (MCL-oriented) specification of the operation *isSentToCoordinator*.

The listing 5.33 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctAlert,isSentToCoordinator,[self,AactCoordinator],
7   Result):- 
8
9/* Post F01 */
10(
11 msrNav([AactCoordinator],
12   [rnInterfaceIN,ieSendAnAlert,[self]],
13   [[ptBoolean,true]])
14)
15-> Result = [ptBoolean,true]
16; Result = [ptBoolean,false]
17.

```

Listing 5.33: **Messir** (Prolog-oriented) implementation of the operation *isSentToCoordinator*.

## 5.11 Primary Types - Operation Schemes for Class ctAuthenticated

### 5.11.1 Operation Model for init

The *init* operation has the following properties:

<b>OPERATION</b>	
<b>init</b>	used to initialize the current object as a new instance of the ctAuthenticated type.
<b>Parameters</b>	
1 <b>Alogin:</b> <b>dtLogin</b>	used to initialize the login field
2 <b>Apwd:</b> <b>dtPassword</b>	used to initialize the password field
<b>Return type</b>	
<b>ptBoolean</b>	
<b>Post-Condition (functional)</b>	
PostF 1	true iff the system poststate includes the current object as a new ctAuthenticated instance having its attributes equal to the ones provided as parameters.

The listing 5.34 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctAuthenticated,init,[Self,
7          Alogin,
8          Apwd],
9      Result):- 
10
11/* Post F01 */
12(
13msrVar(ctAuthenticated,Self),
14
15msrNav([Self],[login],[Alogin]),
16msrNav([Self],[pwd],[Apwd]),
17msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),
18
19/* Post F02 */
20 msrNav([Self],[msrIsNew],[Self])
21)
22-> Result = [ptBoolean,true]
23; Result = [ptBoolean,false]
24.
```

Listing 5.34: **Messip** (Prolog-oriented) implementation of the operation *init*.

## 5.12 Primary Types - Operation Schemes for Class ctCoordinator

### 5.12.1 Operation Model for init

The *init* operation has the following properties:

OPERATION
<i>init</i>
used to initialize the current object as a new instance of the ctCoordinator type.
<i>Parameters</i>
1 <b>Aid: dtCoordinatorID</b> used to initialize the id field 2 <b>Alogin: dtLogin</b> used to initialize the login field 3 <b>Apwd: dtPassword</b> used to initialize the password field
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1    true iff the system poststate includes the current object as a new ctCoordinator instance having its attributes equal to the ones provided as parameters.

The listing 5.35 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{if
```

```

4  (
5  /* Post F01 */
6 let Self:ctCoordinator in
7 Self.id = Aid
8 and Self.login = Alogin
9 and Self.pwd = Apwd
10 and Self.vpIsLogged = false
11 /* Post F02 */
12 and (Self.oclisNew and self = Self)
13 )
14 then (result = true)
15 else (result = false)
16 endif}

```

Listing 5.35: **Messip** (MCL-oriented) specification of the operation *init*.

The listing 5.36 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCoordinator,init,[Self,
7           Aid,
8           Alogin,
9           Apwd],
10      Result):- 
11
12/* Post F01 */
13(
14msrVar(ctCoordinator,Self),
15
16msrNav([Self],[id],[Aid]),
17msrNav([Self],[login],[Alogin]),
18msrNav([Self],[pwd],[Apwd]),
19msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),
20
21/* Post F02 */
22msrNav([Self],[msrIsNew],[Self])
23)
24-> Result = [ptBoolean,true]
25; Result = [ptBoolean,false]
26.

```

Listing 5.36: **Messip** (Prolog-oriented) implementation of the operation *init*.

## 5.13 Primary Types - Operation Schemes for Class ctCrisis

### 5.13.1 Operation Model for init

The *init* operation has the following properties:

OPERATION
<i>init</i>
used to initialize the current object as a new instance of the ctCrisis type.
<i>Parameters</i>
1      Aid: dtCrisisID used to initialize the id field

*continues in next page ...*

**... Operation table continuation**

2	<b>Atype:</b> etCrisisType used to initialize the type field
3	<b>Astatus:</b> etCrisisStatus used to initialize the status field
4	<b>Alocation:</b> dtGPSLocation used to initialize the location field
5	<b>Ainstant:</b> dtDateAndTime used to initialize the instant field
6	<b>Acomment:</b> dtComment used to initialize the comment field
<b>Return type</b>	
ptBoolean	
<b>Post-Condition (functional)</b>	
PostF 1	true iff the system poststate includes the current object as a new ctCrisis instance having its attributes equal to the ones provided as parameters.

The listing 5.37 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{if
3  (
4  /* Post F01 */
5  let Self:ctCrisis in
6  Self.id = Aid
7  and Self.type = Atype
8  and Self.status = Astatus
9  and Self.location = Alocation
10 and Self.instant = Ainstant
11 and Self.comment = Acomment
12 /* Post F02 */
13 and (Self.oclisNew and self = Self)
14 )
15 then (result = true)
16 else (result = false)
17 endif}

```

Listing 5.37: **Messip** (MCL-oriented) specification of the operation *init*.

The listing 5.38 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,init,[Self,
7  Aid,
8  Atype,
9  Astatus,
10 Alocation,
11 Ainstant,
12 Acomment],

```

```

13     Result):-  

14  

15 /* Post F01 */  

16 (  

17msrVar(ctCrisis,Self),  

18  

19msrNav([Self],[id],[Aid]),  

20msrNav([Self],[type],[Atype]),  

21msrNav([Self],[status],[Astatus]),  

22msrNav([Self],[location],[Allocation]),  

23msrNav([Self],[instant],[Ainstant]),  

24msrNav([Self],[comment],[Acomment]),  

25  

26 /* Post F02 */  

27 msrNav([Self],[msrIsNew],[Self])  

28 )  

29-> Result = [ptBoolean,true]  

30 ; Result = [ptBoolean,false]  

31 .

```

Listing 5.38: **Messip** (Prolog-oriented) implementation of the operation *init*.

### 5.13.2 Operation Model for handlingDelayPassed

The `handlingDelayPassed` operation has the following properties:

OPERATION
<b>handlingDelayPassed</b>
used to determine if the crisis stood too longly in a pending status since last reminder.
<b>Return type</b>
ptBoolean
<b>Post-Condition (functional)</b>
PostF 1 true iff the crisis is in pending status and if the duration between the current <code>ctState</code> clock information and the last reminder is greater than the crisis reminder period duration.

The listing 5.39 provides the **Messip** (MCL-oriented) specification of the operation.

```

1 /* Post Functional:*/  

2 postF{let TheSystem:ctState in  

3 let CurrentClockSecondsQty:dtInteger in  

4 let vpLastReminderSecondsQty:dtInteger in  

5 let CrisisReminderPeriod:dtSecond in  

6 if  

7 ( /* Post F01 */  

8   self.rnSystem = TheSystem  

9   and self.status = pending  

10  and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty  

11  and TheSystem.vpLastReminder.toSecondsQty() = vpLastReminderSecondsQty  

12  and TheSystem.crisisReminderPeriod = CrisisReminderPeriod  

13  and CurrentClockSecondsQty.sub(vpLastReminderSecondsQty).gt(CrisisReminderPeriod) = true  

14  )  

15  then (result = true)  

16  else (result = false)  

17 endif}

```

Listing 5.39: **Messip** (MCL-oriented) specification of the operation *handlingDelayPassed*.

The listing 5.40 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,handlingDelayPassed,[Self],
7      Result):- 
8
9/* Post F01 */
10(
11 msrVar(ctState,TheSystem),
12 msrVar(dtInteger,CurrentClockSecondsQty),
13 msrVar(dtInteger,LastReminderSecondsQty),
14 msrVar(dtSecond,CrisisReminderPeriod),
15
16 msrNav([Self],[rnSystem],[TheSystem]),
17
18 msrNav([Self],
19     [status],
20     [[etCrisisStatus,pending]]),
21
22 msrNav([TheSystem],
23     [clock,toSecondsQty,[],],
24     [CurrentClockSecondsQty]),
25
26 msrNav([TheSystem],
27     [vpLastReminder,toSecondsQty,[],],
28     [LastReminderSecondsQty]),
29
30 msrNav([TheSystem],
31     [crisisReminderPeriod],
32     [CrisisReminderPeriod]),
33
34 msrNav([CurrentClockSecondsQty],
35     [sub,[LastReminderSecondsQty],
36     gt, [CrisisReminderPeriod]
37     ],
38     [[ptBoolean,true]])
39
40)
41-> Result = [ptBoolean,true]
42; Result = [ptBoolean,false]
43.

```

Listing 5.40: **Messip** (Prolog-oriented) implementation of the operation *handlingDelayPassed*.

### 5.13.3 Operation Model for maxHandlingDelayPassed

The `maxHandlingDelayPassed` operation has the following properties:

OPERATION
<b><i>maxHandlingDelayPassed</i></b>
used to determine if the crisis stood too longly in a pending status since its creation.
<b><i>Return type</i></b>
ptBoolean
<b><i>Post-Condition (functional)</i></b>
PostF 1 true iff the crisis is in pending status and if the duration between the current ctState clock information and the crisis instant is greater than the maximum reminder period duration.

The listing 5.41 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheSystem:ctState in
3  let CurrentClockSecondsQty:dtInteger in
4  let CrisisInstantSecondsQty:dtInteger in
5  let MaxCrisisReminderPeriod:dtSecond in
6  if
7  ( /* Post F01 */
8  self.rnSystem = TheSystem
9  and self.status = pending
10 and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
11 and Self.instant.toSecondsQty() = CrisisInstantSecondsQty
12 and TheSystem.maxCrisisReminderPeriod = MaxCrisisReminderPeriod
13 and CurrentClockSecondsQty.sub(CrisisInstantSecondsQty)
14 .gt(MaxCrisisReminderPeriod)
15 )
16 )
17 then (result = true)
18 else (result = false)
19 endif}

```

Listing 5.41: **Messip** (MCL-oriented) specification of the operation *maxHandlingDelayPassed*.

The listing 5.42 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,maxHandlingDelayPassed,[Self],
7      Result):-
8
9/* Post F01 */
10(
11 msrVar(ctState,TheSystem),
12 msrVar(dtInteger,CurrentClockSecondsQty),
13 msrVar(dtInteger,CrisisInstantSecondsQty),
14 msrVar(dtSecond,MaxCrisisReminderPeriod),
15
16 msrNav([Self],[rnSystem],[TheSystem]),
17
18 msrNav([Self],
19         [status],
20         [[etCrisisStatus,pending]]),
21
22 msrNav([TheSystem],
23         [clock,toSecondsQty,[],],
24         [CurrentClockSecondsQty]),
25
26 msrNav([Self],
27         [instant,toSecondsQty,[],],
28         [CrisisInstantSecondsQty]),
29
30 msrNav([TheSystem],
31         [maxCrisisReminderPeriod],
32         [MaxCrisisReminderPeriod]),
33
34 msrNav([CurrentClockSecondsQty],
35         [sub,[CrisisInstantSecondsQty],
36         gt, [MaxCrisisReminderPeriod]
37         ],
38         [[ptBoolean,true]])
39

```

```

40)
41-> Result = [ptBoolean,true]
42; Result = [ptBoolean,false]
43.

```

Listing 5.42: **Mess1P** (Prolog-oriented) implementation of the operation *maxHandlingDelayPassed*.

#### 5.13.4 Operation Model for *isSentToCoordinator*

The *isSentToCoordinator* operation has the following properties:

OPERATION
<i>isSentToCoordinator</i>
used to provide a given coordinator with current crisis information.
<i>Parameters</i>
1        AactCoordinator: actCoordinator the message destination actor
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1    true iff the message ieSendACrisis is sent by the simulator to the input interface of the given coordinator actor with the current crisis as parameter value.

The listing 5.43 provides the **Mess1P** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{if
4 (
5 /* Post F01 */
6 AactCoordinator.rnInterfaceIN.ieSendACrisis(self)
7 )
8 then (result = true)
9 else (result = false)
10 endif}

```

Listing 5.43: **Mess1P** (MCL-oriented) specification of the operation *isSentToCoordinator*.

The listing 5.44 provides the **Mess1P** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,isSentToCoordinator,[Self,AactCoordinator],
7      Result):-_
8
9/* Post F01 */
10(
11 msrNav([AactCoordinator],
12        [rnInterfaceIN,ieSendACrisis,[Self]],
13        [[ptBoolean,true]])

```

```

14 )
15 -> Result = [ptBoolean,true]
16 ; Result = [ptBoolean,false]
17 .

```

Listing 5.44: **Messip** (Prolog-oriented) implementation of the operation *isSentToCoordinator*.

### 5.13.5 Operation Model for *isSentToPolice*

The *isSentToPolice* operation has the following properties:

<b>OPERATION</b>	
<i>isSentToPolice</i>	
used to provide a given coordinator with current crisis information.	
<i>Parameters</i>	
1	<b>AactPolice: actPolice</b> the message destination actor
<i>Return type</i>	
ptBoolean	
<i>Post-Condition (functional)</i>	
PostF 1	true iff the message <i>ieSendACrisis</i> is sent by the simulator to the input interface of the given police actor with the current crisis as parameter value.

The listing 5.45 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{if
3  (
4  /* Post F01 */
5  AactPolice.rnInterfaceIN.ieSendACrisis(self)
6  )
7  then (result = true)
8  else (result = false)
9  endif}
10

```

Listing 5.45: **Messip** (MCL-oriented) specification of the operation *isSentToPolice*.

### 5.13.6 Operation Model for *isAllocatedIfPossible*

The *isAllocatedIfPossible* operation has the following properties:

<b>OPERATION</b>	
<i>isAllocatedIfPossible</i>	
used to allocate a crisis to a coordinator if any or to alert the administrator of crisis waiting to be handled.	
<i>Return type</i>	
ptBoolean	
<i>Post-Condition (functional)</i>	

*continues in next page ...*

***... Operation table continuation***

- |         |  |
|---------|--|
| PostF 1 | true iff the duration between the crisis creation and the system's clock is greater than the maximum delay defined and   |
| PostF 2 | if there exist at least one coordinator then (a) the post state associates to the crisis any of the existing coordinators and (b) the coordinator is informed that he is now the handlers of the crisis whose ID is communicated |
| PostF 3 | else a message is sent to all known administrators to request creation of new coordinators.  |

The listing 5.46 provides the **Mess1P** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{if (
3  /* Post F01 */
4  self.maxHandlingDelayPassed()
5  and
6  if (TheSystem.rnactCoordinator->msrIsEmpty = false)
7  then (
8    /* Post F02 */
9    TheSystem.rnactCoordinator->msrAny(true) = TheCoordinatorActor
10   and TheCoordinatorActor.rnctCoordinator = TheCoordinator
11   and self@post.rnHandler = TheCoordinator
12   and self@post.status = handled
13   and self.id.value = TheCrisisIDptString
14   and 'You are now considered as handling the crisis having ID: '
15   .ptStringConcat(TheCrisisIDptString) = TheMessage
16   and TheCoordinatorActor.rnInterfaceIN^ieMessage(TheMessage)
17 )
18 )
19 else ( /* Post F03 */
20   TheSystem.rnactAdministrator
21   ->forAll(rnInterfaceIN.ieMessage('Please add new coordinators to handle pending crisis !'))
22 )
23 endif
24 )
25 then (result = true)
26 else (result = false)
27 endif}

```

Listing 5.46: **Mess1P** (MCL-oriented) specification of the operation *isAllocatedIfPossible*.

The listing 5.47 provides the **Mess1P** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,isAllocatedIfPossible,[Self],
7      Result):-(
8(
9  msrVar(ctState,TheSystem),
10 msrNav([Self],[rnSystem],[TheSystem]),
11
12 msrVar(actCoordinator,TheCoordinatorActor),
13 msrVar(ctCoordinator,TheCoordinator),
14 msrVar(ptString,TheMessage),
15 msrVar(ptString,TheCrisisIDptString),
16

```

```

17  (
18  /* Post F01 */
19  msrNav([Self],
20  [maxHandlingDelayPassed,[[]],
21  [[ptBoolean,true]]),
22
23  ( msrNav([TheSystem],
24  [rnactCoordinator,msrIsEmpty],
25  [[ptBoolean,false]])
26  -> (
27  /* Post F02 */
28  msrNav([TheSystem],
29  [rnactCoordinator,msrAny,msrTrue],
30  [TheCoordinatorActor]),
31
32  msrNav([TheCoordinatorActor],
33  [rnctCoordinator],
34  [TheCoordinator]),
35
36  msrNav([Self],
37  [msmAtPost,rnHandler],
38  [TheCoordinator]),
39
40  msrNav([Self],
41  [msmAtPost,status],
42  [[etCrisisStatus,handled]]),
43
44  msrNav([Self],
45  [id,value],
46  [TheCrisisIDptString]),
47
48  msrNav([[ptString,'You are now considered as handling the crisis having ID: ']],
49  [ptStringConcat,[TheCrisisIDptString]],
50  [TheMessage]),
51
52  msrNav([TheCoordinatorActor],
53  [rnInterfaceIN,
54  ieMessage,[TheMessage]
55  ],
56  [[ptBoolean,true]]))
57  )
58 ; ( /* Post F03 */
59  msrNav([TheSystem],
60  [rnactAdministrator,msrForAll,rnInterfaceIN,
61  ieMessage,[[ptString,'Please add new coordinators to handle pending crisis !']]],
62  [[ptBoolean,true]])
63  )
64  )
65  )
66 )
67-> Result = [ptBoolean,true]
68 ; Result = [ptBoolean,false]
69 .

```

Listing 5.47: **Messir** (Prolog-oriented) implementation of the operation *isAllocatedIfPossible*.

## 5.14 Primary Types - Operation Schemes for Class ctHuman

### 5.14.1 Operation Model for init

The *init* operation has the following properties:

<b>OPERATION</b>
<i>init</i>

used to initialize the current object as a new instance of the ctHuman type.

*continues in next page ...*

*... Operation table continuation*

<i>Parameters</i>	
1	<b>Aid:</b> dtPhoneNumber used to initialize the id field
2	<b>Akind:</b> etHumanKind used to initialize the kind field
<i>Return type</i>	
ptBoolean	
<i>Post-Condition (functional)</i>	
PostF 1 true iff the system poststate includes the current object as a new ctHuman instance having its attributes equal to the ones provided as parameters.	

The listing 5.48 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{if
4 (
5 /* Post F01 */
6 let Self:ctHuman in
7
8 Self.id = Aid
9 and Self.kind = Akind
10
11 /* Post F02 */
12 and (Self.oclisNew and self = Self)
13 )
14 then (result = true)
15 else (result = false)
16 endif}

```

Listing 5.48: **Messip** (MCL-oriented) specification of the operation *init*.

The listing 5.49 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctHuman,init,[Self,
7
8
9 Result):- 
10
11/* Post F01 */
12(
13msrVar(ctHuman,Self),
14
15msrNav([Self],[id],[Aid]),
16msrNav([Self],[kind],[Akind]),
17
18/* Post F02 */
19 msrNav([Self],[msrIsNew],[Self])
20)
21-> Result = [ptBoolean,true]

```

```

22; Result = [ptBoolean, false]
23.

```

Listing 5.49: **Messip** (Prolog-oriented) implementation of the operation *init*.

### 5.14.2 Operation Model for `isAcknowledged`

The `isAcknowledged` operation has the following properties:

<b>OPERATION</b>	
<i>isAcknowledged</i>	
	used to specify the property of having sent an alert acknowledge message to the human having declared the alert through its own communication company.
<i>Return type</i>	
ptBoolean	
<i>Post-Condition (functional)</i>	
PostF 1	true iff the message <code>ieSmsSend</code> is sent to the related input interface of the related communication company actor with the human phone number and the generic message 'The handling of your alert by our services is in progress !'

The listing 5.50 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6msrop(ctHuman,isAcknowledged,[Self],Result):-
7
8/* Post F01 */
9(msrVar(dtPhoneNumber,AdtPhoneNumber),
10 msrVar(dtSMS,AdtSMS),
11
12 msrNav([Self],
13     [id,eq,[AdtPhoneNumber]],
14     [[ptBoolean,true]]),
15 msrNav([AdtSMS],
16     [value,eq,[[ptString,'The handling of your alert by our services is in progress !']]],
17     [[ptBoolean,true]]),
18 msrNav([Self],
19     [rnactComCompany,rnInterfaceIN,ieSmsSend,[AdtPhoneNumber,AdtSMS]],
20     [[ptBoolean,true]]),
21 )
22-> Result = [ptBoolean,true]
23; Result = [ptBoolean,false]
24.

```

Listing 5.50: **Messip** (Prolog-oriented) implementation of the operation *isAcknowledged*.

## 5.15 Primary Types - Operation Schemes for Class `ctPolice`

### 5.15.1 Operation Model for `init`

The `init` operation has the following properties:

<b>OPERATION</b>	
<i>init</i>	
used to initialize the current object as a new instance of the ctPolice type.	
<i>Parameters</i>	
1	<b>Aid: dtPoliceID</b> used to initialize the id field
2	<b>Alogin: dtLogin</b> used to initialize the login field
3	<b>Apwd: dtPassword</b> used to initialize the password field
<i>Return type</i>	
ptBoolean	
<i>Post-Condition (functional)</i>	
PostF 1	true iff the system poststate includes the current object as a new ctPolice instance having its attributes equal to the ones provided as parameters.

The listing 5.51 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{if
3  (
4  /* Post F01 */
5  let Self:ctPolice in
6  Self.id = Aid
7  and Self.login = Alogin
8  and Self.pwd = Apwd
9  and Self.vpIsLogged = false
10 /* Post F02 */
11 and (Self.ocliIsNew and self = Self)
12 )
13 )
14 then (result = true)
15 else (result = false)
16 endif}

```

Listing 5.51: **Messip** (MCL-oriented) specification of the operation *init*.

## 5.16 Primary Types - Operation Schemes for Class ctState

### 5.16.1 Operation Model for init

The *init* operation has the following properties:

<b>OPERATION</b>	
<i>init</i>	
used to initialize the current object as a new instance of the ctState type.	
<i>Parameters</i>	
1	<b>AnextValueForAlertID: dtInteger</b> used to initialize the nextValueForAlertID field
2	<b>AnextValueForCrisisID: dtInteger</b>

*continues in next page ...*

**... Operation table continuation**

3	used to initialize the nextValueForCrisisID field
4	<b>Aclock: dtDateAndTime</b> used to initialize the clock field
5	<b>AcrisisReminderPeriod: dtSecond</b> used to initialize the crisisReminderPeriod field
6	<b>AmaxCrisisReminderPeriod: dtSecond</b> used to initialize the maxCrisisReminderPeriod field
7	<b>AvpLastReminder: dtDateAndTime</b> used to initialize the vpLastReminder field
	<b>AvpStarted: ptBoolean</b> used to initialize the vpStarted field

***Return type***

ptBoolean

***Post-Condition (functional)***

PostF 1	true iff the system poststate includes the current object as a new ctState instance having its attributes equal to the ones provided as parameters.
---------	---

The listing 5.52 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{if
3  (
4    /* Post F01 */
5  let Self:ctState in
6
7
8  Self.nextValueForAlertID = AnextValueForAlertID
9  and Self.nextValueForCrisisID = AnextValueForCrisisID
10 and Self.clock = Aclock
11 and Self.crisisReminderPeriod = AcrisisReminderPeriod
12 and Self.maxCrisisReminderPeriod = AmaxCrisisReminderPeriod
13 and Self.vpLastReminder = AvpLastReminder
14 and Self.vpStarted = AvpStarted
15
16 and (Self.ocliIsNew and self = Self)
17 )
18 then (result = true)
19 else (result = false)
20 endif}

```

Listing 5.52: **Messip** (MCL-oriented) specification of the operation *init*.

The listing 5.53 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctState,init,[Self,
7           AnextValueForAlertID,
8           AnextValueForCrisisID,

```

```

9      Aclock,
10     AcrisisReminderPeriod,
11     AmaxCrisisReminderPeriod,
12     AvpLastReminder,
13     AvpStarted],
14 Result):- 
15 /* Post F01 */
17(
18 msrVar(ctState,Self),
19
20 msrNav([Self],[nextValueForAlertID],[AnextValueForAlertID]),
21 msrNav([Self],[nextValueForCrisisID],[AnextValueForCrisisID]),
22 msrNav([Self],[clock],[Aclock]),
23 msrNav([Self],[crisisReminderPeriod],[AcrisisReminderPeriod]),
24 msrNav([Self],[maxCrisisReminderPeriod],[AmaxCrisisReminderPeriod]),
25 msrNav([Self],[vpLastReminder],[AvpLastReminder]),
26 msrNav([Self],[vpStarted],[AvpStarted]),
27
28 msrNav([Self],[msrIsNew],[Self])
29)
30-> Result = [ptBoolean,true]
31; Result = [ptBoolean,false]
32.
```

Listing 5.53: **Messir** (Prolog-oriented) implementation of the operation *init*.

## 5.17 Primary Types - Operation Schemes for Datatype dtAlertID

### 5.17.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid alert identifiers.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 if the length of the value attribute of a dtAlertID is a ptInteger greater than zero and lower or equal to 20 then the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.54 provides the **Messir** (MCL-oriented) specification of the operation.

```

1 /* Post Functional:*/
2 postF{let TheResult: ptBoolean in
3   (
4     if
5       ( AdtValue.value.length().gt(0)
6         and AdtValue.value.length().leq(20)
7       )
8       then (TheResult = true)
9       else (TheResult = false)
10      endif
11      result = TheResult
12    )}
```

Listing 5.54: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.55 provides the **Mess1P** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(dtAlertID,is,[AdtValue],Result):-
7% msd01
8msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,length,[],gt,[[ptInteger,0]]],
12   [[ptBoolean,true]]),
13   msrNav([AdtValue],
14   [value,length,[],leq,[[ptInteger,20]]],
15   [[ptBoolean,true]])
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20 TheResult = Result
21 .
22
23/*
24| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[[]]]],
25msrNav([X],[is,[],[Result]).
26
27X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[[]]]],
28Result = [ptBoolean,true] ?
29
30yes
31
32| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'012345678901234567890123456789']]],[[]]]],
33msrNav([X],[is,[],[Result]).
34
35X = [dtAlertID,[],[[dtString,[[value,[ptString,'012345678901234567890123456789']]],[[]]]],
36Result = [ptBoolean,false] ?
37
38yes
39*/

```

Listing 5.55: **Mess1P** (Prolog-oriented) implementation of the operation *is*.

## 5.18 Primary Types - Operation Schemes for Datatype dtComment

### 5.18.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid comments.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true iff the length of the string value is not more than 160 characters.

The listing 5.56 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    ( if
4      ( MaxLength = 160
5        and AdtValue.value.length().leq(MaxLength)
6      )
7    then (TheResult = true)
8    else (TheResult = false)
9  endif
10  result = TheResult
11 }
12 }
```

Listing 5.56: **Messip** (MCL-oriented) specification of the operation *is*.

The listing 5.57 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%% dtComment
6
7%msd01
8msrop(dtComment,is,[AdtValue],Result):-%
9  msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MaxLength),
11 (
12   (
13     (
14       MaxLength = [ptInteger,160],
15       msrNav([AdtValue],
16           [value,length,[],leq,[MaxLength]],
17           [[ptBoolean,true]])
18     )
19     -> TheResult = [ptBoolean,true]
20     ; TheResult = [ptBoolean,false]
21   )
22 ),
23 Result = TheResult
24 .
25
26/*
27 | ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[[]]]],%
28 msrNav([X],[is,[],[Result]]).
29 X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[[]]]],%
30 Result = [ptBoolean,true] ?
31 yes
32
33 | ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog
34 to go to the skate park because my friends called me on my mobile phone and told me that a skate
35 star was doing triple back flips.']]],[[]]]],%
36 msrNav([X],[is,[],[Result]]).
37 X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog to go
38 to the skate park because my friends called me on my mobile phone and told me that a skate star
39 was doing triple back flips.']]],[[]]]],%
40 Result = [ptBoolean,false] ?
41 yes
42 */
```

Listing 5.57: **Messip** (Prolog-oriented) implementation of the operation *is*.

## 5.19 Primary Types - Operation Schemes for Datatype dtCoordinatorID

### 5.19.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
<i>is</i>
used to determine which string are considered as valid alert identifiers.
<i>Return type</i>
<code>ptBoolean</code>
<i>Post-Condition (functional)</i>
PostF 1 if the length of the value attribute of a <code>dtCoordinatorID</code> is a <code>ptInteger</code> greater than zero and lower or equal to 5 than the operation returns the <code>ptBoolean</code> true, else the <code>ptBoolean</code> false.

The listing 5.58 provides the **Messir** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3      ( if
4          ( AdtValue.value.length().gt(0)
5              and AdtValue.value.length().leq(5)
6          )
7      then (TheResult = true)
8      else (TheResult = false)
9      endif
10     result = TheResult
11  )}
```

Listing 5.58: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.59 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(dtCoordinatorID,is,[AdtValue],Result):-
7% msd01
8 msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,length,[],gt,[[ptInteger,0]]],
12   [[ptBoolean,true]]),
13   msrNav([AdtValue],
14   [value,length,[],leq,[[ptInteger,5]]],
15   [[ptBoolean,true]])
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20 TheResult = Result
```

21.

Listing 5.59: **Mess1P** (Prolog-oriented) implementation of the operation *is*.

## 5.20 Primary Types - Operation Schemes for Datatype dtCrisisID

### 5.20.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid crisis identifiers.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 if the length of the value attribute of a dtCrisisID is a ptInteger greater than zero and lower or equal to 10 than the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.60 provides the **Mess1P** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    ( if
4      ( AdtValue.value.length().gt(0)
5        and AdtValue.value.length().leq(10)
6      )
7      then (TheResult = true)
8      else (TheResult = false)
9    endif
10   result = TheResult
11 }
12 }
```

Listing 5.60: **Mess1P** (MCL-oriented) specification of the operation *is*.

The listing 5.61 provides the **Mess1P** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(dtCrisisID,is,[AdtValue],Result):-%
7% msd01
8 msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,length,[],gt,[[ptInteger,0]]],
12   [[ptBoolean,true]]),
13   msrNav([AdtValue],
14   [value,length,[],leq,[[ptInteger,10]]],
15   [[ptBoolean,true]])
```

```

16  )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20 TheResult = Result
21 .
22 /*
23 | ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]]],[],[],[],[]]],
24 msrNav([X],[is,[],[Result]).
25 X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]]],[],[],[],[]]],
26 Result = [ptBoolean,true] ?
27 yes
28
29 | ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]]],[],[],[],[]]],
30 msrNav([X],[is,[],[Result]).
31 X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]]],[],[],[],[]]],
32 Result = [ptBoolean,false] ?
33 yes
34 */

```

Listing 5.61: **Messir** (Prolog-oriented) implementation of the operation *is*.

## 5.21 Primary Types - Operation Schemes for Datatype dtGPSLocation

### 5.21.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which couples are considered as valid dtGPSLocation values.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true if both latitude and longitude are valid values according to their <i>is</i> operation.

The listing 5.62 provides the **Messir** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{let TheResult: ptBoolean in
4   ( if
5     ( AdtValue.latitude.is()
6     and AdtValue.longitude.is
7     )
8     then (TheResult = true)
9     else (TheResult = false)
10    endif
11    result = TheResult
12  )

```

Listing 5.62: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.63 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% dtPhoneNumber
7
8% msd01
9msrop(dtGPSLocation,is,[AdtValue],Result):-
10msrVar(ptBoolean,TheResult),
11(
12  (
13    msrNav([AdtValue],
14      [latitude,is,[]],
15      [[ptBoolean,true]]),
16    msrNav([AdtValue],
17      [longitude,is,[]],
18      [[ptBoolean,true]]))
19 )
20 -> TheResult = [ptBoolean,true]
21 ; TheResult = [ptBoolean,false]
22),
23
24 Result = TheResult
25.

```

Listing 5.63: **Messip** (Prolog-oriented) implementation of the operation *is*.

### 5.21.2 Operation Model for *isNearTo*

The *isNearTo* operation has the following properties:

OPERATION
<b><i>isNearTo</i></b>
used to determine if locations are considered enough close to be treated as equivalent in the application domain context. In the context of the iCrash system, we compute the distance between two GPS locations using the following Haversine formula. (more details can be found at: <a href="http://www.movable-type.co.uk/scripts/latlong.html">http://www.movable-type.co.uk/scripts/latlong.html</a> and <a href="http://www.gpsvisualizer.com/calculators#distance">http://www.gpsvisualizer.com/calculators#distance</a> )
<b>Parameters</b>
1 <b>AGPSLocation: dtGPSLocation</b> the GPS location to be compared to.
<b>Return type</b>
ptBoolean
<b>Post-Condition (functional)</b>
PostF 1   if the Haversine formula ( $\text{ACOS}(\text{SIN}(\text{lat1}) * \text{SIN}(\text{lat2}) + \text{COS}(\text{lat1}) * \text{COS}(\text{lat2}) * \text{COS}(\text{lon2-lon1})) * 6371$ , in which latitudes and longitudes are in radians applied to the two dtGPS coordinates is lower to 100 meters) then the predicate is true and false otherwise.

The listing 5.64 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/

```

```

3  postF{let TheResult: ptBoolean in true
4      let EarthRadius: dtReal in
5      let MaxDistance: dtReal in
6      let ComparedLatitude: dtLatitude in
7      let ComparedLongitude: dtLongitude in
8      let R1: dtReal in let R1a: dtReal in
9      let R2: dtReal in let R2a: dtReal in
10
11     ( if
12         ( EarthRadius.value = 6371
13           and MaxDistance.value = 100
14
15           and AdtValue.latitude = ComparedLatitude
16           and AdtValue.longitude = ComparedLongitude
17           and Self.latitude.sin() = R1a
18           and AdtValue.latitude.sin().mul(R1a) = R1
19           and Self.latitude.cos() = R2a
20           and AdtValue.latitude.cos().mul(R2a) = R2
21
22           and AdtValue.longitude = ComparedLongitude
23           and Self.longitude.sub(ComparedLongitude).cos().mul(R2)
24             .add(R1).acos().mul(EarthRadius).sub(MaxDistance)
25             .value.leq(0)
26     )
27     then (TheResult = true)
28   else (TheResult = false)
29   endif
30   result = TheResult
31 )}
```

Listing 5.64: **Messir** (MCL-oriented) specification of the operation *isNearTo*.

The listing 5.65 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% dtGPSLocation
7
8msrop(dtGPSLocation,isNearTo,[Self,AdtValue],Result):-%
9msrVar(ptBoolean,TheResult),
10msrVar(dtReal,EarthRadius),
11msrVar(dtReal,MaxDistance),
12
13msrVar(dtLatitude,ComparedLatitude),
14msrVar(dtLongitude,ComparedLongitude),
15
16msrVar(dtReal,R1),msrVar(dtReal,R1a),
17msrVar(dtReal,R2),msrVar(dtReal,R2a),
18
19(
20  (
21    (
22      % msd01
23      msrNav([EarthRadius],[value],[[ptReal,6371]]),
24      msrNav([MaxDistance],[value],[[ptReal,100]]),
25
26      msrNav([AdtValue],[latitude],[ComparedLatitude]),
27      msrNav([AdtValue],[longitude],[ComparedLongitude]),
28
29      msrNav([Self],[latitude,sin,[],[R1a]]),
30      msrNav([AdtValue],[latitude,sin,[],mul,[R1a]],[],[R1]),
31
32      msrNav([Self],[latitude,cos,[],[R2a]]),
```

```

33 msrNav([AdtValue],[latitude,cos,[],mul,[R2a]],[R2]),
34
35 msrNav([AdtValue],[longitude],[ComparedLongitude]),
36 msrNav([Self],[longitude,sub,[ComparedLongitude],cos,[],mul,[R2],
37      add,[R1],
38      acos,[],
39      mul,[EarthRadius],
40      sub,[MaxDistance],
41      value,leq,[[ptReal,0]]],
42      [[ptBoolean,true]])
43 )
44 -> TheResult = [ptBoolean,true]
45 ; TheResult = [ptBoolean,false]
46 )
47),
48 Result = TheResult
49.

```

Listing 5.65: **Messir** (Prolog-oriented) implementation of the operation *isNearTo*.

## 5.22 Primary Types - Operation Schemes for Datatype dtLatitude

### 5.22.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid dtLatitude.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1     is true if the value is a real in the interval [-90.0 , +90.0].

The listing 5.66 provides the **Messir** (MCL-oriented) specification of the operation.

```

1 /* Post Functional:*/
2 postF{let TheResult: ptBoolean in
3   ( if
4     ( AdtValue.value.geq(-90.0)
5       and AdtValue.value.leq(+90.0)
6     )
7     then (TheResult = true)
8     else (TheResult = false)
9   endif
10   result = TheResult
11 }
12 }

```

Listing 5.66: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.67 provides the **Messir** (Prolog-oriented) implementation of the operation.

%%%%%%%%%%%%%

```

2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6% msd01
7msrop(dtLatitude,is,[AdtValue],Result):-  

8msrVar(ptBoolean,TheResult),  

9(
10 ( msrNav([AdtValue],  

11   [value,geq,[[ptReal,-90.0]]],  

12   [[ptBoolean,true]]),  

13 msrNav([AdtValue],  

14   [value,leq,[[ptReal,+90.0]]],  

15   [[ptBoolean,true]]))  

16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20Result = TheResult
21.

```

Listing 5.67: **Messir** (Prolog-oriented) implementation of the operation *is*.

## 5.23 Primary Types - Operation Schemes for Datatype dtLogin

### 5.23.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid dtLogin.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1    is true if the length of the string value is not more than 20 characters.

The listing 5.68 provides the **Messir** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{let TheResult: ptBoolean in
4   let MaxLength: ptInteger in
5   ( if
6     ( MaxLength = 20
7       and AdtValue.value.length().leq(MaxLength)
8     )
9     then (TheResult = true)
10    else (TheResult = false)
11    endif
12    result = TheResult
13  )

```

Listing 5.68: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.69 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5%% dtComment
6
7%msd01
8msrop(dtLogin,is,[AdtValue],Result) :-
9 msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MaxLength),
11(
12 (
13 (
14     MaxLength = [ptInteger,20],
15     msrNav([AdtValue],
16         [value,length,[],leq,[MaxLength]],
17         [[ptBoolean,true]]))
18 )
19     -> TheResult = [ptBoolean,true]
20 ; TheResult = [ptBoolean,false]
21 )
22),
23 Result = TheResult
24.
25/*
26| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[[]]]],
27msrNav([X],[is,[],[Result]]).
28X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[[]]]],
29Result = [ptBoolean,true] ?
30yes
31
32| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[[]]]],
33msrNav([X],[is,[],[Result]]).
34X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[[]]]],
35Result = [ptBoolean,false] ?
36yes
37*/

```

Listing 5.69: **Messip** (Prolog-oriented) implementation of the operation *is*.

## 5.24 Primary Types - Operation Schemes for Datatype dtLongitude

### 5.24.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid dtLongitude.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1     is true if the value is a real in the interval [-180.0 , +180.0].

The listing 5.70 provides the **Messir** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    (
4      if
5        ( AdtValue.value.geq(-180.0)
6          and AdtValue.value.leq(+180.0)
7        )
8        then (TheResult = true)
9        else (TheResult = false)
10      endif
11      result = TheResult
12    )}
```

Listing 5.70: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.71 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% dtPhoneNumber
7
8% msd01
9msrop(dtLongitude,is,[AdtValue],Result):-%
10msrVar(ptBoolean,TheResult),
11(
12  ( msrNav([AdtValue],
13    [value,geq,[[ptReal,-180.0]]],
14    [[ptBoolean,true]]),
15  msrNav([AdtValue],
16    [value,leq,[[ptReal,+180.0]]],
17    [[ptBoolean,true]])
18  )
19  -> (TheResult = [ptBoolean,true])
20 ; (TheResult = [ptBoolean,false])
21 ),
22
23 Result = TheResult
24 .
```

Listing 5.71: **Messir** (Prolog-oriented) implementation of the operation *is*.

## 5.25 Primary Types - Operation Schemes for Datatype dtPassword

### 5.25.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid dtPassword.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1    is true of the length of the string value is at least 6 characters long.

The listing 5.72 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    let MinLength: ptInteger in
4      if
5        ( MinLength = 6
6          and AdtValue.value.length().geq(MinLength)
7        )
8      then (TheResult = true)
9      else (TheResult = false)
10    endif
11    result = TheResult
12  }
13 }
```

Listing 5.72: **Messip** (MCL-oriented) specification of the operation *is*.

The listing 5.73 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5% dtComment
6
7%msd01
8msrop(dtPassword,is,[AdtValue],Result):-%
9 msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MinLength),
11(
12  (
13    (
14      MinLength = [ptInteger,6],
15      msrNav([AdtValue],
16          [value,length,[],geq,[MinLength]],
17          [[ptBoolean,true]])
18    )
19    -> TheResult = [ptBoolean,true]
20    ; TheResult = [ptBoolean,false]
21  )
22),
23 Result = TheResult
24.
25/*
26| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[[]]]],%
27msrNav([X],[is,[],[Result]).
28X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[[]]]],%
29Result = [ptBoolean,true] ?
30yes
31
32| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[[]]]],%
33msrNav([X],[is,[],[Result]).
34X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[[]]]],%
35Result = [ptBoolean,false] ?
36yes
37*/
```

Listing 5.73: **Messip** (Prolog-oriented) implementation of the operation *is*.

## 5.26 Primary Types - Operation Schemes for Datatype dtPhoneNumber

### 5.26.1 Operation Model for is

The `is` operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid dtPhoneNumber.
<i>Return type</i>
<code>ptBoolean</code>
<i>Post-Condition (functional)</i>
PostF 1    is true if the length of the string value is from 4 to 30 characters. No standard is applied !

The listing 5.74 provides the **Messir** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2 postF{let TheResult: ptBoolean in
3   ( if
4     ( AdtValue.value.length().gt(4)
5       and AdtValue.value.length().leq(30)
6     )
7   then (TheResult = true)
8   else (TheResult = false)
9   endif
10  result = TheResult
11 )}
```

Listing 5.74: **Messir** (MCL-oriented) specification of the operation `is`.

The listing 5.75 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% dtPhoneNumber
7
8% msd01
9msrop(dtPhoneNumber,is,[AdtValue],Result):-
10msrVar(ptBoolean,TheResult),
11(
12  ( msrNav([AdtValue],
13    [value,length,[],gt,[[ptInteger,4]]],
14    [[ptBoolean,true]]),
15    msrNav([AdtValue],
16    [value,length,[],leq,[[ptInteger,30]]],
17    [[ptBoolean,true]])
18  )
19
20 -> TheResult = [ptBoolean,true]
21 ; TheResult = [ptBoolean,false]
```

```

22),
23 Result = TheResult
24.
25/*
26| ?- X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00')]]],[],[]]],
27msrNav([X],[is,[],[Result]]).
28
29X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00')]]],[],[]]],
30
31Result = [ptBoolean,true] ?
32
33yes
34*/

```

Listing 5.75: **Messip** (Prolog-oriented) implementation of the operation *is*.

## 5.27 Primary Types - Operation Schemes for Datatype dtPoliceID

### 5.27.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which string are considered as valid crisis identifiers.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 if the length of the value attribute of a dtCoordinatorID is a ptInteger greater than zero and lower or equal to 5 than the operation returns the ptBoolean true, else the ptBoolean false.

The listing 5.76 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{let TheResult: ptBoolean in
4   ( if
5     ( AdtValue.value.length().gt(0)
6       and AdtValue.value.length().leq(5)
7     )
8     then (TheResult = true)
9     else (TheResult = false)
10    endif
11    result = TheResult
12  }

```

Listing 5.76: **Messip** (MCL-oriented) specification of the operation *is*.

## 5.28 Primary Types - Operation Schemes for Enumeration etAlertStatus

### 5.28.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which litteral belongs to the enumeration.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true iff the value is equal to one of the following values: pending, valid, invalid

The listing 5.77 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    if
4      ( self = pending
5       or self = valid
6       or self = invalid
7     )
8     then (TheResult = true)
9     else (TheResult = false)
10    endif
11    result = TheResult
12  }
13 }
```

Listing 5.77: **Messip** (MCL-oriented) specification of the operation *is*.

The listing 5.78 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% etAlertStatus
7
8 msd01
9msrop(etAlertStatus,is,[AdtValue],Result):- 
10msrVar(ptBoolean,TheResult),
11(
12  (
13    member(AdtValue,[pending, valid, invalid])
14  )
15  -> TheResult = [ptBoolean,true]
16  ; TheResult = [ptBoolean,false]
17),
18 Result = TheResult
19.
```

Listing 5.78: **Messip** (Prolog-oriented) implementation of the operation *is*.

## 5.29 Primary Types - Operation Schemes for Enumeration etCrisisStatus

### 5.29.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which litteral belongs to the enumeration.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true iff the value is equal to one of the following values: pending, handled, solved, closed.

The listing 5.79 provides the **Messip** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    ( if
4      ( self = pending
5        or self = handled
6        or self = solved
7        or self = closed
8      )
9    then (TheResult = true)
10   else (TheResult = false)
11  endif
12  result = TheResult
13 )}
```

Listing 5.79: **Messip** (MCL-oriented) specification of the operation *is*.

The listing 5.80 provides the **Messip** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% etCrisisStatus
7
8% msd01
9msrop(etCrisisStatus,is,[AdtValue],Result):-
10msrVar(ptBoolean,TheResult),
11(
12  (
13    member(AdtValue,[pending, handled, solved, closed])
14  )
15  -> TheResult = [ptBoolean,true]
16 ; TheResult = [ptBoolean,false]
17),
18 Result = TheResult
```

19.

Listing 5.80: **Messir** (Prolog-oriented) implementation of the operation *is*.

## 5.30 Primary Types - Operation Schemes for Enumeration etCrisisType

### 5.30.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which litteral belongs to the enumeration.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true iff the value is equal to one of the following values: small, medium, huge

The listing 5.81 provides the **Messir** (MCL-oriented) specification of the operation.

```

1  /* Post Functional:*/
2  postF{let TheResult: ptBoolean in
3    ( if
4      ( self = small
5        or self = medium
6        or self = huge
7      )
8    then (TheResult = true)
9    else (TheResult = false)
10   endif
11   result = TheResult
12 }

```

Listing 5.81: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.82 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% etCrisisType
7
8% msd01
9msrop(etCrisisType,is,[AdtValue],Result):-%
10msrVar(ptBoolean,TheResult),
11(
12  (
13    member(AdtValue,[small, medium, huge])
14 )

```

```

15 -> TheResult = [ptBoolean,true]
16 ; TheResult = [ptBoolean,false]
17),
18 Result = TheResult
19.
```

Listing 5.82: **Messir** (Prolog-oriented) implementation of the operation *is*.

## 5.31 Primary Types - Operation Schemes for Enumeration etHumanKind

### 5.31.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which literal belongs to the enumeration.
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true iff the value is equal to one of the following values: <code>witness, victim, anonymous</code>

The listing 5.83 provides the **Messir** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{let TheResult: ptBoolean in
4   ( if
5     ( self = witness
6     or self = victim
7     or self = anonymous
8   )
9   then (TheResult = true)
10  else (TheResult = false)
11  endif
12  result = TheResult
13 }
```

Listing 5.83: **Messir** (MCL-oriented) specification of the operation *is*.

The listing 5.84 provides the **Messir** (Prolog-oriented) implementation of the operation.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% etHumanKind
7
8% msd01
9msrop(etHumanKind,is,[AdtValue],Result):-
10msrVar(ptBoolean,TheResult),
```

```

11 (
12   (
13     member(AdtValue,[witness,victim,anonymous])
14   )
15 -> TheResult = [ptBoolean,true]
16 ; TheResult = [ptBoolean,false]
17 ),
18 Result = TheResult
19 .

```

Listing 5.84: **Messip** (Prolog-oriented) implementation of the operation *is*.

## 5.32 Secondary Types - Operation Schemes for Classes

There are no elements in this category in the system analysed.

## 5.33 Secondary Types - Operation Schemes for Datatype dtSMS

### 5.33.1 Operation Model for *is*

The *is* operation has the following properties:

OPERATION
<i>is</i>
used to determine which strings are considered as valid comments
<i>Return type</i>
ptBoolean
<i>Post-Condition (functional)</i>
PostF 1 true iff the length of the string value is not more than 160 characters.

The listing 5.85 provides the **Messip** (MCL-oriented) specification of the operation.

```

1
2 /* Post Functional:*/
3 postF{let TheResult: ptBoolean in
4   let MaxLength: ptInteger in
5   ( if
6     ( MaxLength = 160
7       and AdtValue.value.length().leq(MaxLength)
8     )
9     then (TheResult = true)
10    else (TheResult = false)
11  endif
12  result = TheResult
13 )}

```

Listing 5.85: **Messip** (MCL-oriented) specification of the operation *is*.

The listing 5.86 provides the **Messip** (Prolog-oriented) implementation of the operation.

<sup>1</sup>%%%%%%%%%%%%%

```

2 /* DISCONTIGUOUS PREDICATES */
3 :- multifile msrop/4.
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %% dtComment
6
7%msd01
8msrop(dtSMS,is,[AdtValue],Result):-
9 msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MaxLength),
11 (
12   (
13     MaxLength = [ptInteger,160],
14     msrNav([AdtValue],
15       [value,length,[],leq,[MaxLength]],
16       [[ptBoolean,true]]))
17   )
18   -> TheResult = [ptBoolean,true]
19   ; TheResult = [ptBoolean,false]
20 )
21 )
22),
23 Result = TheResult
24.

```

Listing 5.86: **Mess1P** (Prolog-oriented) implementation of the operation *is*.

## 5.34 Secondary Types - Operation Schemes for Enumerations

There are no elements in this category in the system analysed.



# Chapter 6

## Test Model(s)

### 6.1 Test Model for testcase01

this positive test case intends to verify the correctness of the execution of a simple instance of the `suDeployAndRun` use case.

#### 6.1.1 Test Steps Specification

##### 6.1.1.1 testcase01-ts01oeCreateSystemAndEnvironment-actMsrCreator.outactMsrCreator.oeCreate

The `testcase01-ts01oeCreateSystemAndEnvironment-actMsrCreator.outactMsrCreator.oeCreate` has the following properties:

TEST STEP	
<i>ts01oeCreateSystemAndEnvironment</i>	
This test step initializes the system state and environment.	
<i>Test Sent Message</i>	
TSM 1	<p><b>out:Creator</b></p> <p>sends to system</p> <p><b>actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment</b> (AqtyComCompanies)</p>
<i>Variables</i>	
V 1	<b>Creator:icrash.environment.actMsrCreator</b> only actMsrCreator actors can trigger the system and environment creation and initialization.
<i>Constraints</i>	
C 1	the number of communication company actor instances present in the environment is equal to four to represent all the communication companies available in Luxembourg.
<i>Oracle Constraints</i>	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.1 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   Creator:actMsrCreator
4   AqtyComCompanies: ptInteger
5 }
6
7 constraints{
8   AqtyComCompanies = 4
9 }
10
11 oracle{
12   constraints{
13   true
14 }
15 }
```

Listing 6.1: **Messir** (MCL-oriented) specification of the test step *testcase01-ts01oeCreateSystemAndEnvironment*.

The listing 6.2 provides the **Messir** (Prolog-oriented) implementation of the test step.

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrTest/1.
4%%%%%%%%%%%%%
5
6%-----
7:-msrTestAddStep([system,sim,1,1]). %-----%
8%-----%
9msrTest([[system,sim,1,1],
10      [[target,oeCreateSystemAndEnvironment],
11       [context,Context],
12       [inputParameters,InputParameters],
13       [outputParameters,OutputParameters],
14       [comments,Comments],
15       TestResult]
16     ]):- %-----%
17%%-----%
18(
19(
20% Step 0
21
22%% Context Declaration
23%% N.A.
24
25%% Input Parameters Declaration
26msrVar(ptInteger,AqtyComCompanies),
27
28%% Output Parameters Declaration
29%% N.A.
30
31%% Context Specification
32%% N.A.
33
34%% Input Parameters Specification
35AqtyComCompanies = [ptInteger,4],
36
37%% Output Parameters Specification
38%% N.A.
39
40%% Test Specification
41Target = launchCreateSystemAndEnvironment,
42ParametersList =
43[ [AqtyComCompanies],
44 Result
45], !,
```

```

46GoalGet=..[Target | ParametersList],
47
48%% Oracle specification
49OracleGet=..[true]
50)
51->
52%% Test Interpretation
53((GoalGet,!))
54-> ((OracleGet,!))
55 -> TestResult = [success]
56 ; TestResult = [failedAtOracle]
57; TestResult = [failedAtGoal]
58)
59; TestResult = [failedAtTestDeclarationOrSpecification]
60),
61%% Test Outcome
62Context = [],
63InputParameters = ['AqtyComCompanies',AqtyComCompanies],
64OutputParameters = [],
65Comments = 'System launch ! '
66.

```

Listing 6.2: **Messip** (Prolog-oriented) implementation of the test step *testcase01-ts01oeCreateSystemAndEnvironment*.

### 6.1.1.2 testcase01-ts02oeSetClock-actActivator.outactActivator.oeSetClock

The *testcase01-ts02oeSetClock-actActivator.outactActivator.oeSetClock* has the following properties:

<b>TEST STEP</b>	
<b><i>ts02oeSetClock</i></b>	
test the update of the current time.	
<b><i>Test Sent Message</i></b>	
TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actActivator.outactActivator.oeSetClock (ACurrentClock)</b>
<b><i>Variables</i></b>	
V 1	<b>TheActor:actActivator</b> proactive actor responsible of requesting the update of the system's clock.
<b><i>Constraints</i></b>	
C 1	TheActor is any instance existing in the current environment status.
C 2	ACurrentClock is a fixed date equal to the 24th November 2017 at 15:20:00 using a 24-hours notation <sup>1</sup> .
<b><i>Oracle Constraints</i></b>	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.3 provides the **Messip** (MCL-oriented) specification of the test step.

<sup>1</sup>for more details see the ISO 8601 Data elements and interchange formats Information interchange Representation of dates and times - <http://www.iso.org/iso/home/standards/iso8601.htm>

```

1
2 variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 24
12  ACurrentClock.time.hour.value = 15
13  ACurrentClock.time.minute.value = 20
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18   constraints{
19     true
20   }
21 }
```

Listing 6.3: **Messip** (MCL-oriented) specification of the test step *testcase01-ts02oeSetClock*.

#### 6.1.1.3 testcase01-ts03oeLogin-actAdministrator.outactAdministrator.oeLogin

The `testcase01-ts03oeLogin-actAdministrator.outactAdministrator.oeLogin` has the following properties:

<b>TEST STEP</b>	
<b>ts03oeLogin</b>	
test the authentified access of the administrator	
<b>Test Sent Message</b>	
TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actAdministrator.outactAdministrator.oeLogin</b> ( <code>AdtLogin</code> , <code>AdtPassword</code> )
<b>Variables</b>	
V 1	<b>TheActor:actAdministrator</b> an actAdministrator actor as subtype of actAuthenticated can send oeLogin messages to the system.
<b>Constraints</b>	
C 1	TheActor is any <code>actAdministrator</code> instance existing in the environment. It is thus expected that there exist at least one.
C 2	AdtLogin has its value attribute equal to the primitive string 'icrashadmin' (which is the correct administrator login known by the system after the step one.)
C 3	AdtPassword has its value attribute equal to the primitive string '7WXC1359' (which is the correct administrator password known by the system after the step one.)
<b>Oracle Constraints</b>	
OC 1	the <code>AMessage</code> value is expected to be equal to the primitive string 'You are logged ! Welcome ...'
OC 2	TheActor receives from system ieMessage( <code>AMessage</code> )

The listing 6.4 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actAdministrator
4   AdtLogin:dtLogin
5   AdtPassword:dtPassword
6 }
7
8 constraints{
9   TheActor=TheSystem.rnactAdministrator->any2(true)
10  AdtLogin.value.eq('icrashadmin')
11  AdtPassword.value.eq('7WXC1359')
12 }
13
14 oracle{
15   variables{
16     AMessage:ptString
17   }
18   constraints{
19     AMessage = 'You are logged ! Welcome ...'
20     TheActor.inactAdministrator.ieMessage(AMessage)
21   }
22 }
```

Listing 6.4: **Messir** (MCL-oriented) specification of the test step *testcase01-ts03oeLogin*.

#### 6.1.1.4 testcase01-ts04oeAddCoordinator-actAdministrator.outactAdministrator.oeAddCoordinator

The `testcase01-ts04oeAddCoordinator-actAdministrator.outactAdministrator.oeAddCoordinator` has the following properties:

<b>TEST STEP</b>	
<b><i>ts04oeAddCoordinator</i></b>	
to test the add of a new coordinator by an administrator.	
<b><i>Test Sent Message</i></b>	
TSM 1	<p><b>out:TheActor</b></p> <p>sends to system</p> <p><b>actAdministrator.outactAdministrator.oeAddCoordinator</b> (<b>AdtCoordinatorID</b>, <b>AdtLogin</b>, <b>AdtPassword</b>)</p>
<b><i>Variables</i></b>	
V 1	<p><b>TheActor:actAdministrator</b></p> <p>actAdministrator actors as being the only one allowed to add coordinators.</p>
<b><i>Constraints</i></b>	
C 1	TheActor is any <code>actAdministrator</code> instance existing in the environment. It is expected that there exists at least one which is the same during all the test case.
C 2	AdtCoordinatorID is equal to 1 to set the new coordinator ID
C 3	AdtLogin has its value attribute equal to the primitive string 'steve' which is the ID defined for the new coordinator.
C 4	AdtPassword has its value attribute equal to the primitive string 'pwdMessirExcalibur2017' which is the password to be set for steve.
<b><i>Oracle Constraints</i></b>	

***continues in next page ...***

**... Test Step table continuation**

OC 1	the administrator should have been acknowledged for the adding of the new coordinator.
------	--

The listing 6.5 provides the **Messir** (MCL-oriented) specification of the test step.

```

1  variables{
2    TheActor : actAdministrator
3    AdtCoordinatorID : dtCoordinatorID
4    AdtLogin:dtLogin
5    AdtPassword:dtPassword
6  }
7
8
9  constraints{
10   TheActor = TheSystem.rnactAdministrator->any2(true)
11   AdtCoordinatorID.value.eq('1')
12   AdtLogin.value.eq('steve')
13   AdtPassword.value.eq('pwdMessirExcalibur2017')
14 }
15
16 oracle{
17   constraints{
18     TheActor.inactAdministrator.ieCoordinatorAdded()
19   }
20 }
```

Listing 6.5: **Messir** (MCL-oriented) specification of the test step *testcase01-ts04oeAddCoordinator*.

#### 6.1.1.5 testcase01-ts05oeLogout-actAdministrator.outactAdministrator.oeLogout

The `testcase01-ts05oeLogout-actAdministrator.outactAdministrator.oeLogout` has the following properties:

<b>TEST STEP</b>	
<i>ts05oeLogout</i>	
to test the logout of a connected administrator.	
<i>Test Sent Message</i>	
TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actAdministrator.outactAdministrator.oeLogout ()</b>
<i>Variables</i>	
V 1	<b>TheActor:actAdministrator</b> an actAdministrator actor as subtype of actAuthenticated can send oeLogout messages to the system.
<i>Constraints</i>	
C 1	TheActor is any <code>actAdministrator</code> instance existing in the environment. It is expected that there exists at least one which is the same during all the test case.
<i>Oracle Constraints</i>	

*continues in next page ...*

**... Test Step table continuation**

OC 1	the AMessage value is expected to be equal to the primitive string 'You are logged out ! Good Bye ...'
OC 2	the administrator should have received the messahe AMessage.

The listing 6.6 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actAdministrator
4 }
5
6 constraints{
7   TheActor = TheSystem.rnactAdministrator->any2(true)
8 }
9
10 oracle{
11   variables{
12     AMessage:ptString
13   }
14 constraints{
15   AMessage = 'You are logged out ! Good Bye ...'
16   TheActor.inactAdministrator.ieMessage(AMessage)
17 }
18 }
```

Listing 6.6: **Messip** (MCL-oriented) specification of the test step *testcase01-ts05oeLogout*.

#### 6.1.1.6 testcase01-ts06oeSetClock02-actActivator.outactActivator.oeSetClock

The `testcase01-ts06oeSetClock02-actActivator.outactActivator.oeSetClock` has the following properties:

<b>TEST STEP</b>	
<b><i>ts06oeSetClock02</i></b>	
test the update of the current time.	
<b><i>Test Sent Message</i></b>	
TSM 1	<b>out:TheActor</b>  <b>sends to system</b>  <b>actActivator.outactActivator.oeSetClock (ACurrentClock)</b>
<b><i>Variables</i></b>	
V 1	<b>TheActor:icrash.environment.actActivator</b> proactive actors responsible of requesting the update of the system's clock.
<b><i>Constraints</i></b>	
C 1	TheActor is any instance existing in the current environment status.
C 2	ACurrentClock is a fixed date equal to the 26th November 2017 at 10:15:00 using a 24-hours notation.
<b><i>Oracle Constraints</i></b>	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.7 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 26
12  ACurrentClock.time.hour.value = 10
13  ACurrentClock.time.minute.value = 15
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18   constraints{
19     true
20   }
21 }
```

Listing 6.7: **Messip** (MCL-oriented) specification of the test step *testcase01-ts06oeSetClock02*.

#### 6.1.1.7 testcase01-ts07oeAlert1-actComCompany.outactComCompany.oeAlert

The `testcase01-ts07oeAlert1-actComCompany.outactComCompany.oeAlert` has the following properties:

<b>TEST STEP</b>	
<b>ts07oeAlert1</b>	
tests the declaration of a new alert functionality.	
<b>Test Sent Message</b>	
TSM 1	<p><b>out:TheActor</b></p> <p>sends to system</p> <p><b>actComCompany.outactComCompany.oeAlert</b> (AetHumanKind, AdtDate, AdtTime, AdtPhoneNumber, AdtGPSLocation, AdtComment)</p>
<b>Variables</b>	
V 1	<p><b>TheActor:actComCompany</b></p> <p>actComCompany actors transfer alert declaration messages.</p>
<b>Constraints</b>	
C 1	TheActor is any instance existing in the current environment status. It is expected to exist at least one.
C 2	AetHumanKind is equal to witness
C 3	AdtDate is equal to the 26th of November 2017
C 4	AdtTime is equal to 10:10:16 using a 24-hours.
C 5	AdtPhoneNumber is equal to the ptString value '+3524666445252'.
C 6	AdtGPSLocation is equal to (49.627675 , 6.159590).
C 7	AdtComment is equal to '3 cars involved in an accident.'

*continues in next page ...*

**... Test Step table continuation**

<i>Oracle Constraints</i>	
OC 1	AdtSMS is equal to the ptString 'Your alert has been registered. We will handle it and keep you informed'.
OC 2	AdtSMS is sent to the phone number AdtPhoneNumber using the communication company having sent the alert using its ieSmsSend input message.

The listing 6.8 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actComCompany
4   AetHumanKind:etHumanKind
5   AdtDate:dtDate
6   AdtTime:dtTime
7   AdtPhoneNumber:dtPhoneNumber
8   AdtGPSLocation:dtGPSLocation
9   AdtComment:dtComment
10 }
11
12 constraints{
13   TheActor = TheSystem.rnactComCompany->any2(true)
14   AetHumanKind = witness
15   AdtDate.year.value = 2017
16   AdtDate.month.value = 11
17   AdtDate.day.value = 26
18   AdtTime.hour.value = 10
19   AdtTime.minute.value = 10
20   AdtTime.second.value = 16
21   AdtPhoneNumber.value = '+3524666445252'
22   AdtGPSLocation.latitude.value = 49.627675
23   AdtGPSLocation.longitude.value = 6.159590
24   AdtComment.value = '3 cars involved in an accident.'
25 }
26
27 oracle{
28   variables{
29     AdtSMS:dtSMS
30   }
31   constraints{
32     AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
33     TheActor.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
34   }
35 }
```

Listing 6.8: **Messip** (MCL-oriented) specification of the test step *testcase01-ts07oeAlert1*.

#### 6.1.1.8 testcase01-ts08oeSetClock03-actActivator.outactActivator.oeSetClock

The *testcase01-ts08oeSetClock03-actActivator.outactActivator.oeSetClock* has the following properties:

TEST STEP
<i>ts08oeSetClock03</i>
test the update of the current time.
<i>Test Sent Message</i>

*continues in next page ...*

**... Test Step table continuation**

TSM 1	<p><b>out:</b>TheActor</p> <p>sends to system</p> <p><b>actActivator.outactActivator.oeSetClock (ACurrentClock)</b></p>
<b>Variables</b>	
V 1	<b>TheActor:actActivator</b> proactive actor responsible of requesting the update of the system's clock.
<b>Constraints</b>	
C 1	TheActor is any instance existing in the current environment status.
C 2	ACurrentClock is a fixed date equal to the 26th November 2017 at 10:30:00 using a 24-hours notation.
<b>Oracle Constraints</b>	
OC 1	true for testing only the executability (is available and can be triggered) of the operation.

The listing 6.9 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 26
12  ACurrentClock.time.hour.value = 10
13  ACurrentClock.time.minute.value = 30
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18   constraints{
19     true
20   }
21 }
```

Listing 6.9: **Messir** (MCL-oriented) specification of the test step *testcase01-ts08oeSetClock03*.

### 6.1.1.9 testcase01-ts09oeSollicitateCrisisHandling-actActivator.outactActivator.oeSollicitateCrisisHandling

The *testcase01-ts09oeSollicitateCrisisHandling-actActivator.outactActivator.oeSollicitateCrisisHandling* has the following properties:

TEST STEP
<b>ts09oeSollicitateCrisisHandling</b>
test the proactive sollication to handle an alert.
<b>Test Sent Message</b>

*continues in next page ...*

*... Test Step table continuation*

TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actActivator.outactActivator.oeSollicitateCrisisHandling ()</b>
<i>Variables</i>	
V 1	<b>TheActor:icrash.environment.actActivator</b> proactive actor responsible of triggering sollicitation functionality.
<i>Constraints</i>	
C 1	TheActor is any instance existing in the current environment status. It is expected to exist at least one.
<i>Oracle Variables</i>	
OV 1	<b>TheAdministrator:actAdministrator</b> actAdministrator actors can be sollicitated to handle alerts.
OV 2	<b>TheCoordinator:actCoordinator</b> actCoordinator actors can be sollicitated to handle alerts.
OV 3	<b>AMessageForCrisisHandlers:ptString</b> messages sent to sollicitated actors are of type ptString.
<i>Oracle Constraints</i>	
OC 1	TheAdministrator is any instance existing in the current environment status. It is expected to exist at least one.
OC 2	TheCoordinator is any instance existing in the current environment status. It is expected to exist at least one.
OC 3	AMessageForCrisisHandlers is equal to the ptString 'There are alerts pending since more than the defined delay. Please REACT !'
OC 4	TheCoordinator and TheAdministrator have received the message AMessageForCrisisHandlers.

The listing 6.10 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actActivator
4 }
5
6 constraints{
7   TheActor = TheSystem.rnactActivator->any2(true)
8 }
9
10 oracle{
11   variables{
12     TheAdministrator:actAdministrator
13     TheCoordinator:actCoordinator
14     AMessageForCrisisHandlers:ptString
15   }
16   constraints{
17     TheAdministrator = TheSystem.rnactAdministrator->any2(true)
18     TheCoordinator = TheSystem.rnactCoordinator->any2(true)
19     AMessageForCrisisHandlers = 'There are alerts pending since more than the defined delay. Please
                                  REACT !'
20   TheAdministrator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)

```

```

21     TheCoordinator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
22   }
23 }
```

Listing 6.10: **Messir** (MCL-oriented) specification of the test step *testcase01-ts09oeSollicitateCrisisHandling*.

#### 6.1.1.10 testcase01-ts10oeLogin02-actAuthenticated.outactAuthenticated.oeLogin

The `testcase01-ts10oeLogin02-actAuthenticated.outactAuthenticated.oeLogin` has the following properties:

<b>TEST STEP</b>	
<i>ts10oeLogin02</i>	
test the authentified access of the coordinator	
<i>Test Sent Message</i>	
TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actAuthenticated.outactAuthenticated.oeLogin (AdtLogin, AdtPassword)</b>
<i>Variables</i>	
V 1	<b>TheActor:actCoordinator</b> an actCoordinator actor as subtype of actAuthenticated can send oeLogin messages to the system.
<i>Constraints</i>	
C 1	TheActor is any <code>actAdministrator</code> instance existing in the environment. It is thus expected that there exist at least one.
C 2	AdtLogin has its value attribute equal to the primitive string 'icrashadmin' (which is the correct administrator login known by the system after the step one.)
C 3	AdtPassword has its value attribute equal to the primitive string '7WXC1359' (which is the correct administrator password known by the system after the step one.)
<i>Oracle Constraints</i>	
OC 1	the <code>AMessage</code> value is expected to be equal to the primitive string 'You are logged ! Welcome ...'

The listing 6.11 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AdtLogin:dtLogin
5   AdtPassword:dtPassword
6 }
7
8 constraints{
9   TheActor = TheSystem.rnactCoordinator->select(a | a.rnctCoordinator.login.value.eq('steve'))->any2
10    (true)
11   AdtLogin.value.eq('steve')
12   AdtPassword.value.eq('pwdMessirExcalibur2017')
```

```

12 }
13
14 oracle{
15   variables{
16     AMesssage:ptString
17   }
18   constraints{
19     AMesssage = 'You are logged ! Welcome ...'
20     TheActor.inactAuthenticated.ieMessage(AMesssage)
21   }
22 }
```

Listing 6.11: **Messip** (MCL-oriented) specification of the test step *testcase01-ts10oeLogin02*.

#### 6.1.1.11 testcase01-ts11oeGetCrisisSet-actCoordinator.outactCoordinator.oeGetCrisisSet

The *testcase01-ts11oeGetCrisisSet-actCoordinator.outactCoordinator.oeGetCrisisSet* has the following properties:

TEST STEP	
<i>ts11oeGetCrisisSet</i> cf. actor documentation	
<i>Test Sent Message</i>	
TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actCoordinator.outactCoordinator.oeGetCrisisSet</b> ( <i>AetCrisisStatus</i> )
<i>Variables</i>	
V 1	<b>TheActor:icrash.environment.actCoordinator</b> cf. actor documentation
V 2	<b>AetCrisisStatus:icrash.concepts.primarytypes.datatypes.etCrisisStatus</b> cf. actor documentation
V 3	<b>ActCrisis:icrash.concepts.primarytypes.classes.ctCrisis</b> cf. actor documentation
<i>Constraints</i>	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	<i>AetCrisisStatus</i> value is pending
<i>Oracle Constraints</i>	
OC 1	ActCrisis is any <i>ctCrisis</i> instance that has been sent to TheActor.

The listing 6.12 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AetCrisisStatus : etCrisisStatus
5 }
6
7 constraints{
```

```

8   TheActor=TheSystem.rnactCoordinator
9       ->select(a | a.bnctCoordinator.login.value.eq('steve'))
10      ->any2(true)
11  AetCrisisStatus = pending
12 }
13
14 oracle{
15   variables{
16     ActCrisis:ctCrisis
17   }
18   constraints{
19     TheActor.bnactCoordinator.ieSendACrisis(ActCrisis)
20   }
21 }
```

Listing 6.12: **Messir** (MCL-oriented) specification of the test step *testcase01-ts11oeGetCrisisSet*.

### 6.1.1.12 testcase01-ts12oeSetCrisisHandler-actCoordinator.outactCoordinator.oeSetCrisisHandler

The *testcase01-ts12oeSetCrisisHandler-actCoordinator.outactCoordinator.oeSetCrisisHandler* has the following properties:

<b>TEST STEP</b>	
<i>ts12oeSetCrisisHandler</i> cf. actor documentation	
<i>Test Sent Message</i>	
TSM 1	<b>out:</b> TheActor sends to system <b>actCoordinator.outactCoordinator.oeSetCrisisHandler</b> ( <b>AdtCrisisID</b> )
<i>Variables</i>	
V 1	<b>TheActor:icrash.environment.actCoordinator</b> cf. actor documentation
V 2	<b>TheComCompany:icrash.environment.actComCompany</b> cf. actor documentation
V 3	<b>TheCoordinator:icrash.environment.actCoordinator</b> cf. actor documentation
V 4	<b>AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID</b> cf. actor documentation
V 5	<b>AMessage:lu.uni.lassy.messir.libraries.primitives.ptString</b> cf. actor documentation
V 6	<b>AdtPhoneNumber:icrash.concepts.primarytypes.datatypes.dtPhoneNumber</b> cf. actor documentation
V 7	<b>AdtSMS:icrash.concepts.secondarytypes.datatypes.dtSMS</b> cf. actor documentation
V 8	<b>ActAlert:icrash.concepts.primarytypes.classes.ctAlert</b> cf. actor documentation
<i>Constraints</i>	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID as a value of 1

*continues in next page ...*

**... Test Step table continuation**

C 3	AMessage is the string 'You are now considered as handling the crisis !'
C 4	AdtPhoneNumber
C 5	AdtSMS has for value the string 'The handling of your alert by our services is in progress !'
<b>Oracle Constraints</b>	
OC 1	there is a communication company actor that received the message ieSmsSend(AdtPhoneNumber,AdtSMS)
OC 2	there is a coordinator actor that received an alert using the message ieSendAnAlert(ActAlert)

The listing 6.13 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AdtCrisisID : dtCrisisID
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactCoordinator
9     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
10    ->any2(true)
11 }
12
13 oracle{
14   variables{
15     AMessage:ptString
16     AdtPhoneNumber:dtPhoneNumber
17     AdtSMS:dtSMS
18     ActAlert:ctAlert
19     TheComCompany: actComCompany
20     TheCoordinator:actCoordinator
21   }
22   constraints{
23     AMessage = 'You are now considered as handling the crisis !'
24     AdtSMS.value = 'The handling of your alert by our services is in progress !'
25     TheComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
26     TheCoordinator.inactCoordinator.ieSendAnAlert(ActAlert)
27     TheActor.inactAuthenticated.ieMessage(AMessage)
28   }
29 }
```

Listing 6.13: **Messip** (MCL-oriented) specification of the test step *testcase01-ts13oeSetCrisisHandler*.

#### 6.1.1.13 testcase01-ts13oeSetClock04-actActivator.outactActivator.oeSetClock

The *testcase01-ts13oeSetClock04-actActivator.outactActivator.oeSetClock* has the following properties:

TEST STEP
<i>ts13oeSetClock04</i>
cf. actor documentation
<i>Test Sent Message</i>

*continues in next page ...*

**... Test Step table continuation**

TSM 1	<p><b>out:</b>TheActor</p> <p>sends to system</p> <p><b>actActivator.outactActivator.oeSetClock (ACurrentClock)</b></p>
<b>Variables</b>	
V 1	<b>TheActor:</b> icrash.environment.actActivator cf. actor documentation
V 2	<b>ACurrentClock:</b> lu.uni.lassy.messir.libraries.calendar.dtDateAndTime cf. actor documentation
<b>Constraints</b>	
C 1	TheActor
C 2	ACurrentClock

The listing 6.14 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 26
12  ACurrentClock.time.hour.value = 10
13  ACurrentClock.time.minute.value = 45
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18   constraints{
19     true
20   }
21 }
```

Listing 6.14: **Messir** (MCL-oriented) specification of the test step *testcase01-ts13oeSetClock04*.

#### 6.1.1.14 testcase01-ts14oeValidateAlert-actCoordinator.outactCoordinator.oeValidateAlert

The `testcase01-ts14oeValidateAlert-actCoordinator.outactCoordinator.oeValidateAlert` has the following properties:

TEST STEP
<b>ts14oeValidateAlert</b>
cf. actor documentation

*Test Sent Message*

*continues in next page ...*

*... Test Step table continuation*

TSM 1	<b>out:</b> TheActor sends to system <b>actCoordinator.outactCoordinator.oeValidateAlert</b> (AdtAlertID)
<i>Variables</i>	
V 1	<b>TheActor:icrash.environment.actCoordinator</b> cf. actor documentation
V 2	<b>AdtAlertID:icrash.concepts.primarytypes.datatypes.dtAlertID</b> cf. actor documentation
V 3	<b>AMessage:lu.uni.lassy.messir.libraries.primitives.ptString</b> cf. actor documentation
<i>Constraints</i>	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtAlertID
C 3	AMessage
<i>Oracle Constraints</i>	
OC 1	

The listing 6.15 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AdtAlertID : dtAlertID
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactCoordinator
9     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
10    ->any2(true)
11 }
12
13 oracle{
14   variables{
15     AMessage:ptString
16   }
17   constraints{
18     AMessage = 'The Alert is now declared as valid !'
19     TheActor.actAuthenticated.inactAuthenticated.ieMessage(AMessage)
20   }
21 }
```

Listing 6.15: **Messir** (MCL-oriented) specification of the test step *testcase01-ts14oeValidateAlert*.

#### 6.1.1.15 testcase01-ts15oeAlert2-actComCompany.outactComCompany.oeAlert

The `testcase01-ts15oeAlert2-actComCompany.outactComCompany.oeAlert` has the following properties:

<b>TEST STEP</b>	
<i>ts15oeAlert2</i> cf. actor documentation	
<i>Test Sent Message</i>	
TSM 1	<p><b>out:TheActor</b></p> <p>sends to system</p> <p><b>actComCompany.outactComCompany.oeAlert</b> (AetHumanKind, AdtDate, AdtTime, AdtPhoneNumber, AdtGPSLocation, AdtComment)</p>
<i>Variables</i>	
V 1	<b>TheActor:icrash.environment.actComCompany</b> cf. actor documentation
V 2	<b>AetHumanKind:icrash.concepts.primarytypes.datatypes.etHumanKind</b> cf. actor documentation
V 3	<b>AdtDate:lu.uni.lassy.messir.libraries.calendar.dtDate</b> cf. actor documentation
V 4	<b>AdtTime:lu.uni.lassy.messir.libraries.calendar.dtTime</b> cf. actor documentation
V 5	<b>AdtPhoneNumber:icrash.concepts.primarytypes.datatypes.dtPhoneNumber</b> cf. actor documentation
V 6	<b>AdtGPSLocation:icrash.concepts.primarytypes.datatypes.dtGPSLocation</b> cf. actor documentation
V 7	<b>AdtComment:icrash.concepts.primarytypes.datatypes.dtComment</b> cf. actor documentation
V 8	<b>AdtSMS:icrash.concepts.secondarytypes.datatypes.dtSMS</b> cf. actor documentation
<i>Constraints</i>	
C 1	TheActor
C 2	AetHumanKind
C 3	AdtDate
C 4	AdtTime
C 5	AdtPhoneNumber
C 6	AdtGPSLocation
C 7	AdtComment
C 8	AdtSMS
<i>Oracle Constraints</i>	
OC 1	

The listing 6.16 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actComCompany
4   AetHumanKind:etHumanKind
5   AdtDate:dtDate
6   AdtTime:dtTime

```

```

7  AdtPhoneNumber:dtPhoneNumber
8  AdtGPSLocation:dtGPSLocation
9  AdtComment:dtComment
10 }
11
12 constraints{
13   TheActor = TheSystem.rnactComCompany->any2(true)
14   AetHumanKind = witness
15   AdtDate.year.value = 2017
16   AdtDate.month.value = 11
17   AdtDate.day.value = 26
18   AdtTime.hour.value = 10
19   AdtTime.minute.value = 20
20   AdtTime.second.value = 00
21   AdtPhoneNumber.value = '+3524666445000'
22   AdtGPSLocation.latitude.value = 49.627095
23   AdtGPSLocation.longitude.value = 6.160251
24   AdtComment.value = 'A car crash just happened.'
25 }
26
27 oracle{
28   variables{
29     AdtSMS:dtSMS
30   }
31   constraints{
32     AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
33     TheActor.actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
34   }
35 }
```

Listing 6.16: **Messir** (MCL-oriented) specification of the test step *testcase01-ts15oeAlert2*.

#### 6.1.1.16 testcase01-ts16oeSetClock05-actActivator.outactActivator.oeSetClock

The `testcase01-ts16oeSetClock05-actActivator.outactActivator.oeSetClock` has the following properties:

<b>TEST STEP</b>	
<b>ts16oeSetClock05</b>	
cf. actor documentation	
<b>Test Sent Message</b>	
TSM 1	<b>out:TheActor</b>  <b>sends to system</b>  <b>actActivator.outactActivator.oeSetClock (ACurrentClock)</b>
<b>Variables</b>	
V 1	<b>TheActor:icrash.environment.actActivator</b> cf. actor documentation
V 2	<b>ACurrentClock:lu.uni.lassy.messir.libraries.calendar.dtDateAndTime</b> cf. actor documentation
<b>Constraints</b>	
C 1	TheActor
C 2	ACurrentClock

The listing 6.17 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor:actActivator
4   ACurrentClock:dtDateAndTime
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactActivator->any2(true)
9   ACurrentClock.date.year.value = 2017
10  ACurrentClock.date.month.value = 11
11  ACurrentClock.date.day.value = 26
12  ACurrentClock.time.hour.value = 12
13  ACurrentClock.time.minute.value = 45
14  ACurrentClock.time.second.value = 00
15 }
16
17 oracle{
18   constraints{
19     true
20   }
21 }
```

Listing 6.17: **Messir** (MCL-oriented) specification of the test step *testcase01-ts16oeSetClock05*.

#### 6.1.1.17 testcase01-ts17oeSetCrisisStatus-actCoordinator.outactCoordinator.oeSetCrisisStatus

The *testcase01-ts17oeSetCrisisStatus-actCoordinator.outactCoordinator.oeSetCrisisStatus* has the following properties:

<b>TEST STEP</b>	
<i>ts17oeSetCrisisStatus</i>	
cf. actor documentation	
<b>Test Sent Message</b>	
TSM 1	<p><b>out:TheActor</b></p> <p>sends to system</p> <p><b>actCoordinator.outactCoordinator.oeSetCrisisStatus</b> (AdtCrisisID, AetCrisisStatus)</p>
<b>Variables</b>	
V 1	<b>TheActor:icrash.environment.actCoordinator</b> cf. actor documentation
V 2	<b>AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID</b> cf. actor documentation
V 3	<b>AetCrisisStatus:icrash.concepts.primarytypes.datatypes.etCrisisStatus</b> cf. actor documentation
V 4	<b>AMessage:lu.uni.lassy.messir.libraries.primitives.ptString</b> cf. actor documentation
<b>Constraints</b>	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID

*continues in next page ...*

**... Test Step table continuation**

C 3	AetCrisisStatus
C 4	AMessage
<b>Oracle Constraints</b>	
OC 1	

The listing 6.18 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AdtCrisisID : dtCrisisID
5   AetCrisisStatus : etCrisisStatus
6 }
7
8 constraints{
9   TheActor=TheSystem.rnactCoordinator
10   ->select(a | a.rnctCoordinator.login.value.eq('steve'))
11   ->any2(true)
12 }
13
14 oracle{
15   variables{
16     AMessage:ptString
17   }
18   constraints{
19     AMessage = 'The crisis status has been updated !'
20     TheActor.inactAuthenticated.ieMessage(AMessage)
21   }
22 }
```

Listing 6.18: **Messir** (MCL-oriented) specification of the test step *testcase01-ts17oeSetCrisisStatus*.

#### 6.1.1.18 testcase01-ts18oeReportOnCrisis-actCoordinator.outactCoordinator.oeReportOnCrisis

The *testcase01-ts18oeReportOnCrisis-actCoordinator.outactCoordinator.oeReportOnCrisis* has the following properties:

<b>TEST STEP</b>	
<i>ts18oeReportOnCrisis</i> cf. actor documentation	
<b>Test Sent Message</b>	
TSM 1	<b>out:TheActor</b> <b>sends to system</b> <b>actCoordinator.outactCoordinator.oeReportOnCrisis</b> (AdtCrisisID, AdtComment)
<b>Variables</b>	
V 1	TheActor:icrash.environment.actCoordinator cf. actor documentation
V 2	AdtCrisisID:icrash.concepts.primarytypes.datatypes.dtCrisisID

***continues in next page ...***

**... Test Step table continuation**

V 3	cf. actor documentation <b>AdtComment:icrash.concepts.primarytypes.datatypes.dtComment</b>
V 4	cf. actor documentation <b>AMessage:lu.uni.lassy.messir.libraries.primitives.ptString</b>
<b>Constraints</b>	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID
C 3	AdtComment
C 4	AMessage
<b>Oracle Constraints</b>	
OC 1	

The listing 6.19 provides the **Messir** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AdtCrisisID : dtCrisisID
5   AdtComment : dtComment
6 }
7
8 constraints{
9   TheActor=TheSystem.rnactCoordinator
10    ->select(a | a.rnctCoordinator.login.value.eq('steve'))
11    ->any2(true)
12 }
13
14 oracle{
15   variables{
16     AMessage:ptString
17   }
18   constraints{
19     AMessage = 'The crisis comment has been updated !'
20     TheActor.inactAuthenticated.ieMessage(AMessage)
21   }
22 }
```

Listing 6.19: **Messir** (MCL-oriented) specification of the test step *testcase01-ts18oeReportOnCrisis*.

#### 6.1.1.19 testcase01-ts19oeCloseCrisis-actCoordinator.outactCoordinator.oeCloseCrisis

The `testcase01-ts19oeCloseCrisis-actCoordinator.outactCoordinator.oeCloseCrisis` has the following properties:

TEST STEP
<i>ts19oeCloseCrisis</i> cf. actor documentation
<i>Test Sent Message</i>

*continues in next page ...*

*... Test Step table continuation*

TSM 1	<b>out:</b> TheActor sends to system <b>actCoordinator.outactCoordinator.oeCloseCrisis</b> (AdtCrisisID)
<i>Variables</i>	
V 1	<b>TheActor:</b> icrash.environment.actCoordinator cf. actor documentation
V 2	<b>AdtCrisisID:</b> icrash.concepts.primarytypes.datatypes.dtCrisisID cf. actor documentation
V 3	<b>AMessage:</b> lu.uni.lassy.messir.libraries.primitives.ptString cf. actor documentation
<i>Constraints</i>	
C 1	TheActor is the coordinator actor related to a coordinator in the system's state having steve as login value
C 2	AdtCrisisID
C 3	AMessage
<i>Oracle Constraints</i>	
OC 1	

The listing 6.20 provides the **Messip** (MCL-oriented) specification of the test step.

```

1
2 variables{
3   TheActor : actCoordinator
4   AdtCrisisID : dtCrisisID
5 }
6
7 constraints{
8   TheActor=TheSystem.rnactCoordinator
9     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
10    ->any2(true)
11 }
12
13 oracle{
14   variables{
15     AMessage:ptString
16   }
17   constraints{
18     AMessage = 'The crisis is now closed !'
19     TheActor.inactAuthenticated.ieMessage(AMessage)
20   }
21 }
```

Listing 6.20: **Messip** (MCL-oriented) specification of the test step *testcase01-ts19oeCloseCrisis*.

### 6.1.2 Test Case Instance - instance01

this positive test case intends to verify the correctness of the execution of a simple instance of the suDeployAndRun use case.

### 6.1.3 Test Case Instance - instance01Part01

The first part of a simple and complete testcase instance for *iCrash*.

Figure 6.1 Sequence diagram representing the first part of a simple and complete testcase instance for *iCrash*.

### 6.1.4 Test Case Instance - instance01Part02

The second part of a simple and complete testcase instance for *iCrash*.

Figure 6.2 Sequence diagram representing the second part of a simple and complete testcase instance for *iCrash*.

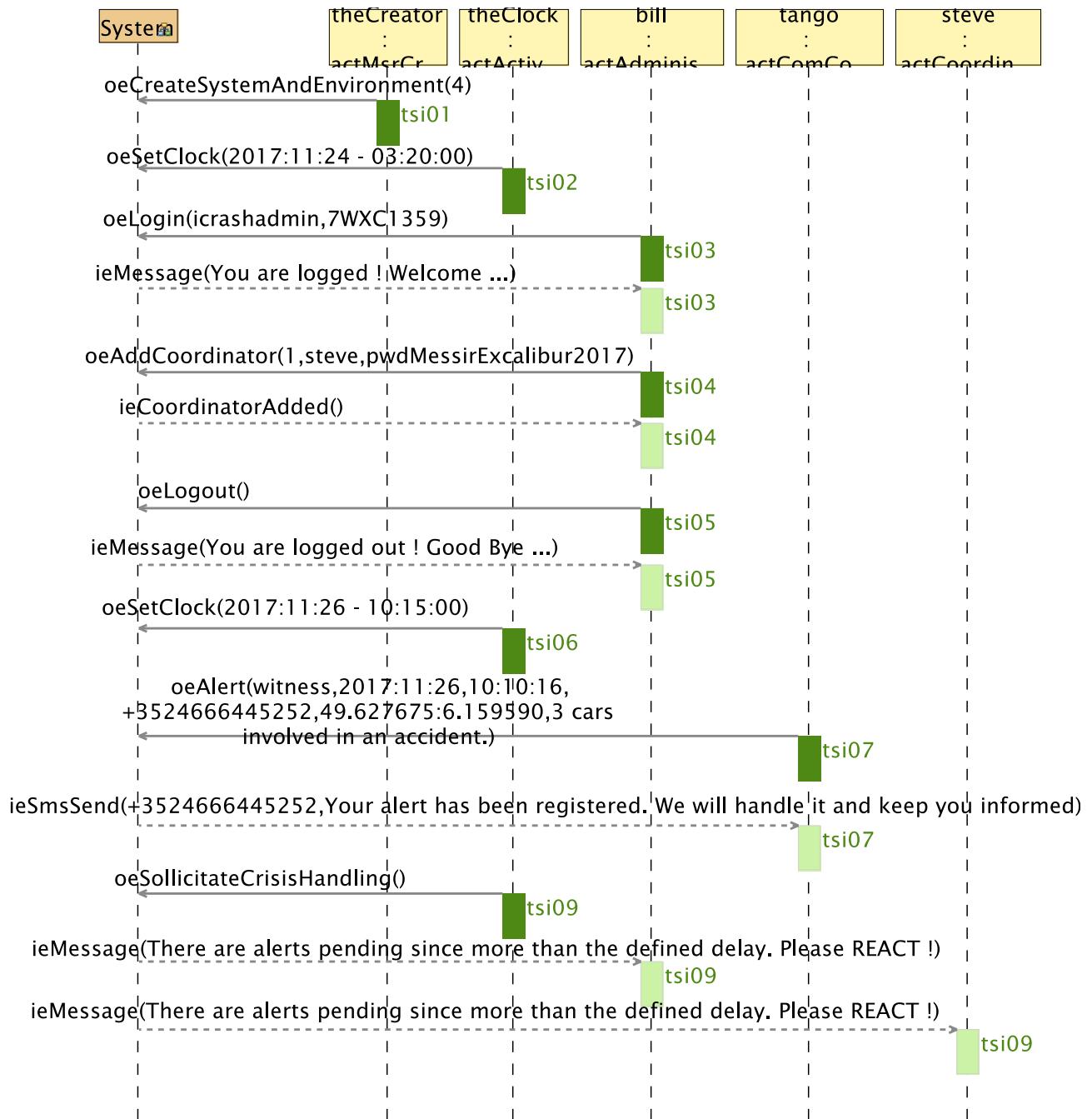


Figure 6.1: tci-testcase01-instance01-Part01 testcase instance sequence diagram

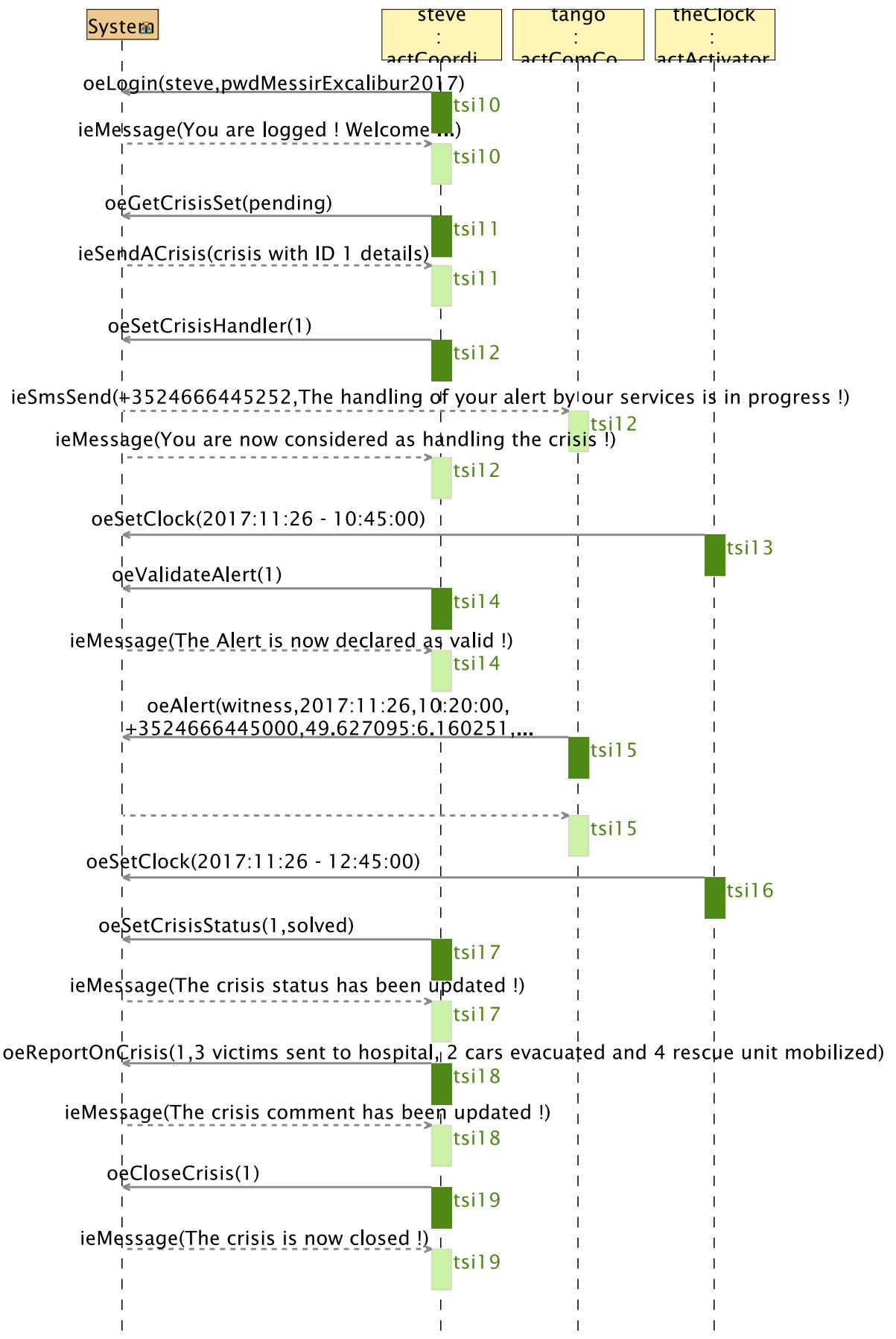


Figure 6.2: tci-testcase01-instance01-Part02 testcase instance sequence diagram

# Chapter 7

## Additional Constraints

### 7.1 Quality Constraints

Description of all the constraints that concern the required quality criteria according to their ISO definition [3].

#### 7.1.1 Functional suitability

Constraints on the degree to which the product provides functions that meet stated and implied needs when the product is used under specified conditions.

##### 7.1.1.1 Functional completeness

List of requirements on the degree to which the set of functions covers all the specified tasks and user objectives.

1. (to be filled)

##### 7.1.1.2 Functional correctness

List of requirements on the degree to which the set of functions covers all the specified tasks and user objectives.

1. (to be filled)

##### 7.1.1.3 Functional appropriateness

List of requirements on the degree to which the functions facilitate the accomplishment of specified tasks and objectives.

1. (to be filled)

#### 7.1.2 Performance efficiency

Constraints on the performance relative to the amount of resources used under stated conditions

##### 7.1.2.1 Time behaviour

List of requirements on the degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

1. (to be filled)

### 7.1.2.2 Resource utilization

List of requirements on the degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

1. (to be filled)

### 7.1.2.3 Capacity

List of requirements on the degree to which the maximum limits of a product or system parameter meet requirements.

1. (to be filled)

### 7.1.3 Compatibility

Constraints on the degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment.

#### 7.1.3.1 Co-existence

List of requirements on the degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

1. (to be filled)

#### 7.1.3.2 Interoperability

List of requirements on the degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

1. (to be filled)

### 7.1.4 Usability

Constraints on the usability degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

#### 7.1.4.1 Appropriateness recognizability

List of requirements on the degree to which users can recognize whether a product or system is appropriate for their needs.

1. (to be filled)

#### 7.1.4.2 Learnability

List of requirements on the degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use.

1. (to be filled)

#### 7.1.4.3 Operability

List of requirements on the degree to which a product or system has attributes that make it easy to operate and control.

1. (to be filled)

#### 7.1.4.4 User error protection

List of requirements on the degree to which a system protects users against making errors.

1. (to be filled)

#### 7.1.4.5 User interface aesthetics

List of requirements on the degree to which a user interface enables pleasing and satisfying interaction for the user.

1. (to be filled)

#### 7.1.4.6 Accessibility

List of requirements on the degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

1. (to be filled)

### 7.1.5 Reliability

Constraints on the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

#### 7.1.5.1 Maturity

List of requirements on the degree to which a system, product or component meets needs for reliability under normal operation.

1. (to be filled)

#### 7.1.5.2 Availability

List of requirements on the degree to which a system, product or component is operational and accessible when required for use.

1. (to be filled)

#### 7.1.5.3 Fault tolerance

List of requirements on the degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

1. (to be filled)

#### **7.1.5.4 Recoverability**

List of requirements on the degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

1. (to be filled)

#### **7.1.6 Security**

Constraints on the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

##### **7.1.6.1 Confidentiality**

List of requirements on the degree to which a product or system ensures that data are accessible only to those authorized to have access.

1. (to be filled)

##### **7.1.6.2 Integrity**

List of requirements on the degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

1. (to be filled)

##### **7.1.6.3 Non-repudiation**

List of requirements on the degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.

1. (to be filled)

##### **7.1.6.4 Accountability**

List of requirements on the degree to which the actions of an entity can be traced uniquely to the entity.

1. (to be filled)

##### **7.1.6.5 Authenticity**

List of requirements on the degree to which the identity of a subject or resource can be proved to be the one claimed.

1. (to be filled)

#### **7.1.7 Maintainability**

Constraints on the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers.

### 7.1.7.1 Modularity

List of requirements on the degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

1. (to be filled)

### 7.1.7.2 Reusability

List of requirements on the degree to which an asset can be used in more than one system, or in building other assets.

1. (to be filled)

### 7.1.7.3 Analysability

List of requirements on the degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

1. (to be filled)

### 7.1.7.4 Modifiability

List of requirements on the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

1. (to be filled)

### 7.1.7.5 Testability

List of requirements on the degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

1. (to be filled)

## 7.1.8 Portability

Constraints on the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

### 7.1.8.1 Adaptability

List of requirements on the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

1. (to be filled)

### 7.1.8.2 Installability

List of requirements on the degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.

1. (to be filled)

### 7.1.8.3 Replaceability

List of requirements on the degree to which a product can replace another specified software product for the same purpose in the same environment.

1. (to be filled)

## 7.2 Other Constraints

Any other unclassified constraints judged as required for the product under development.

## Appendix A

Specification project  
`lu.uni.lassy.excalibur.examples.icrash`

## A.1 Use Cases Model

This section contains the use cases elicited during the requirements elicitation phase. The use cases are textually described as suggested by the **Messip** method and inspired by the standard Cokburn template [2].

### A.1.1 Use Cases

#### A.1.1.1 subfunction-oeCloseCrisis

the `actCoordinator`'s and `actPolice`'s goal is to declare a crisis as closed.

USE-CASE DESCRIPTION	
<i>Name</i>	oeCloseCrisis
<i>Scope</i>	system
<i>Level</i>	subfunction
<i>Primary actor(s)</i>	
1	<code>actCoordinator[active]</code>
<i>Goal(s) description</i>	
the <code>actCoordinator</code> 's and <code>actPolice</code> 's goal is to declare a crisis as closed.	
<i>Protocol condition(s)</i>	
1	the iCrash system has been deployed.
<i>Pre-condition(s)</i>	
1	none
<i>Main post-condition(s)</i>	
1	the crisis is known by the system to be closed.
2	a message <code>ieMessage(AMessage)</code> is sent to the <code>actCoordinator</code> to inform him that his crisis is now considered as closed.
3	a message <code>ieMessage(AMessage)</code> is sent to the <code>actPolice</code> to inform him that his crisis is now considered as closed.

Figure A.1 shows the use case diagram for the oeCloseCrisis subfunction use case

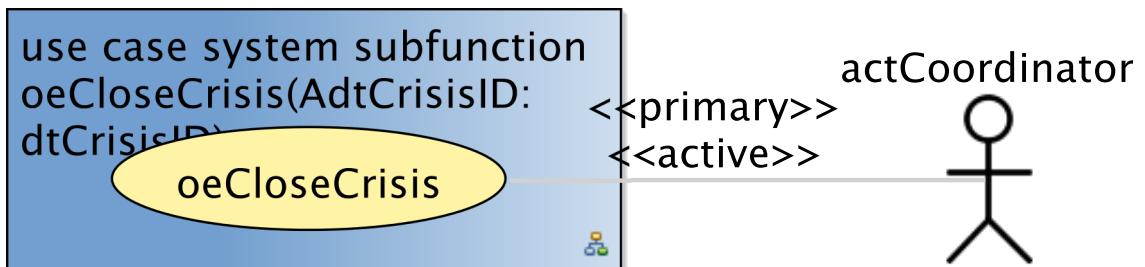


Figure A.1: oeCloseCrisis subfunction use case

## Appendix B

# Messir Specification Files Listing

### B.1 File ./src-gen/messir-spec/.views.msr

```
1 //
2 //DON'T TOUCH THIS FILE !!!
3 //
4 package uuid7e0d382938204f3c9036c123484468fb {
5 Concept Model {}
6 }
```

Listing B.1: Messir Spec. file .views.msr.

### B.2 File ./src-gen/messir-spec/operations/concepts/secondarytypes-datatatypes/dtSMS.msr

```
1 package icrash.operations.concepts.secondarytypes.datatypes.dtSMS{
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.calendar
5 import lu.uni.lassy.messir.libraries.math
6
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.concepts.primarytypes.classes
9 import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11
12 Operation Model {
13 operation: icrash.concepts.secondarytypes.datatypes.dtSMS.is():ptBoolean{
14   postF{
15     let TheResult: ptBoolean in
16     let MaxLength: ptInteger in
17     ( if
18       ( MaxLength = 160
19         and AdtValue.value.length().leq(MaxLength)
20       )
21       then (TheResult = true)
22       else (TheResult = false)
23     endif
24     result = TheResult
25   }
26 prolog{ "src/Operations/Concepts/SecondaryTypesDatatypes/SecondaryTypesDatatypes-dtSMS-is.pl"
27 }
28 }
29 }
```

Listing B.2: Messir Spec. file dtSMS.msr.

### B.3 File ./src-gen/messir-spec/operations/environment/environment-actActivator-oeSetClock.msr

```

1 package icrash.operations.environment.actActivator.oeSetClock {
2
3 import icrash.environment
4
5 import lu.uni.lassy.messir.libraries.primitives
6 import lu.uni.lassy.messir.libraries.calendar
7 import lu.uni.lassy.messir.libraries.math
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11
12 Operation Model {
13
14 operation: actActivator.outactActivator.oeSetClock(AcurrentClock:dtDateAndTime):ptBoolean
15 {
16 prep{
17 let TheSystem: ctState in
18 let AvpStarted: ptBoolean in
19
20 /* PreP01 */
21 self.rnActor.rnSystem = TheSystem
22 and self.rnActor.rnSystem.vpStarted = AvpStarted
23 and AvpStarted = true
24 and TheSystem.clock.lt(AcurrentClock)
25 }
26 pref{true}
27
28 postF{
29 let TheSystem: ctState in
30 self.rnActor.rnSystem = TheSystem
31
32 /* PostF01 */
33 and TheSystem@post.clock = AcurrentClock
34 }
35 postP{true}
36
37 prolog{"src/Operations/Environment/OUT/outactActivator-oeSetClock.pl"}
38
39 }
40 }
41 }
```

Listing B.3: Messir Spec. file environment-actActivator-oeSetClock.msr.

### B.4 File ./src-gen/messir-spec/operations/environment/environment-actActivator-oeSollicitateCrisisHandling.msr

```

1 package icrash.operations.environment.actActivator.oeSollicitateCrisisHandling {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.datatypes
9 import icrash.concepts.primarytypes.classes
10 import icrash.environment
11
12 Operation Model {
13
14 operation: actActivator.outactActivator.oeSollicitateCrisisHandling():ptBoolean
15 {
16 prep{
17 let TheSystem: ctState in
```

## B.5. FILE /SRC-GEN/MESSIR-SPEC.../ENVIRONMENT-ACTADMINISTRATOR-OEADDCOORDINATOR.MSR

```

18 let AvpStarted: ptBoolean in
19 let ColctCrisisToHandle:
20   Bag(ctCrisis) in
21
22 self.rnActor.rnSystem = TheSystem
23
24 /* PreP01 */
25 and TheSystem.vpStarted
26
27 /* PreP02 */
28 and TheSystem.rnctCrisis->select(handlingDelayPassed( ))
29   = ColctCrisisToHandle
30 and ColctCrisisToHandle->size().geq(1)
31 }
32 preF{true}
33
34 postF{
35 let TheSystem: ctState in
36 let AMessageForCrisisHandlers: dtComment in
37 let ColctCrisisToAllocateIfPossible:Bag(ctCrisis) in
38
39 self.rnActor.rnSystem = TheSystem
40 /* PostF01 */
41 and TheSystem.rnctCrisis->select(maxHandlingDelayPassed( ))
42   = ColctCrisisToAllocateIfPossible
43 and ColctCrisisToAllocateIfPossible->forAll(isAllocatedIfPossible())
44
45 /* PostF02 */
46 and TheSystem.rnctCrisis->select(handlingDelayPassed( ))
47 = ColctCrisisToHandle
48
49 and ColctCrisisToHandle->msrColSubtract(ColctCrisisToAllocateIfPossible)
50   = ColctCrisisToRemind
51
52 and if (ColctCrisisToRemind->size().geq(1))
53   then (AMessageForCrisisHandlers.value
54     ='There are alerts pending since more than the defined delay. Please REACT !'
55     and TheSystem.rnactAdministrator.
56       rnInterfaceIN^ieMessage(AMessageForCrisisHandlers)
57       and TheSystem.rnactCoordinator
58         ->forAll(rnInterfaceIN^ieMessage(AMessageForCrisisHandlers))
59   )
60 else true
61 endif
62 }
63 postP{
64 let TheSystem: ctState in
65 let TheClock: dtDateAndTime in
66
67 self.rnActor.rnSystem = TheSystem
68 and TheSystem.clock = TheClock
69 and TheSystem@post.vpLastReminder = TheClock
70 }
71
72 prolog{"src/Operations/Environment/OUT/outactActivator-oeSollicitateCrisisHandling.pl"}
73 }
74 }
75 }

```

Listing B.4: Messir Spec. file environment-actActivator-oeSollicitateCrisisHandling.msr.

## B.5 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeAddCoordinator.msr

```

1 package icrash.operations.environment.actAdministrator.oeAddCoordinator {
2
3 import lu.uni.lassy.messir.libraries.primitives
4

```

```

5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10
11 operation: actAdministrator.outactAdministrator.oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID,
12   AdtLogin:dtLogin, AdtPassword:dtPassword):ptBoolean
13 {
14   let TheSystem: ctState in
15   let TheActor:actAdministrator in
16
17   self.rnActor.rnSystem = TheSystem
18   and self.rnActor = TheActor
19
20 /* PreP01 */
21 and TheSystem.vpStarted = true
22 /* PreP02 */
23 and TheActor.rnctAuthenticated.vpIsLogged = true
24 }
25 preF{
26   let TheSystem: ctState in
27   let TheActor:actAdministrator in
28   let ColctCoordinators:Bag(ctCoordinator) in
29
30   self.rnActor.rnSystem = TheSystem
31   and self.rnActor = TheActor
32 /* PreF01 */
33 and TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
34   = ColctCoordinators
35 and ColctCoordinators->isEmpty() = true
36 }
37 postF{
38   let TheSystem: ctState in
39   let TheactCoordinator:actCoordinator in
40   let ThectCoordinator:ctCoordinator in
41   self.rnActor.rnSystem = TheSystem
42   and self.rnActor = TheActor
43 /* PostF01 */
44 TheactCoordinator.init()
45 /* PostF02 */
46 and ThectCoordinator.init(AdtCoordinatorID,AdtLogin,AdtPassword)
47
48 /* PostF03 */
49 and TheactCoordinator@post.rnctCoordinator = ThectCoordinator
50
51 /* PostF04 */
52 and ThectCoordinator@post.rnactAuthenticated = TheactCoordinator
53
54 /* PostF05 */
55 and TheActor.rnInterfaceIN^ieCoordinatorAdded()
56 }
57 postP{true}
58
59 prolog{"src/Operations/Environment/OUT/outactAdministrator-oeAddCoordinator.pl"}
60 }
61 }
62 }

```

Listing B.5: Messir Spec. file environment-actAdministrator-oeAddCoordinator.msr.

## B.6 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeAddPolice.msr

```

1 package icrash.operations.environment.actAdministrator.oeAddPolice {
2
3 import lu.uni.lassy.messir.libraries.primitives

```

## B.7 FILE /SRC-GEN.../ENVIRONMENT-ACTADMINISTRATOR-OEDELETECOORDINATOR.MSR191

```
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.environment
8
9 Operation Model {
10
11 operation: actAdministrator.outactAdministrator.oeAddPolice(AdtPoliceID:dtPoliceID, AdtLogin:dtLogin
12 , AdtPassword:dtPassword):ptBoolean
13 {
14 let TheSystem: ctState in
15 let TheActor:actAdministrator in
16
17 self.rnActor.rnSystem = TheSystem
18 and self.rnActor = TheActor
19
20 /* PreP01 */
21 and TheSystem.vpStarted = true
22 /* PreP02 */
23 and TheActor.rnctAuthenticated.vpIsLogged = true
24 }
25 preP{
26 let TheSystem: ctState in
27 let TheActor:actAdministrator in
28 let ColctPolices:Bag(ctPolice) in
29
30 self.rnActor.rnSystem = TheSystem
31 and self.rnActor = TheActor
32 /* PostF01 */
33 and TheSystem.rnctPolice->select(id.eq(AdtPoliceID))
34 = ColctPolices
35 and ColctPolices->isEmpty() = true
36 }
37 postF{
38 let TheSystem: ctState in
39 let TheactPolice:actPolice in
40 let ThectPolice:ctPolice in
41 self.rnActor.rnSystem = TheSystem
42 and self.rnActor = TheActor
43 /* PostF01 */
44 TheactPolice.init()
45 /* PostF02 */
46 and ThectPolice.init(AdtPoliceID,AdtLogin,AdtPassword)
47
48 /* PostF03 */
49 and TheactPolice@post.rnctPolice = ThectPolice
50
51 /* PostF04 */
52 and ThectPolice@post.rnactAuthenticated = TheactPolice
53
54 /* PostF05 */
55 and TheActor.rnInterfaceIN^iePoliceAdded()
56 }
57 postP{true}
58
59 prolog{"src/Operations/Environment/OUT/outactAdministrator-oeAddPolice.pl"}
60 }
61 }
62 }
```

Listing B.6: Messir Spec. file environment-actAdministrator-oeAddPolice.msr.

## B.7 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeDeleteCoordinator.msr

```
1 package icrash.operations.environment.actAdministrator.oeDeleteCoordinator {
2
```

```

3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.environment
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11
12 Operation Model {
13
14 operation: actAdministrator.outactAdministrator.oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID
15 ):ptBoolean
15 {
16 prep{
17 let TheSystem: ctState in
18 let TheActor:actAdministrator in
19
20 self.rnActor.rnSystem = TheSystem
21 and self.rnActor = TheActor
22
23 /* PreP01 */
24 and TheSystem.vpStarted = true
25 /* PreP02 */
26 and TheActor.rnctAuthenticated.vpIsLogged = true
27 }
28 pref{
29 let TheSystem: ctState in
30 let TheActor:actAdministrator in
31
32 self.rnActor.rnSystem = TheSystem
33 and self.rnActor = TheActor
34 /* PreF01 */
35 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
36 = ColctCoordinators
37 and ColctCoordinators->size().eq(1)
38 }
39 postF{
40 let TheSystem: ctState in
41 let TheActor:actAdministrator in
42 let ThectCoordinator:ctCoordinator in
43 self.rnActor.rnSystem = TheSystem
44 and self.rnActor = TheActor
45 /* PostF01 */
46 TheSystem.rnctCoordinator->select(id.eq(AdtCoordinatorID))
47 = ThectCoordinator
48 and ThectCoordinator.rnactCoordinator->forAll(msrIsKilled)
49 and ThectCoordinator.msrIsKilled
50
51 /* PostF02 */
52 and TheActor.rnInterfaceIN^ieCoordinatorDeleted()
53
54 /* Post Protocol:*/
55 /* PostP01 */
56 and true
57 }
58 postP{true}
59
60 prolog{"src/Operations/Environment/OUT/outactAdministrator-oeDeleteCoordinator.pl"}
61 }
62 }
63 }

```

Listing B.7: Messir Spec. file environment-actAdministrator-oeDeleteCoordinator.msr.

## B.8 File ./src-gen/messir-spec/operations/environment/environment-actAdministrator-oeDeletePolice.msr

```

1 package icrash.operations.environment.actAdministrator.oeDeletePolice {

```

## B.9. FILE /SRC-GEN/MESSIR-SPEC/OPERATIONS.../ENVIRONMENT-ACTAUTHENTICATED.MSR193

```
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.environment
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11
12 Operation Model {
13
14 operation: actAdministrator.outactAdministrator.oeDeletePolice(AdtPoliceID:dtPoliceID):ptBoolean
15 {
16 prep{
17 let TheSystem: ctState in
18 let TheActor:actAdministrator in
19
20 self.rnActor.rnSystem = TheSystem
21 and self.rnActor = TheActor
22
23 /* PreP01 */
24 and TheSystem.vpStarted = true
25 /* PreP02 */
26 and TheActor.rnctAuthenticated.vpIsLogged = true
27 }
28 pref{
29 let TheSystem: ctState in
30 let TheActor:actAdministrator in
31
32 self.rnActor.rnSystem = TheSystem
33 and self.rnActor = TheActor
34 /* PreF01 */
35 TheSystem.rnctPolice->select(id.eq(AdtPoliceID))
36 = ColctPolices
37 and ColctPolices->size().eq(1)
38 }
39 postF{
40 let TheSystem: ctState in
41 let TheActor:actAdministrator in
42 let ThectPolice:ctPolice in
43 self.rnActor.rnSystem = TheSystem
44 and self.rnActor = TheActor
45 /* PostF01 */
46 TheSystem.rnctPolice->select(id.eq(AdtPoliceID))
47 = ThectPolice
48 and ThectPolice.rnactPolice->forAll(msrIsKilled)
49 and ThectPolice.msrIsKilled
50
51 /* PostF02 */
52 and TheActor.rnInterfaceIN^iePoliceDeleted()
53
54 /* Post Protocol:*/
55 /* PostP01 */
56 and true
57 }
58 postP{true}
59
60 prolog{"src/Operations/Environment/OUT/outactAdministrator-oeDeletePolice.pl"}
61 }
62 }
63 }
```

Listing B.8: Messir Spec. file environment-actAdministrator-oeDeletePolice.msr.

## B.9 File ./src-gen/messir-spec/operations/environment/environment-actAuthenticated.msr

```
1 package icrash.operations.environment.actAuthenticated{
```

```

2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.concepts.secondarytypes.datatypes
8 import icrash.concepts.secondarytypes.classes
9 import icrash.environment
10
11 Operation Model {
12
13 operation: actAuthenticated.outactAuthenticated.oeLogin(AdtLogin:dtLogin, AdtPassword:dtPassword):
14     ptBoolean
15 {
16     let TheSystem: ctState in
17     let TheActor:actAuthenticated in
18     self.rnActor.rnSystem = TheSystem
19     and self.rnActor = TheActor
20
21 /* PreP01 */
22 and TheSystem.vpStarted = true
23 /* PreP02 */
24 and TheActor.rnctAuthenticated.vpIsLogged = false
25 }
26 preF{
27 /* PreF01 */
28 true
29 }
30 postF{
31 let TheSystem: ctState in
32 let TheactAuthenticated:actAuthenticated in
33
34 let AptStringMessageForTheactAuthenticated: ptString in
35 let AptStringMessageForTheactAdministrator:ptString in
36
37 self.rnActor.rnSystem = TheSystem
38 and self.rnActor = TheactAuthenticated
39
40 and /* PostF01 */
41 if (TheactAuthenticated.rnctAuthenticated.pwd
42     = AdtPassword
43     and TheactAuthenticated.rnctAuthenticated.login
44     = AdtLogin
45     )
46 then (AptStringMessageForTheactAuthenticated.eq('You are logged ! Welcome ...')
47         and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
48         )
49 else (AptStringMessageForTheactAuthenticated
50     .eq('Wrong identification information ! Please try again ...')
51     and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
52     and AptStringMessageForTheactAdministrator.eq('Intrusion tentative !')
53     and TheSystem.rnactAdministrator
54     .rnInterfaceIN^ieMessage(AptStringMessageForTheactAdministrator)
55     )
56 endif
57 }
58 postP{
59 let TheSystem: ctState in
60 let TheactAuthenticated:actAuthenticated in
61
62 self.rnActor.rnSystem = TheSystem
63 and self.rnActor = TheactAuthenticated
64 /* PostP01 */
65 if (TheactAuthenticated.rnctAuthenticated.pwd = AdtPassword
66     and TheactAuthenticated.rnctAuthenticated.login = AdtLogin
67     )
68 then (TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = true)
69 else true
70 endif

```

## B.10. FILE /SRC-GEN/MESSIR-SPEC/OPERATIONS/ENVIRONMENT/ENVIRONMENT-ACTCOMCOMPANY

```
71 }
72 prolog{"src/Operations/Environment/OUT/outactAuthenticated-oeLogin.pl"}
73 }
74 /*-----*/
75
76 operation: actAuthenticated.outactAuthenticated.oeLogout():ptBoolean{
77
78 preP{
79 let TheSystem: ctState in
80 let TheActor:actAdministrator in
81 self.rnActor.rnSystem = TheSystem
82 and self.rnActor = TheActor
83
84 /* PreP01 */
85 and TheSystem.vpStarted = true
86 /* PreP02 */
87 and TheActor.rnctAuthenticated.vpIsLogged = true
88 }
89 preF{
90 /* PreF01 */
91 true
92 }
93 postF{
94 let TheSystem: ctState in
95 let TheactAuthenticated:actAuthenticated in
96 let AptStringMessageForTheactAuthenticated: ptString in
97
98 self.rnActor.rnSystem = TheSystem
99 and self.rnActor = TheactAuthenticated
100
101 /* PostF01 */
102 AptStringMessageForTheactAuthenticated.eq('You are logged out ! Good Bye ...')
103 and TheactAuthenticated.rnInterfaceIN^ieMessage(AptStringMessageForTheactAuthenticated)
104 }
105 postP{
106 let TheSystem: ctState in
107 let TheactAuthenticated:actAuthenticated in
108
109 self.rnActor.rnSystem = TheSystem
110 and self.rnActor = TheactAuthenticated.asset
111 /* PostP01 */
112 TheactAuthenticated.rnctAuthenticated@post.vpIsLogged = false
113 }
114 prolog{"src/Operations/Environment/OUT/outactAuthenticated-oeLogout.pl"}
115 }
116 }
117 }
```

Listing B.9: Messir Spec. file environment-actAuthenticated.msr.

## B.10 File ./src-gen/messir-spec/operations/environment/environment-actComCompany.msr

```
1 // Do not add/remove lines because code is inserted in slides
2
3 package icrash.operations.environment.actComCompany{
4
5 import lu.uni.lassy.messir.libraries.primitives
6 import lu.uni.lassy.messir.libraries.calendar
7 import lu.uni.lassy.messir.libraries.math
8
9 import icrash.concepts.primarytypes.datatypes
10 import icrash.concepts.primarytypes.classes
11 import icrash.concepts.secondarytypes.datatypes
12
13 import icrash.environment
14
15 Operation Model {
```

```

16
17 operation: actComCompany.outactComCompany.oeAlert(
18   AetKind:etHumanKind,
19   AdtMyDate:dtDate,
20   AdtTime:dtTime,
21   AdtPhoneNumber:dtPhoneNumber,
22   AdtGPSLocation:dtGPSLocation,
23   AdtComment:dtComment
24 ) :ptBoolean{
25
26 preP{
27   let TheSystem: ctState in
28   self.rnActor.rnSystem = TheSystem
29
30 /* PreP01 */
31 and TheSystem.vpStarted = true
32 }
33 pref{
34   let TheSystem: ctState in
35   self.rnActor.rnSystem = TheSystem
36
37 /* PreF01 */
38 and (TheSystem.clock.date.gt(AdtDate)
39   or (TheSystem.clock.date.eq(AdtDate)
40     and TheSystem.clock.time.gt(AdtTime)
41   )
42 )
43 }
44 postF{
45   let TheSystem: ctState in
46
47   let ActHuman:ctHuman in
48   let TheactComCompany:actComCompany in
49   let ActAlert:ctAlert in
50   let AAlertInstant:dtDateAndTime in
51   let AetAlertStatus:etAlertStatus in
52   let ActAlertNearBy:ctAlert in
53   let ActCrisis:ctCrisis in
54   let AdtCrisisID:dtCrisisID in
55   let AetCrisisType:etCrisisType in
56   let AetCrisisStatus:etCrisisStatus in
57   let ACrisisInstant:dtDateAndTime in
58   let ACrisisdtComment:dtComment in
59   let AptStringMessage:ptString in
60   let AdtSMS:dtSMS in
61   let AdtAlertID:dtAlertID in
62
63   self.rnActor.rnSystem = TheSystem
64   and self.rnActor = TheactComCompany
65 /* PostF01 */
66 TheSystem.nextValueForAlertID=PrenextValueForAlertID
67 and PrenextValueForAlertID.add(1) = PostnextValueForAlertID
68 and TheSystem@post.nextValueForAlertID = PostnextValueForAlertID
69
70 /* PostF02 */
71 and AAlertInstant.date=AdtDate
72 and AAlertInstant.time=AdtTime
73
74 and AetAlertStatus=pending
75
76 and TheSystem.nextValueForAlertID.todtString().eq(AdtAlertID)
77
78 and ActAlert.init(AdtAlertID,
79   AetAlertStatus,
80   AdtGPSLocation,
81   AAlertInstant,
82   AdtComment)
83
84 /* PostF03 */
85 and TheSystem.rnctAlert.select(location.isNearTo(AdtGPSLocation)) = ColctAlertsNearBy

```

## B.11. FILE /SRC-GEN/MESSIR-SPEC.../ENVIRONMENT-ACTCOORDINATOR-OECLOSECRISIS.MSR197

```

86 and if (ColctAlertsNearBy->size()=0)
87 then (TheSystem.nextValueForCrisisID = PrenextValueForCrisisID
88 and PrenextValueForCrisisID.add(1) = PostnextValueForCrisisID
89 and TheSystem@post.nextValueForCrisisID = PostnextValueForCrisisID
90 and TheSystem.nextValueForCrisisID.todtString().eq(AdtCrisisID)
91 and AdtCrisisType = small
92 and AetCrisisStatus = pending
93 and ACrisisInstant= AAlertInstant
94 and ACrisisdtComment = 'no reporting yet defined'
95 and ActCrisis.init( AdtCrisisID,
96         AdtCrisisType,
97         AetCrisisStatus,
98         AdtGPSLocation,
99         ACrisisInstant,
100        ACrisisdtComment)
101    )
102 else (ColctAlertsNearBy.rnTheCrisis->msrAny(true) = ActCrisis)
103 endif
104
105 /* PostF04 */
106 and ActAlert@post.rnTheCrisis = ActCrisis
107
108 /* PostF05 */
109 and TheSystem.rnctHuman->select(id.eq(AdtPhoneNumber)) = HumanColl
110
111 and HumanColl->select(kind.etEq(AetHumanKind)) = HumanCol2
112 and if (HumanCol2->msrIsEmpty)
113 then (ActHuman.init(AdtPhoneNumber,AetHumanKind)
114 and ActHuman@post.rnactComCompany = TheactComCompany
115 )
116 else (HumanCol2->any(true) = ActHuman)
117 endif
118
119 and ActHuman.rnSignaled->msrIncluding(ActAlert) = ColAlerts
120
121 and ActHuman@post.rnSignaled = ColAlerts
122
123 /* PostF06 */
124 AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
125 and TheactComCompany.rnInterfaceIN^ieSmsSend(AdtPhoneNumber,AdtSMS)
126 }
127 /* Post Protocol:*/
128 /* PostP01 */
129 postP{true}
130
131 prolog{"src/Operations/Environment/OUT/outactComCompany-oeAlert.pl"}
132 }
133 }
134 }
```

Listing B.10: Messir Spec. file environment-actComCompany.msr.

## B.11 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeCloseCrisis.msr

```

1 package icrash.operations.environment.actCoordinator.oeCloseCrisis {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeCloseCrisis.pl"}
```

```

14 }
15 }
16 }
```

Listing B.11: Messir Spec. file environment-actCoordinator-oeCloseCrisis.msr.

## B.12 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeGetAlertsSet.msr

```

1 package icrash.operations.environment.actCoordinator.oeGetAlertsSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.datatypes
9 import icrash.environment
10
11 Operation Model {
12
13 operation: actCoordinator.outactCoordinator.oeGetAlertsSet(AetAlertStatus:etAlertStatus):ptBoolean{
14 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeGetAlertsSet.pl"}
15 }
16 }
17 }
```

Listing B.12: Messir Spec. file environment-actCoordinator-oeGetAlertsSet.msr.

## B.13 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeGetCrisisSet.msr

```

1 package icrash.operations.environment.actCoordinator.oeGetCrisisSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean
13 {
14 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeGetCrisisSet.pl"}
15 }
16 }
```

Listing B.13: Messir Spec. file environment-actCoordinator-oeGetCrisisSet.msr.

## B.14 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeInvalidateAlert.msr

```

1 package icrash.operations.environment.actCoordinator.oeInvalidateAlert {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
```

## B.15. FILE /SRC-GEN/MESSIR-SPEC.../ENVIRONMENT-ACTCOORDINATOR-OEREPORTONCRISIS.MSR

```
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeInvalidateAlert(AdtAlertID:dtAlertID):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeInvalidateAlert.pl"}
14 }
15 }
16 }
```

Listing B.14: Messir Spec. file environment-actCoordinator-oeInvalidateAlert.msr.

## B.15 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeReportOnCrisis.msr

```
1 package icrash.operations.environment.actCoordinator.oeReportOnCrisis {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:
13 dtComment):ptBoolean{
14 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeReportOnCrisis.pl"}
15 }
16 }
17 }
```

Listing B.15: Messir Spec. file environment-actCoordinator-oeReportOnCrisis.msr.

## B.16 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeSetCrisisHandler.msr

```
1 package icrash.operations.environment.actCoordinator.oeSetCrisisHandler {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.datatypes
9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.secondarytypes.datatypes
11 import icrash.environment
12
13 Operation Model {
14
15 operation: actCoordinator.outactCoordinator.oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean{
16 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeSetCrisisHandler.pl"}
17 }
18
19 }
20 }
```

Listing B.16: Messir Spec. file environment-actCoordinator-oeSetCrisisHandler.msr.

## B.17 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeSetCrisisStatus.msr

```

1 package icrash.operations.environment.actCoordinator.oeSetCrisisStatus {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeSetCrisisStatus(AdtCrisisID:dtCrisisID,
    AetCrisisStatus:etCrisisStatus):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeSetCrisisStatus.pl"}
14 }
15
16 }
17 }
```

Listing B.17: Messir Spec. file environment-actCoordinator-oeSetCrisisStatus.msr.

### B.18 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeSetCrisisType.msr

```

1 package icrash.operations.environment.actCoordinator.oeSetCrisisType {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeSetCrisisType(AdtCrisisID:dtCrisisID, AetCrisisType:
    etCrisisType):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeSetCrisisType.pl"}
14 }
15
16 }
17 }
```

Listing B.18: Messir Spec. file environment-actCoordinator-oeSetCrisisType.msr.

### B.19 File ./src-gen/messir-spec/operations/environment/environment-actCoordinator-oeValidateAlert.msr

```

1 package icrash.operations.environment.actCoordinator.oeValidateAlert {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actCoordinator.outactCoordinator.oeValidateAlert(AdtAlertID:dtAlertID):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactCoordinator-oeValidateAlert.pl"}
14 }
15
16 }
17 }
```

Listing B.19: Messir Spec. file environment-actCoordinator-oeValidateAlert.msr.

## B.20 File ./src-gen/messir-spec/operations/environment/environment-actMsrCreator-init.msr

```

1 package icrash.operations.icrash.environment.actMsrCreator.init {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.environment
5
6 Operation Model {
7
8 operation: actMsrCreator.init():ptBoolean{}
9 // generic operation provided by the simulator
10 }
11 }
```

Listing B.20: Messir Spec. file environment-actMsrCreator-init.msr.

## B.21 File ./src-gen/messir-spec/operations/environment/environment-actMsrCreator-oeCreateSystemAndEnvironment.msr

```

1 package icrash.operations.environment.actMsrCreator.oeCreateSystemAndEnvironment{
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.concepts.primarytypes.classes
9 import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11 import icrash.environment
12
13 Operation Model {
14
15 operation: actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger):
16     ptBoolean
17 {preP{true}
18 preF{true}
19 postF{
20     let TheSystem: ctState in
21     let AactMsrCreator: actMsrCreator in
22     let AactAdministrator: actAdministrator in
23     let AnextValueForAlertID: dtInteger in
24     let AnextValueForCrisisID: dtInteger in
25     let Aclock: dtDateAndTime in
26     let AcrisisReminderPeriod: dtSecond in
27     let AmaxCrisisReminderPeriod: dtSecond in
28     let AvpStarted: ptBoolean in
29
30     /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
31     AnextValueForAlertID.value.eq(1)
32     and AnextValueForCrisisID.value.eq(1)
33     and Aclock.date.year.value = 1970
34     and Aclock.date.month.value = 01
35     and Aclock.date.day.value = 01
36     and Aclock.time.hour.value = 00
37     and Aclock.time.minute.value = 00
38     and Aclock.time.second.value = 00
39     and AcrisisReminderPeriod.value.eq(300)
40     and AmaxCrisisReminderPeriod.value.eq(1200)
41     and AvpStarted = true
42     and TheSystem.init(AnextValueForAlertID,
43         AnextValueForCrisisID,
44         Aclock,
45         AcrisisReminderPeriod,
46         AmaxCrisisReminderPeriod,
```

```

47         Aclock,
48         AvpStarted
49     )
50 /* PostF02*/
51 and AactMsrCreator.init()
52 /* PostF03 */
53 and let AactComCompanyCol: Bag(actComCompany) in
54 AactComCompanyCol->size() = AqtyComCompanies
55 AactComCompanyCol-> forAll(init())
56 /* PostF04*/
57 and AactAdministrator.init()
58 /* PostF05*/
59 and let AactActivator:actActivator in
60 AactActivator.init()
61 /* PostF06 */
62 and let ActAdministrator:ctAdministrator in
63   let AdtLogin:dtLogin in
64   let AdtPassword:dtPassword in
65   AdtLogin.value.eq('icrashadmin')
66   and AdtPassword.value.eq('7WXC1359')
67   and ActAdministrator.init(AdtLogin,AdtPassword)
68 /* PostF07*/
69 and ActAdministrator@post.rnactAuthenticated = AactAdministrator}
70 postP{true}
71
72 prolog{ "src/Operations/Environment/OUT/outactMsrCreator-oeCreateSystemAndEnvironment.pl"}
73
74 }
75 }
76
77 }
```

Listing B.21: Messir Spec. file environment-actMsrCreator-oeCreateSystemAndEnvironment.msr.

## B.22 File ./src-gen/messir-spec/operations/environment/environment-actPolice-oeCloseCrisis.msr

```

1 package icrash.operations.environment.actPolice.oeCloseCrisis {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actPolice.outactPolice.oeCloseCrisis(AdtCrisisID:dtCrisisID):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactPolice-oeCloseCrisis.pl"}
14 }
15 }
16 }
```

Listing B.22: Messir Spec. file environment-actPolice-oeCloseCrisis.msr.

## B.23 File ./src-gen/messir-spec/operations/environment/environment-actPolice-oeGetCrisisSet.msr

```

1 package icrash.operations.environment.actPolice.oeGetCrisisSet {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
```

## B.24 FILE /SRC-GEN/MESSIR-SPEC.../ENVIRONMENT-ACTPOLICE-OEREPORTONCRISIS.MSR203

```
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actPolice.outactPolice.oeGetCrisisSet(AetCrisisStatus:etCrisisStatus):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactPolice-oeGetCrisisSet.pl"}
14 }
15 }
16 }
```

Listing B.23: Messir Spec. file environment-actPolice-oeGetCrisisSet.msr.

## B.24 File ./src-gen/messir-spec/operations/environment/environment-actPolice-oeReportOnCrisis.msr

```
1 package icrash.operations.environment.actPolice.oeReportOnCrisis {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actPolice.outactPolice.oeReportOnCrisis(AdtCrisisID:dtCrisisID, AdtComment:dtComment):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactPolice-oeReportOnCrisis.pl"}
14 }
15 }
16 }
17 }
```

Listing B.24: Messir Spec. file environment-actPolice-oeReportOnCrisis.msr.

## B.25 File ./src-gen/messir-spec/operations/environment/environment-actPolice-oeSetCrisisHandler.msr

```
1 package icrash.operations.environment.actPolice.oeSetCrisisHandler {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.datatypes
9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.secondarytypes.datatypes
11 import icrash.environment
12
13 Operation Model {
14
15 operation: actPolice.outactPolice.oeSetCrisisHandler(AdtCrisisID:dtCrisisID):ptBoolean{
16 prolog{"src/Operations/Environment/OUT/outactPolice-oeSetCrisisHandler.pl"}
17 }
18 }
19 }
20 }
```

Listing B.25: Messir Spec. file environment-actPolice-oeSetCrisisHandler.msr.

## B.26 File ./src-gen/messir-spec/operations/environment/environment-actPolice-oeSetCrisisStatus.msr

```

1 package icrash.operations.environment.actPolice.oeSetCrisisStatus {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5 import lu.uni.lassy.messir.libraries.string
6 import lu.uni.lassy.messir.libraries.calendar
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.environment
9
10 Operation Model {
11
12 operation: actPolice.outactPolice.oeSetCrisisStatus(AdtCrisisID:dtCrisisID, AetCrisisStatus:
    etCrisisStatus):ptBoolean{
13 prolog{"src/Operations/Environment/OUT/outactPolice-oeSetCrisisStatus.pl"}
14 }
15
16 }
17 }
```

Listing B.26: Messir Spec. file environment-actPolice-oeSetCrisisStatus.msr.

## B.27 File ./src-gen/messir-spec/environment/environment.msr

```

1 package icrash.environment{
2
3 import icrash.concepts.primarytypes.datatypes
4 import icrash.concepts.primarytypes.classes
5 import icrash.concepts.secondarytypes.datatypes
6 import lu.uni.lassy.messir.libraries.primitives
7 import lu.uni.lassy.messir.libraries.math
8 import lu.uni.lassy.messir.libraries.calendar
9
10 Environment Model {
11
12 actor actMsrCreator role rnactMsrCreator cardinality [1..1] {
13
14     operation init():ptBoolean
15
16     input interface inactMsrCreator {
17     }
18     output interface outactMsrCreator {
19         operation oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger ):ptBoolean
20     }
21 }
22
23 actor actAdministrator
24     role rnactAdministrator
25     cardinality [1..1]
26     extends actAuthenticated {
27
28     operation init():ptBoolean
29
30     output interface outactAdministrator{
31
32         operation oeAddCoordinator(
33             AdtCoordinatorID:dtCoordinatorID ,
34             AdtLogin:dtLogin ,
35             AdtPassword:dtPassword ):ptBoolean
36
37         operation oeDeleteCoordinator(
38             AdtCoordinatorID:dtCoordinatorID ):ptBoolean
39
40         operation oeAddPolice(
41             AdtPoliceID:dtPoliceID ,
42             AdtLogin:dtLogin ,
43             AdtPassword:dtPassword ):ptBoolean
44
45         operation oeDeletePolice(
46             AdtPoliceID:dtPoliceID ):ptBoolean
47 }
```

```

47 }
48
49 input interface inactAdministrator{
50
51     operation ieCoordinatorAdded():ptBoolean
52     operation ieCoordinatorDeleted():ptBoolean
53
54     operation iePoliceAdded():ptBoolean
55     operation iePoliceDeleted():ptBoolean
56 }
57 }
58
59 actor actCoordinator
60     role rnactCoordinator
61     cardinality [0...*]
62     extends actAuthenticated{
63
64     operation init():ptBoolean
65
66     output interface outactCoordinator{
67         operation oeInvalidateAlert(AdtAlertID:dtAlertID ):ptBoolean
68         operation oeCloseCrisis(AdtCrisisID:dtCrisisID ):ptBoolean
69         operation oeGetAlertsSet(AetAlertStatus:etAlertStatus ):ptBoolean
70         operation oeGetCrisisSet(AetCrisisStatus:etCrisisStatus ):ptBoolean
71         operation oeSetCrisisHandler(AdtCrisisID:dtCrisisID ):ptBoolean
72         operation oeReportOnCrisis(
73             AdtCrisisID:dtCrisisID ,
74             AdtComment:dtComment
75             ):ptBoolean
76         operation oeSetCrisisStatus(
77             AdtCrisisID:dtCrisisID ,
78             AetCrisisStatus:etCrisisStatus
79             ):ptBoolean
80         operation oeSetCrisisType(
81             AdtCrisisID:dtCrisisID ,
82             AetCrisisType:etCrisisType
83             ):ptBoolean
84         operation oeValidateAlert(AdtAlertID:dtAlertID ):ptBoolean
85     }
86
87     input interface inactCoordinator{
88         operation ieSendAnAlert(ActAlert:ctAlert ):ptBoolean
89         operation ieSendACrisis(ActCrisis:ctCrisis ):ptBoolean
90     }
91 }
92
93 actor actPolice role rnactPolice cardinality [0...*] extends actAuthenticated {
94
95     operation init():ptBoolean
96     input interface inactPolice {
97         operation ieSendACrisis(ActCrisis:ctCrisis ):ptBoolean
98     }
99     output interface outactPolice {
100         operation oeCloseCrisis(AdtCrisisID:dtCrisisID ):ptBoolean
101         operation oeGetCrisisSet(AetCrisisStatus:etCrisisStatus ):ptBoolean
102         operation oeSetCrisisHandler(AdtCrisisID:dtCrisisID ):ptBoolean
103         operation oeReportOnCrisis(
104             AdtCrisisID:dtCrisisID ,
105             AdtComment:dtComment
106             ):ptBoolean
107         operation oeSetCrisisStatus(
108             AdtCrisisID:dtCrisisID ,
109             AetCrisisStatus:etCrisisStatus
110             ):ptBoolean
111     }
112 }
113
114 actor actComCompany role rnactComCompany cardinality [0...*]{
115
116     operation init():ptBoolean

```

```

117
118   output interface outactComCompany{
119     operation oeAlert(
120       AetHumanKind:etHumanKind ,
121       AdtDate:dtDate ,
122       AdtTime:dtTime ,
123       AdtPhoneNumber:dtPhoneNumber ,
124       AdtGPSLocation:dtGPSLocation ,
125       AdtComment:dtComment
126     ):ptBoolean
127   }
128
129   input interface inactComCompany{
130     operation ieSmsSend(AdtPhoneNumber:dtPhoneNumber ,
131       AdtSMS:dtsMS
132     ):ptBoolean
133   }
134 }
135
136 actor actAuthenticated role rnactAuthenticated cardinality [0...*]{
137
138   operation init():ptBoolean
139
140   output interface outactAuthenticated{
141     operation oeLogin(AdtLogin:dtLogin , AdtPassword:dtPassword ):ptBoolean
142     operation oeLogout():ptBoolean
143   }
144
145   input interface inactAuthenticated{
146     operation ieMessage(AMessage:ptString):ptBoolean
147   }
148 }
149
150 actor actActivator[proactive] role rnactActivator cardinality [1..1]{
151
152   operation init():ptBoolean
153
154   output interface outactActivator{
155     proactive operation oeSollicitateCrisisHandling():ptBoolean
156     proactive operation oeSetClock(AcurrentClock:dtDateAndTime ):ptBoolean
157   }
158
159   input interface inactActivator{
160   }
161 }
162 }
163 }
```

Listing B.27: Messir Spec. file environment.msr.

## B.28 File [./src-gen/messir-spec/concepts/primarytypes-associations.msr](#)

```

1 package icrash.concepts.primarytypes.associations {
2
3 import icrash.concepts.primarytypes.datatypes
4 import icrash.concepts.primarytypes.classes
5 import icrash.environment
6 import lu.uni.lassy.messir.libraries.primitives
7
8 Concept Model {
9
10 Primary Types{
11
12 // Internal
13
14 association assctAlertctCrisis
15 ctAlert(rnAlerts)[1...*]
```

## B.29. FILE /SRC-GEN/MESSIR-SPEC.../PRIMARYTYPES-CLASSES-CTADMINISTRATOR.MSR207

```
16 ctCrisis (rnTheCrisis)[1..1]
17
18 association assctAlertctHuman
19 ctAlert(rnSignaled)[1..*]
20 ctHuman (rnSignaler)[1..1]
21
22 association assctCrisisctPolice
23 ctCrisis(rnHandled)[0..*]
24 ctPolice(rnHandler)[0..1]
25
26 association assctCrisisctCoordinator
27 ctCrisis(rnHandled)[0..*]
28 ctCoordinator(rnHandler)[0..1]
29
30 // With Actors
31
32 association assctHumanactComCompany
33 ctHuman(rnctHuman)[0..*]
34 actComCompany(rnactComCompany)[1..1]
35
36 association assctCoordinatoractCoordinator
37 ctCoordinator(rnctCoordinator)[1..1]
38 actCoordinator(rnactCoordinator)[1..1]
39
40 association assctPoliceactPolice
41 ctPolice(rnctPolice)[1..1]
42 actPolice(rnactPolice)[1..1]
43
44 association assctAuthenticatedactAuthenticated
45 ctAuthenticated(rnctAuthenticated)[1..1]
46 actAuthenticated(rnactAuthenticated)[1..1]
47
48 }
49 }
50 }
```

Listing B.28: Messir Spec. file primarytypes-associations.msr.

## B.29 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctAdministrator.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctAdministrator.init {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.concepts.primarytypes.classes
6
7 Operation Model {
8
9 operation: icrash.concepts.primarytypes.classes.ctAdministrator.init(Alogin:dtLogin, Apwd:dtPassword
   ):ptBoolean
10 {
11 postF{
12 if
13 (
14 let Self:ctAdministrator in
15 /* Post F01 */
16 Self.login = Alogin
17 //Self.login(Alogin)
18 and Self.pwd = Apwd
19 and Self.vpIsLogged = false
20
21 /* Post F02 */
22 and (Self.oclIsNew and self = Self)
23 )
24 then (result = true)
25 else (result = false)
26 endif
```

```

27 }
28 prolog{ "src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAdministrator-init.pl"
29 }
30 }
31 }

```

Listing B.29: Messir Spec. file primarytypes-classes-ctAdministrator.msr.

### B.30 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctAlert.msr

```

1 package icrash.operations.concepts.primarytypes.classes.ctAlert{
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.calendar
5
6 import icrash.concepts.primarytypes.datatypes
7 import icrash.concepts.primarytypes.classes
8
9 import icrash.environment
10
11 Operation Model {
12
13 operation: icrash.concepts.primarytypes.classes.ctAlert.init(Aid:dtAlertID , Astatus:etAlertStatus ,
   Alocation:dtGPSLocation , Ainstant:dtDateAndTime , Acomment:dtComment
14 ):ptBoolean{
15 postF{
16 if
17 (
18 /* Post F01 */
19 let Self:ctAlert in
20 Self.id = Aid
21 and Self.status = Astatus
22 and Self.location = Alocation
23 and Self.instant = Ainstant
24 and Self.comment = Acomment
25 /* Post F02 */
26 and (Self.ocliIsNew and self = Self)
27 )
28 then (result = true)
29 else (result = false)
30 endif
31 }
32 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAlert-init.pl"}
33 }
34
35 operation: icrash.concepts.primarytypes.classes.ctAlert.isSentToCoordinator(AactCoordinator:
   actCoordinator ):ptBoolean
36 {
37 postF{
38 if
39 (
40 /* Post F01 */
41 AactCoordinator.rnInterfaceIN.ieSendAnAlert(self)
42 )
43 then (result = true)
44 else (result = false)
45 endif
46 }
47 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAlert-isSentToCoordinator.
   pl"}
48
49 }
50 }
51 }

```

Listing B.30: Messir Spec. file primarytypes-classes-ctAlert.msr.

### B.31 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctAuthenticated.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctAuthenticated {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.concepts.primarytypes.classes
6
7 Operation Model {
8
9 operation: icrash.concepts.primarytypes.classes.ctAuthenticated.init(Alogin:dtLogin, Apwd:dtPassword
10   ):ptBoolean{
11 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctAuthenticated-init.pl"}
12 }
13
14 }
```

Listing B.31: Messir Spec. file primarytypes-classes-ctAuthenticated.msr.

### B.32 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctCoordinator.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctCoordinator.init {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.concepts.primarytypes.classes
6
7 Operation Model {
8
9 operation: icrash.concepts.primarytypes.classes.ctCoordinator.init(Aid:dtCoordinatorID, Alogin:
10   dtLogin, Apwd:dtPassword):ptBoolean
11 {
12 if
13 (
14 /* Post F01 */
15 let Self:ctCoordinator in
16 Self.id = Aid
17 and Self.login = Alogin
18 and Self.pwd = Apwd
19 and Self.vpIsLogged = false
20 /* Post F02 */
21 and (Self.oclIsNew and self = Self)
22 )
23 then (result = true)
24 else (result = false)
25 endif}
26 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCoordinator-init.pl"}
27 }
28 }
29 }
```

Listing B.32: Messir Spec. file primarytypes-classes-ctCoordinator.msr.

### B.33 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctCrisis.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctCrisis {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
```

```

5 import lu.uni.lassy.messir.libraries.calendar
6
7 import icrash.concepts.primarytypes.datatypes
8 import icrash.concepts.primarytypes.classes
9 import icrash.concepts.secondarytypes.datatypes
10 import icrash.concepts.secondarytypes.classes
11 import lu.uni.lassy.messir.libraries.primitives
12
13 import icrash.environment
14
15 Operation Model {
16 //-----
17 operation: icrash.concepts.primarytypes.classes.ctCrisis.init(
18     Aid:dtCrisisID,
19     Atype:etCrisisType,
20     Astatus:etCrisisStatus,
21     Alocation:dtGPSLocation,
22     Ainstant:dtDateAndTime,
23     Acomment:dtComment
24 ) :ptBoolean{
25 postF{
26 if
27 (
28 /* Post F01 */
29 let Self:ctCrisis in
30 Self.id = Aid
31 and Self.type = Atype
32 and Self.status = Astatus
33 and Self.location = Alocation
34 and Self.instant = Ainstant
35 and Self.comment = Acomment
36 /* Post F02 */
37 and (Self.oclIsNew and self = Self)
38 )
39 then (result = true)
40 else (result = false)
41 endif}
42 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-init.pl"}
43 //-----
44 operation: icrash.concepts.primarytypes.classes.ctCrisis.handlingDelayPassed():ptBoolean
45 {
46 postF{
47 let TheSystem:ctState in
48 let CurrentClockSecondsQty:dtInteger in
49 let vpLastReminderSecondsQty:dtInteger in
50 let CrisisReminderPeriod:dtSecond in
51 if
52 ( /* Post F01 */
53 self.rnSystem = TheSystem
54 and self.status = pending
55 and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
56 and TheSystem.vpLastReminder.toSecondsQty() = vpLastReminderSecondsQty
57 and TheSystem.crisisReminderPeriod = CrisisReminderPeriod
58 and CurrentClockSecondsQty.sub(vpLastReminderSecondsQty).gt(CrisisReminderPeriod) = true
59 )
60 then (result = true)
61 else (result = false)
62 endif
63 }
64 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-handlingDelayPassed
       .pl"}
65 //-----
66 operation: icrash.concepts.primarytypes.classes.ctCrisis.maxHandlingDelayPassed():ptBoolean
67 {
68 postF{
69 let TheSystem:ctState in
70 let CurrentClockSecondsQty:dtInteger in
71 let CrisisInstantSecondsQty:dtInteger in
72 let MaxCrisisReminderPeriod:dtSecond in
73 if

```

### B.33. FILE /SRC-GEN/MESSIR-SPEC/OPERATIONS.../PRIMARYTYPES-CLASSES-CTCRISIS.MSR211

```
74 ( /* Post F01 */
75 self.rnSystem = TheSystem
76 and self.status = pending
77 and TheSystem.clock.toSecondsQty() = CurrentClockSecondsQty
78 and Self.instant.toSecondsQty() = CrisisInstantSecondsQty
79 and TheSystem.maxCrisisReminderPeriod = MaxCrisisReminderPeriod
80 and CurrentClockSecondsQty.sub(CrisisInstantSecondsQty)
81 .gt(MaxCrisisReminderPeriod)
82 )
83 then (result = true)
84 else (result = false)
85 endif
86 }
87 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-
maxHandlingDelayPassed.pl"}
88 //-----
89 operation: icrash.concepts.primarytypes.classes.ctCrisis.isSentToCoordinator(AactCoordinator:
actCoordinator):ptBoolean
90 {
91 postF{
92 if
93 (
94 /* Post F01 */
95 AactCoordinator.rnInterfaceIN.ieSendACrisis(self)
96 )
97 then (result = true)
98 else (result = false)
99 endif}
100 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToCoordinator
.pi"}
101 //-----
102 operation: icrash.concepts.primarytypes.classes.ctCrisis.isSentToPolice(AactPolice:actPolice):
ptBoolean
103 {
104 postF{
105 if
106 (
107 /* Post F01 */
108 AactPolice.rnInterfaceIN.ieSendACrisis(self)
109 )
110 then (result = true)
111 else (result = false)
112 endif}
113 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-isSentToPolice.pl"
}
114 //-----
115 operation: icrash.concepts.primarytypes.classes.ctCrisis.isAllocatedIfPossible():ptBoolean
116 {
117 postF{
118 if (
119 /* Post F01 */
120 self.maxHandlingDelayPassed()
121 and
122 if (TheSystem.rnactCoordinator->msrIsEmpty = false)
123 then (
124 /* Post F02 */
125 TheSystem.rnactCoordinator->msrAny(true) = TheCoordinatorActor
126 and TheCoordinatorActor.rnctCoordinator = TheCoordinator
127 and self@post.rnHandler = TheCoordinator
128 and self@post.status = handled
129 and self.id.value = TheCrisisIDptString
130 and 'You are now considered as handling the crisis having ID: '
131 .ptStringConcat(TheCrisisIDptString) = TheMessage
132 and TheCoordinatorActor.rnInterfaceIN^ieMessage(TheMessage)
133 )
134 else ( /* Post F03 */
135 TheSystem.rnactAdministrator
136 ->forall(rnInterfaceIN.ieMessage('Please add new coordinators to handle pending crisis !'))
137 )
138 endif
```

```

139  )
140 then (result = true)
141 else (result = false)
142 endif
143 }
144 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctCrisis-
    isAllocatedIfPossible.pl"}
145 }
146 }
147 }
```

Listing B.33: Messir Spec. file primarytypes-classes-ctCrisis.msr.

### B.34 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctHuman.msr

```

1 package icrash.operations.concepts.primarytypes.classes.ctHuman.init {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5
6 import icrash.concepts.primarytypes.classes
7
8 Operation Model {
9
10 operation: icrash.concepts.primarytypes.classes.ctHuman.init(Aid:dtPhoneNumber, Akind:etHumanKind):
    ptBoolean
11 {
12 postF{
13 if
14 (
15 /* Post F01 */
16 let Self:ctHuman in
17
18 Self.id = Aid
19 and Self.kind = Akind
20
21 /* Post F02 */
22 and (Self.oclIsNew and self = Self)
23 )
24 then (result = true)
25 else (result = false)
26 endif
27 }
28 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctHuman-init.pl"}
29 }
30 operation: icrash.concepts.primarytypes.classes.ctHuman.isAcknowledged():ptBoolean{
31 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctHuman-isAcknowledged.pl"}
32 }
33 }
34 }
```

Listing B.34: Messir Spec. file primarytypes-classes-ctHuman.msr.

### B.35 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctPolice.msr

```

1 package icrash.operations.concepts.primarytypes.classes.ctPolice.init {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.concepts.primarytypes.classes
6
7 Operation Model {
8 }
```

### B.36. FILE /SRC-GEN/MESSIR-SPEC/OPERATIONS.../PRIMARYTYPES-CLASSES-CTSTATE.MSR213

```
9 operation: icrash.concepts.primarytypes.classes.ctPolice.init(Aid:dtPoliceID, Alogin:dtLogin, Apwd:
    dtPassword):ptBoolean
10 {
11 postF{
12 if
13 (
14 /* Post F01 */
15 let Self:ctPolice in
16 Self.id = Aid
17 and Self.login = Alogin
18 and Self.pwd = Apwd
19 and Self.vpIsLogged = false
20 /* Post F02 */
21 and (Self.oclIsNew and self = Self)
22 )
23 then (result = true)
24 else (result = false)
25 endif
26 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctPolice-init.pl"}
27 }
28 }
29 }
```

Listing B.35: Messir Spec. file primarytypes-classes-ctPolice.msr.

### B.36 File ./src-gen/messir-spec/operations/concepts/primarytypes-classes/primarytypes-classes-ctState.msr

```
1 package icrash.operations.concepts.primarytypes.classes.ctState{
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.calendar
5 import lu.uni.lassy.messir.libraries.math
6
7 import icrash.concepts.primarytypes.classes
8
9 Operation Model {
10
11 operation: icrash.concepts.primarytypes.classes.ctState.init(
12 AnextValueForAlertID: dtInteger,
13 AnextValueForCrisisID: dtInteger ,
14 dtAclock:dtDateAndTime,
15 AcrisisReminderPeriod: dtSecond,
16 AmaxCrisisReminderPeriod: dtSecond ,
17 AvpLastReminder: dtDateAndTime ,
18 AvpStarted:ptBoolean ):ptBoolean{
19 postF{
20 if
21 (
22 /* Post F01 */
23 let Self:ctState in
24
25 Self.nextValueForAlertID = AnextValueForAlertID
26 and Self.nextValueForCrisisID = AnextValueForCrisisID
27 and Self.clock = Aclock
28 and Self.crisisReminderPeriod = AcrisisReminderPeriod
29 and Self.maxCrisisReminderPeriod = AmaxCrisisReminderPeriod
30 and Self.vpLastReminder = AvpLastReminder
31 and Self.vpStarted = AvpStarted
32
33 and (Self.oclIsNew and self = Self)
34 )
35 then (result = true)
36 else (result = false)
37 endif
38 }
39 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesClasses-ctState-init.pl" }
40 }
```

$$\begin{array}{r} 41 \\ 42 \end{array} \quad \left. \begin{array}{l} \\ \end{array} \right\}$$

Listing B.36: Messir Spec. file primarytypes-classes-ctState.msr.

B.37 File ./src-gen/messir-spec/concepts/primarytypes-classes.msr

```

1 package icrash.concepts.primarytypes.classes {
2
3 import icrash.concepts.primarytypes.datatypes
4 import icrash.environment
5 import lu.uni.lassy.messir.libraries.primitives
6 import lu.uni.lassy.messir.libraries.math
7 import lu.uni.lassy.messir.libraries.calendar
8
9 Concept Model {
10
11 Primary Types{
12
13 state class ctState {
14     attribute nextValueForAlertID:dtInteger
15     attribute nextValueForCrisisID:dtInteger
16     attribute clock:dtDateAndTime
17     attribute crisisReminderPeriod:dtSecond
18     attribute maxCrisisReminderPeriod:dtSecond
19     attribute vpLastReminder:dtDateAndTime
20     attribute vpStarted:ptBoolean
21
22 operation init( AnextValueForAlertID:dtInteger,
23                 AnextValueForCrisisID:dtInteger,
24                 Aclock:dtDateAndTime,
25                 AcrisisReminderPeriod:dtSecond ,
26                 AmaxCrisisReminderPeriod:dtSecond ,
27                 AvpLastReminder:dtDateAndTime ,
28                 AvpStarted:ptBoolean ): ptBoolean
29 }
30
31 class ctAlert role rnctAlert cardinality [0...*]{
32     attribute id:dtAlertID
33     attribute status: etAlertStatus
34     attribute location:dtGPSLocation
35     attribute instant:dtDateAndTime
36     attribute comment:dtComment
37
38 operation init( Aid:dtAlertID ,
39                 Astatus:etAlertStatus ,
40                 Alocation:dtGPSLocation ,
41                 Ainstant:dtDateAndTime ,
42                 Acomment:dtComment ):ptBoolean
43 operation isSentToCoordinator(AactCoordinator:actCoordinator ):ptBoolean
44
45 }
46
47 class ctCrisis role rnctCrisis cardinality [0...*]{
48     attribute id:dtCrisisID
49     attribute type:etCrisisType
50     attribute status: etCrisisStatus
51     attribute location:dtGPSLocation
52     attribute instant:dtDateAndTime
53     attribute comment:dtComment
54
55 operation init(
56                 Aid:dtCrisisID ,
57                 Atype:etCrisisType ,
58                 Astatus:etCrisisStatus ,
59                 Alocation:dtGPSLocation ,
60                 Ainstant:dtDateAndTime ,
61                 Acomment:dtComment ):ptBoolean
62

```

```

63     operation handlingDelayPassed():ptBoolean
64         operation maxHandlingDelayPassed():ptBoolean
65     operation isSentToCoordinator(AactCoordinator:actCoordinator ):ptBoolean
66     operation isAllocatedIfPossible():ptBoolean
67     operation isSentToPolice(AactPolice:actPolice ):ptBoolean
68 }
69
70 class ctHuman role rnctHuman cardinality [0..*]{
71     attribute id:dtPhoneNumber
72     attribute kind:etHumanKind
73
74     operation init(
75         Aid:dtPhoneNumber ,
76         Akind:etHumanKind ):ptBoolean
77     operation isAcknowledged():ptBoolean
78 }
79
80 class ctAuthenticated
81     role rnctAuthenticated
82     cardinality [0..*]{
83
84     attribute login:dtLogin
85     attribute pwd: dtPassword
86     attribute vpIsLogged:ptBoolean
87
88     operation init(
89         Alogin:dtLogin ,
90         Apwd:dtPassword ):ptBoolean
91 }
92
93 class ctCoordinator
94     role rnctCoordinator
95     cardinality [0..*]
96     extends ctAuthenticated{
97
98     attribute id:dtCoordinatorID
99
100    operation init(
101        Aid:dtCoordinatorID ,
102        Alogin:dtLogin ,
103        Apwd:dtPassword ):ptBoolean
104 }
105
106 class ctPolice
107     role rnctPolice
108     cardinality [0..*]
109     extends ctAuthenticated{
110
111     attribute id:dtPoliceID
112
113     operation init(
114         Aid:dtPoliceID ,
115         Alogin:dtLogin ,
116         Apwd:dtPassword ):ptBoolean
117
118 }
119
120 class ctAdministrator
121     role rnctAdministrator
122     cardinality [1..1]
123     extends ctAuthenticated{
124
125     operation init(
126         Alogin:dtLogin ,
127         Apwd:dtPassword ):ptBoolean
128 }
129 }
130 }
```

```
131 }
```

Listing B.37: Messir Spec. file primarytypes-classes.msr.

### B.38 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtAlertID.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtAlertID{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtAlertID.is():ptBoolean{
8
9     postF{
10       let TheResult: ptBoolean in
11       ( if
12         ( AdtValue.value.length().gt(0)
13           and AdtValue.value.length().leq(20)
14         )
15       then (TheResult = true)
16       else (TheResult = false)
17     endif
18     result = TheResult
19   }
20   prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtAlertID-is.pl"}
21 }
22 }
23 }
```

Listing B.38: Messir Spec. file primarytypes-datatypes-dtAlertID.msr.

### B.39 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtComment.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtComment{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtComment.is():ptBoolean{
8
9     postF{
10       let TheResult: ptBoolean in
11       ( if
12         ( MaxLength = 160
13           and AdtValue.value.length().leq(MaxLength)
14         )
15       then (TheResult = true)
16       else (TheResult = false)
17     endif
18     result = TheResult
19   }
20 }
21 prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtComment-is.pl"}
22 }
23 }
24 }
```

Listing B.39: Messir Spec. file primarytypes-datatypes-dtComment.msr.

## B.40 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatype.../primarytypes-datatype.../primarytypes-dtCoordinatorID.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtCoordinatorID{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6   operation: icrash.concepts.primarytypes.datatypes.dtCoordinatorID.is():ptBoolean{
7
8     postF{
9       let TheResult: ptBoolean in
10      ( if
11        ( AdtValue.value.length().gt(0)
12          and AdtValue.value.length().leq(5)
13        )
14        then (TheResult = true)
15        else (TheResult = false)
16      endif
17      result = TheResult
18    )
19  }
20  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtCoordinatorID-is.pl"
21  }
22 }
23 }
```

Listing B.40: Messir Spec. file primarytypes-datatype.../primarytypes-dtCoordinatorID.msr.

## B.41 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatype.../primarytypes-datatype.../dtCrisisID.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtCrisisID{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtCrisisID.is():ptBoolean{
8
9     postF{
10       let TheResult: ptBoolean in
11      ( if
12        ( AdtValue.value.length().gt(0)
13          and AdtValue.value.length().leq(10)
14        )
15        then (TheResult = true)
16        else (TheResult = false)
17      endif
18      result = TheResult
19    )
20  }
21  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtCrisisID-is.pl"
22  }
23 }
24 }
```

Listing B.41: Messir Spec. file primarytypes-datatype.../dtCrisisID.msr.

## B.42 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatype.../primarytypes-datatype.../dtGPSLocation.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtGPSLocation{
```

```

2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.math
5
6 import icrash.concepts.primarytypes.datatypes
7 import icrash.concepts.primarytypes.classes
8 import icrash.concepts.secondarytypes.datatypes
9 import icrash.concepts.secondarytypes.classes
10
11 Operation Model {
12
13   operation: icrash.concepts.primarytypes.datatypes.dtGPSLocation.is():ptBoolean{
14     postF{
15       let TheResult: ptBoolean in
16       ( if
17         ( AdtValue.latitude.is()
18           and AdtValue.longitude.is
19         )
20         then (TheResult = true)
21         else (TheResult = false)
22       endif
23       result = TheResult
24     )
25   }
26   prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtGPSLocation-is.pl"}
27 }
28   operation: icrash.concepts.primarytypes.datatypes.dtGPSLocation.isNearTo(aGPSLocation:
29     dtGPSLocation):ptBoolean{
30     postF{
31       let TheResult: ptBoolean in true
32       let EarthRadius: dtReal in
33       let MaxDistance: dtReal in
34       let ComparedLatitude: dtLatitude in
35       let ComparedLongitude: dtLongitude in
36       let R1: dtReal in let R1a: dtReal in
37       let R2: dtReal in let R2a: dtReal in
38
39       ( if
40         ( EarthRadius.value = 6371
41           and MaxDistance.value = 100
42           and AdtValue.latitude = ComparedLatitude
43           and AdtValue.longitude = ComparedLongitude
44           and Self.latitude.sin() = R1a
45           and AdtValue.latitude.sin().mul(R1a) = R1
46           and Self.latitude.cos() = R2a
47           and AdtValue.latitude.cos().mul(R2a) = R2
48
49           and AdtValue.longitude = ComparedLongitude
50           and Self.longitude.sub(ComparedLongitude).cos().mul(R2)
51             .add(R1).acos().mul(EarthRadius).sub(MaxDistance)
52             .value.leq(0)
53       )
54       then (TheResult = true)
55       else (TheResult = false)
56     endif
57     result = TheResult
58   )
59 }
60   prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtGPSLocation-isNearTo
61     .pl"}
62 }
63   operation: icrash.concepts.primarytypes.datatypes.dtLatitude.is():ptBoolean{
64     postF{
65       let TheResult: ptBoolean in
66       ( if
67         ( AdtValue.value.geq(-90.0)
68           and AdtValue.value.leq(+90.0)
69         )
70         then (TheResult = true)

```

#### B.43 FILE /SRC-GEN/MESSIR-SPEC/OPERATIONS.../PRIMARYTYPES-DATATYPES-DTLOGIN.MSR219

```
70     else (TheResult = false)
71   endif
72   result = TheResult
73 }
74 prolog{ "src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtLatitude-is.pl"
75 }
76 operation: icrash.concepts.primarytypes.datatypes.dtLongitude.is():ptBoolean{
77 postF{
78   let TheResult: ptBoolean in
79   ( if
80     ( AdtValue.value.geq(-180.0)
81       and AdtValue.value.leq(+180.0)
82     )
83     then (TheResult = true)
84     else (TheResult = false)
85   endif
86   result = TheResult
87 }
88 prolog{ "src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtLongitude-is.pl"
89 }
90 }
91 }
```

Listing B.42: Messir Spec. file primarytypes-datatypes-dtGPSLocation.msr.

#### B.43 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtLogin.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtLogin{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtLogin.is():ptBoolean{
8     postF{
9       let TheResult: ptBoolean in
10      let MaxLength: ptInteger in
11      ( if
12        ( MaxLength = 20
13          and AdtValue.value.length().leq(MaxLength)
14        )
15        then (TheResult = true)
16        else (TheResult = false)
17      endif
18      result = TheResult
19    )
20  }
21  prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtLogin-is.pl"}
22 }
23 }
24 }
```

Listing B.43: Messir Spec. file primarytypes-datatypes-dtLogin.msr.

#### B.44 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-dtPassword.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.dtPassword{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtPassword.is():ptBoolean{
8     postF{
```

```

9   let TheResult: ptBoolean in
10  let MinLength: ptInteger in
11  ( if
12    ( MinLength = 6
13      and AdtValue.value.length().geq(MinLength)
14    )
15    then (TheResult = true)
16    else (TheResult = false)
17  endif
18  result = TheResult
19 )
20 }
21 prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtPassword-is.pl"}
22 }
23 }
24 }
```

Listing B.44: Messir Spec. file primarytypes-datatatypes-dtPassword.msr.

#### B.45 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatatypes-dtPhoneNumber.msr

```

1 package icrash.operations.concepts.primarytypes.datatypes.dtPhoneNumber{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.dtPhoneNumber.is():ptBoolean{
8
9   postF{
10   let TheResult: ptBoolean in
11   ( if
12     ( AdtValue.value.length().gt(4)
13       and AdtValue.value.length().leq(30)
14     )
15     then (TheResult = true)
16     else (TheResult = false)
17   endif
18   result = TheResult
19 )
20 }
21 prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtPhoneNumber-is.pl"}
22 }
23 }
24 }
```

Listing B.45: Messir Spec. file primarytypes-datatatypes-dtPhoneNumber.msr.

#### B.46 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatatypes-dtPoliceID.msr

```

1 package icrash.operations.concepts.primarytypes.datatypes.dtPoliceID{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6   operation: icrash.concepts.primarytypes.datatypes.dtPoliceID.is():ptBoolean{
7
8   postF{
9   let TheResult: ptBoolean in
10  ( if
11    ( AdtValue.value.length().gt(0)
12      and AdtValue.value.length().leq(5)
13    )
14    then (TheResult = true)
```

#### B.47 FILE /SRC-GEN/MESSIR-SPEC.../PRIMARYTYPES-DATATYPES-ETALERTSTATUS.MSR221

```
15    else (TheResult = false)
16    endif
17    result = TheResult
18  )
19 }
20 prolog{"src/Operations/Concepts/PrimaryTypesDatatypes/PrimaryTypesDatatypes-dtPoliceID-is.pl"}
21 }
22 }
23 }
```

Listing B.46: Messir Spec. file primarytypes-datatypes-dtPoliceID.msr.

#### B.47 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etAlertStatus.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etAlertStatus{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etAlertStatus.is():ptBoolean{
8     postF{
9       let TheResult: ptBoolean in
10      ( if
11        ( self = pending
12        or self = valid
13        or self = invalid
14      )
15      then (TheResult = true)
16      else (TheResult = false)
17      endif
18      result = TheResult
19    )
20  }
21 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etAlertStatus-is.pl"}
22 }
23 }
24 }
```

Listing B.47: Messir Spec. file primarytypes-datatypes-etAlertStatus.msr.

#### B.48 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etCrisisStatus.msr

```
1 package icrash.operations.concepts.primarytypes.datatypes.etCrisisStatus{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etCrisisStatus.is():ptBoolean{
8     postF{
9       let TheResult: ptBoolean in
10      ( if
11        ( self = pending
12        or self = handled
13        or self = solved
14        or self = closed
15      )
16      then (TheResult = true)
17      else (TheResult = false)
18      endif
19      result = TheResult
20    )
21 }
```

```

22 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etCrisisStatus-is.pl"}
23 }
24 }
25 }
```

Listing B.48: Messir Spec. file primarytypes-datatatypes-etCrisisStatus.msr.

#### B.49 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etCrisisType.msr

```

1 package icrash.operations.concepts.primarytypes.datatypes.etCrisisType{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etCrisisType.is():ptBoolean{
8     postF{
9       let TheResult: ptBoolean in
10      ( if
11        ( self = small
12        or self = medium
13        or self = huge
14      )
15      then (TheResult = true)
16      else (TheResult = false)
17    endif
18    result = TheResult
19  )
20 }
21 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etCrisisType-is.pl"}
22 }
23 }
24 }
```

Listing B.49: Messir Spec. file primarytypes-datatatypes-etCrisisType.msr.

#### B.50 File ./src-gen/messir-spec/operations/concepts/primarytypes-datatypes/primarytypes-datatypes-etHumanKind.msr

```

1 package icrash.operations.concepts.primarytypes.datatypes.etHumanKind{
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 Operation Model {
6
7   operation: icrash.concepts.primarytypes.datatypes.etHumanKind.is():ptBoolean{
8     postF{
9       let TheResult: ptBoolean in
10      ( if
11        ( self = witness
12        or self = victim
13        or self = anonymous
14      )
15      then (TheResult = true)
16      else (TheResult = false)
17    endif
18    result = TheResult
19  )
20 prolog{"src/Operations/Concepts/PrimaryTypesClasses/PrimaryTypesDatatypes-etHumanKind-is.pl"}
21 }
22 }
23 }
```

Listing B.50: Messir Spec. file primarytypes-datatypes-etHumanKind.msr.

## B.51 File ./src-gen/messir-spec/concepts/primarytypes-datatatypes.msr

```

1 package icrash.concepts.primarytypes.datatypes {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.string
5 import lu.uni.lassy.messir.libraries.math
6 import lu.uni.lassy.messir.libraries.calendar
7
8 Concept Model {
9
10 Primary Types {
11
12     datatype dtAlertID {
13         operation is():ptBoolean
14     }
15     datatype dtCrisisID {
16         operation is():ptBoolean
17     }
18     datatype dtLogin {
19         operation is():ptBoolean
20     }
21     datatype dtPassword {
22         operation is():ptBoolean
23     }
24     datatype dtCoordinatorID {
25         operation is():ptBoolean
26     }
27     datatype dtPoliceID {
28         operation is():ptBoolean
29     }
30     datatype dtPhoneNumber {
31         operation is():ptBoolean
32     }
33     datatype dtComment {
34         operation is():ptBoolean
35     }
36     datatype dtLatitude {
37         operation is():ptBoolean
38     }
39     datatype dtLongitude {
40         operation is():ptBoolean
41     }
42     datatype dtGPSLocation {
43         attribute latitude: dtLatitude
44         attribute longitude: dtLongitude
45         operation is():ptBoolean
46         operation isNearTo(AGPSLocation:dtGPSLocation ):ptBoolean
47     }
48
49     enum etCrisisStatus {
50         constants["pending", "handled", "solved", "closed"]
51         operation is():ptBoolean
52     }
53     enum etAlertStatus {
54         constants["pending", "valid", "invalid"]
55         operation is():ptBoolean
56     }
57     enum etCrisisType {
58         constants["small", "medium", "huge"]
59         operation is():ptBoolean
60     }
61     enum etHumanKind {
62         constants["witness", "victim", "anonymous"]
63         operation is():ptBoolean
64     }
65
66 }

```

```
67 }
68 }
```

Listing B.51: Messir Spec. file primarytypes-datatypes.msr.

### B.52 File                    ./src-gen/messir-spec/concepts/secondarytypes- associations.msr

```
1 package icrash.concepts.secondarytypes.associations {
2
3 Concept Model {
4
5 Secondary Types{
6
7 }
8 }
9 }
```

Listing B.52: Messir Spec. file secondarytypes-associations.msr.

### B.53 File                    ./src-gen/messir-spec/concepts/secondarytypes- classes.msr

```
1 package icrash.concepts.secondarytypes.classes {
2
3 Concept Model {
4
5 Secondary Types{
6
7 }
8 }
9 }
```

Listing B.53: Messir Spec. file secondarytypes-classes.msr.

### B.54 File                    ./src-gen/messir-spec/concepts/secondarytypes- datatypes.msr

```
1 package icrash.concepts.secondarytypes.datatypes {
2
3 import lu.uni.lassy.messir.libraries.primitives
4 import lu.uni.lassy.messir.libraries.string
5
6 import icrash.concepts.primarytypes.datatypes
7
8 Concept Model {
9
10 Secondary Types {
11
12 datatype dtSMS {
13   attribute value: ptString
14   operation is():ptBoolean
15 }
16 }
17 }
18 }
```

Listing B.54: Messir Spec. file secondarytypes-datatypes.msr.

### B.55 File ./src-gen/messir-spec/usecases/subfunctions-usecases.msr

```

1 package icrash.usecases.subfunctions {
2
3 import lu.uni.lassy.messir.libraries.primitives
4
5 import icrash.concepts.primarytypes.datatypes
6 import icrash.concepts.primarytypes.classes
7 import icrash.concepts.secondarytypes.datatypes
8 import lu.uni.lassy.messir.libraries.primitives
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.calendar
11
12 import icrash.environment
13
14 Use Case Model {
15
16 //-----
17 use case system subfunction oeAddCoordinator(AdtCoordinatorID:dtCoordinatorID, AdtLogin:dtLogin,
18     AdtPassword:dtPassword){
19     actor actAdministrator[primary,active]
20     returned messages {
21         ieCoordinatorAdded() returned to actAdministrator
22     }
23 }
24 //-----
25 use case system subfunction oeAddPolice(AdtPoliceID:dtPoliceID, AdtLogin:dtLogin, AdtPassword:
26     dtPassword){
27     actor actAdministrator[primary,active]
28     returned messages {
29         iePoliceAdded() returned to actAdministrator
30     }
31 }
32 //-----
33 use case system subfunction oeAlert(
34     AetKind:etHumanKind,
35     AdtMyDate:dtDate,
36     AdtTime:dtTime,
37     AdtPhoneNumber:dtPhoneNumber,
38     AdtGPSLocation:dtGPSLocation,
39     AdtComment:dtComment) {
40     actor actComCompany[primary,active]
41     returned messages {
42         ieSmsSend(AdtPhoneNumber,AdtSMS) returned to actComCompany
43     }
44 }
45 //-----
46 use case system subfunction oeInvalidateAlert(AdtAlertID:dtAlertID) {
47     actor actCoordinator[primary,active]
48     actor actComCompany[secondary,passive]
49     returned messages {
50         ieMessage(AMessage) returned to actCoordinator
51     }
52 }
53 //-----
54
55 use case system subfunction oeCloseCrisis(AdtCrisisID:dtCrisisID) {
56     actor actCoordinator[primary,active]
57     // actor actPolice[secondary,active] ??????????????????????
58     returned messages {
59         ieMessage(AMessage) returned to actCoordinator
60     }
61 }
62
63 use case system subfunction oeCreateSystemAndEnvironment(AqtyComCompanies:ptInteger) {
64     actor actMsrCreator[primary,active]
65 }
66
67 use case system subfunction oeDeleteCoordinator(AdtCoordinatorID:dtCoordinatorID) {
68     actor actAdministrator[primary,active]

```

```

69  returned messages {
70    ieCoordinatorDeleted() returned to actAdministrator
71  }
72 }
73 //-----
74 use case system subfunction oeDeletePolice(AdtPoliceID:dtPoliceID) {
75   actor actAdministrator[primary,active]
76   returned messages {
77     iePoliceDeleted() returned to actAdministrator
78   }
79 }
80 //-----
81 use case system subfunction oeGetAlertsSet(AetAlertStatus:etAlertStatus) {
82   actor actCoordinator[primary,active]
83   returned messages {
84     ieSendAnAlert(ActAlert) returned to actCoordinator
85   }
86 }
87 //-----
88 use case system subfunction oeGetCrisisSet(AetCrisisStatus:etCrisisStatus){
89   actor actCoordinator[primary,active]
90   returned messages {
91     ieSendACrisis(ActCrisis) returned to actCoordinator
92   }
93 }
94
95 ///////////////////////////////////////////////////////////////////
96
97 //-----
98 use case system subfunction oeSetCrisisHandler(AdtCrisisID:dtCrisisID) {
99   actor actCoordinator[primary,active]
100  actor actCoordinator[secondary,passive]
101  actor actComCompany[secondary,passive,multiple]
102  returned messages {
103    ieMessage(AMessage)
104    returned to actCoordinator
105    ieSendAnAlert(ActAlert)
106    returned to actCoordinator
107    ieSmsSend(AdtPhoneNumber,AdtSMS)
108    returned to actComCompany
109  }
110 }
111 //-----
112 use case system subfunction oeLogin(AdtLogin:dtLogin , AdtPassword:dtPassword) {
113   actor actAuthenticated[primary,active]
114   returned messages {
115     ieMessage(AMessage) returned to actAuthenticated
116   }
117 }
118 //-----
119 use case system subfunction oeLogout() {
120   actor actAuthenticated[primary,active]
121   returned messages {
122     ieMessage(AMessage) returned to actAuthenticated
123   }
124 }
125 //-----
126 use case system subfunction oeReportOnCrisis(AdtCrisisID:dtCrisisID,AdtComment:dtComment) {
127   actor actCoordinator[primary,active]
128   returned messages {
129     ieMessage(AMessage) returned to actCoordinator
130   }
131 }
132 //-----
133 use case system subfunction oeSetClock(AcurrentClock:dtDateAndTime) {
134   actor actActivator[primary,proactive]
135 }
136 //-----
137 use case system subfunction oeSetCrisisStatus(AdtCrisisID:dtCrisisID ,AetCrisisStatus:
138   etCrisisStatus) {

```

```

138 actor actCoordinator[primary,active]
139 //actor actPolice [primary,active]
140 returned messages {
141   ieMessage(AMessage) returned to actCoordinator
142 }
143 }
144 //-----
145 //
146 use case system subfunction oeSetCrisisType(AdtCrisisID:dtCrisisID ,AetCrisisType:etCrisisType) {
147   actor actCoordinator[primary,active]
148   returned messages {
149     ieMessage(AMessage) returned to actCoordinator
150   }
151 }
152 //-----
153 use case system subfunction oeSollicitateCrisisHandling() {
154   actor actActivator[primary,proactive]
155   actor actCoordinator[secondary,passive,multiple]
156   actor actPolice[secondary,passive,multiple]
157   actor actAdministrator[secondary,passive]
158   returned messages {
159     ieMessage(AMessage) returned to actCoordinator
160   }
161   //ieMessage(AMessage) returned to actPolice
162   //ieMessage(AMessage) returned to actAdministrator
163 }
164 }
165 }
166 }
167 //-----
168 use case system subfunction oeValidateAlert(AdtAlertID:dtAlertID) {
169   actor actCoordinator[primary,active]
170   returned messages {
171     ieMessage(AMessage) returned to actCoordinator
172   }
173 }
174 }
175 }
176 }

```

Listing B.55: Messir Spec. file subfunctions-usecases.msr.

## B.56 File ./src-gen/messir-spec/test/tc-testcase01.msr

```

1 package lu.uni.lassy.excalibur.examples.icrash.tests.testcase01 {
2
3 import lu.uni.lassy.messir.libraries.string
4 import lu.uni.lassy.messir.libraries.primitives
5 import lu.uni.lassy.messir.libraries.math
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.associations
9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.primarytypes.datatypes
11 import icrash.concepts.secondarytypes.datatypes
12 import icrash.environment
13
14 Test Model{
15   test case testcase01 order 01 {
16   //-----
17   test step ts01oeCreateSystemAndEnvironment order 01 {
18     variables{
19       Creator:actMsrCreator
20       AqtyComCompanies: ptInteger
21     }
22     constraints{
23       AqtyComCompanies = 4
24     }
25     test message{

```

```

26      out:Creator sends to system actMsrCreator.outactMsrCreator.oeCreateSystemAndEnvironment(
27          AqtyComCompanies)
28    }
29  oracle{
30    constraints{
31      true
32    }
33    prolog{"src/Tests/system/01/system-sim-01-01-oeCreateSystemAndEnvironment.pl"}
34  }
35 //-----
36 test step ts02oeSetClock order 02{
37   variables{
38     TheActor:actActivator
39     ACurrentClock:dtDateAndTime
40   }
41   constraints{
42     TheActor=TheSystem.rnactActivator->any2(true)
43
44     ACurrentClock.date.year.value = 2017
45     ACurrentClock.date.month.value = 11
46     ACurrentClock.date.day.value = 24
47     ACurrentClock.time.hour.value = 15
48     ACurrentClock.time.minute.value = 20
49     ACurrentClock.time.second.value = 00
50   }
51   test message{
52     out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
53   }
54  oracle{
55    constraints{
56      true
57    }
58  }
59 }
60 //-----
61
62 test step ts03oeLogin order 03{
63   variables{
64     TheActor : actAdministrator
65     AdtLogin:dtLogin
66     AdtPassword:dtPassword
67   }
68   constraints{
69     TheActor=TheSystem.rnactAdministrator->any2(true)
70     AdtLogin.value.eq('icrashadmin')
71     AdtPassword.value.eq('7WXC1359')
72   }
73   test message{
74     out:TheActor sends to system actAdministrator.outactAdministrator.oeLogin(AdtLogin,AdtPassword)
75   }
76  oracle{
77   variables{
78     AMessage:ptString
79   }
80   constraints{
81     AMessage = 'You are logged ! Welcome ...'
82     TheActor.inactAdministrator.ieMessage(AMessage)
83   }
84 }
85 }
86 //-----
87 test step ts04oeAddCoordinator order 04{
88   variables{
89     TheActor : actAdministrator
90     AdtCoordinatorID : dtCoordinatorID
91     AdtLogin:dtLogin
92     AdtPassword:dtPassword
93   }
94   constraints{

```

```

95     TheActor = TheSystem.rnactAdministrator->any2(true)
96     AdtCoordinatorID.value.eq('1')
97     AdtLogin.value.eq('steve')
98     AdtPassword.value.eq('pwdMessirExcalibur2017')
99   }
100  test message{
101    out:TheActor
102    sends to system actAdministrator.outactAdministrator.oeAddCoordinator
103      (AdtCoordinatorID,
104       AdtLogin,
105       AdtPassword)
106  }
107 oracle{
108   constraints{
109     TheActor.inactAdministrator.ieCoordinatorAdded()
110   }
111 }
112 }
113 //-----
114 test step ts05oeLogout order 05{
115   variables{
116     TheActor : actAdministrator
117   }
118   constraints{
119     TheActor = TheSystem.rnactAdministrator->any2(true)
120   }
121   test message{
122     out:TheActor sends to system actAdministrator.outactAdministrator.oeLogout()
123   }
124 oracle{
125   variables{
126     AMessage:ptString
127   }
128   constraints{
129     AMessage = 'You are logged out ! Good Bye ...'
130     TheActor.inactAdministrator.ieMessage(AMessage)
131   }
132 }
133 }
134 //-----
135 test step ts06oeSetClock02 order 06{
136   variables{
137     TheActor:actActivator
138     ACurrentClock:dtDateAndTime
139   }
140   constraints{
141     TheActor=TheSystem.rnactActivator->any2(true)
142     ACurrentClock.date.year.value = 2017
143     ACurrentClock.date.month.value = 11
144     ACurrentClock.date.day.value = 26
145     ACurrentClock.time.hour.value = 10
146     ACurrentClock.time.minute.value = 15
147     ACurrentClock.time.second.value = 00
148   }
149   test message{
150     out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
151   }
152 oracle{
153   constraints{
154     true
155   }
156 }
157 }
158 //-----
159 test step ts07oeAlert1 order 07{
160   variables{
161     TheActor : actComCompany
162     AetHumanKind:etHumanKind
163     AdtDate:dtDate
164     AdtTime:dtTime

```

```

165     AdtPhoneNumber:dtPhoneNumber
166     AdtGPSLocation:dtGPSLocation
167     AdtComment:dtComment
168 }
169 constraints{
170     TheActor = TheSystem.rnactComCompany->any2(true)
171     AetHumanKind = witness
172     AdtDate.year.value = 2017
173     AdtDate.month.value = 11
174     AdtDate.day.value = 26
175     AdtTime.hour.value = 10
176     AdtTime.minute.value = 10
177     AdtTime.second.value = 16
178     AdtPhoneNumber.value = '+3524666445252'
179     AdtGPSLocation.latitude.value = 49.627675
180     AdtGPSLocation.longitude.value = 6.159590
181     AdtComment.value = '3 cars involved in an accident.'
182 }
183 test message{
184     out:TheActor
185     sends to system actComCompany.outactComCompany.oeAlert( AetHumanKind,
186                                         AdtDate,
187                                         AdtTime,
188                                         AdtPhoneNumber,
189                                         AdtGPSLocation,
190                                         AdtComment)
191 }
192 oracle{
193     variables{
194         AdtSMS:dtSMS
195     }
196     constraints{
197         AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
198         TheActor.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
199     }
200 }
201 }
202 //-----
203 test step ts08oeSetClock03 order 08{
204     variables{
205         TheActor:actActivator
206         ACurrentClock:dtDateAndTime
207     }
208     constraints{
209         TheActor=TheSystem.rnactActivator->any2(true)
210         ACurrentClock.date.year.value = 2017
211         ACurrentClock.date.month.value = 11
212         ACurrentClock.date.day.value = 26
213         ACurrentClock.time.hour.value = 10
214         ACurrentClock.time.minute.value = 30
215         ACurrentClock.time.second.value = 00
216     }
217     test message{
218         out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
219     }
220     oracle{
221         constraints{
222             true
223         }
224     }
225 }
226 //-
227 test step ts09oeSollicitateCrisisHandling order 09{
228     variables{
229         TheActor : actActivator
230     }
231     constraints{
232         TheActor = TheSystem.rnactActivator->any2(true)
233     }
234     test message{

```

```

235     out:TheActor sends to system actActivator.outactActivator.oeSollicitateCrisisHandling()
236 }
237 oracle{
238 variables{
239     TheAdministrator:actAdministrator
240     TheCoordinator:actCoordinator
241     AMessageForCrisisHandlers:ptString
242 }
243 constraints{
244     TheAdministrator = TheSystem.rnactAdministrator->any2(true)
245     TheCoordinator = TheSystem.rnactCoordinator->any2(true)
246     AMessageForCrisisHandlers = 'There are alerts pending since more than the defined delay. Please
REACT !'
247
248     TheAdministrator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
249     TheCoordinator.inactAdministrator.ieMessage(AMessageForCrisisHandlers)
250
251 /* this oracle should be written like this:
252
253     oracle{
254     variables{
255         TheAdministrator:actAdministrator
256         AMessageForCrisisHandlers:ptString
257     }
258     constraints{
259         AMessageForCrisisHandlers = 'There are alerts pending since more than the defined delay. Please
REACT !'
260         TheAdministrator = TheSystem.rnactAdministrator->any2(true)
261
262         TheSystem.rnactCoordinator->forAll(TheCoordinator:actCoordinator | TheCoordinator.
actAuthenticated.inactAuthenticated.ieMessage(AMessage))
263
264     // receives from system is for step instances
265
266 */
267 }
268 }
269 }
270 /-----
271 test step ts10oeLogin02 order 10{
272 variables{
273     TheActor : actCoordinator
274     AdtLogin:dtLogin
275     AdtPassword:dtPassword
276 }
277 constraints{
278     TheActor = TheSystem.rnactCoordinator->select(a | a.rnctCoordinator.login.value.eq('steve'))->
any2(true)
279     AdtLogin.value.eq('steve')
280     AdtPassword.value.eq('pwdMessirExcalibur2017')
281 }
282 test message{
283     out:TheActor sends to system actAuthenticated.outactAuthenticated.oeLogin(AdtLogin,AdtPassword)
284 }
285 oracle{
286 variables{
287     AMessage:ptString
288 }
289 constraints{
290     AMessage = 'You are logged ! Welcome ...'
291     TheActor.inactAuthenticated.ieMessage(AMessage)
292 }
293 }
294 }
295 /-----
296 test step ts11oeGetCrisisSet order 11{
297 variables{
298     TheActor : actCoordinator
299     AetCrisisStatus : etCrisisStatus
300 }

```

```

301   constraints{
302     TheActor=TheSystem.rnactCoordinator
303     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
304     ->any2(true)
305     AetCrisisStatus = pending
306   }
307   test message{
308     out:TheActor sends to system actCoordinator.outactCoordinator.oeGetCrisisSet(AetCrisisStatus)
309   }
310   oracle{
311 //TODO - make consistent with test step implementation by adding Prolog code for input messages
312   variables{
313     ActCrisis:ctCrisis
314   }
315   constraints{
316     TheActor.inactCoordinator.ieSendACrisis(ActCrisis)
317   }
318 }
319 }

320 //-----
321 test step ts12oeSetCrisisHandler order 12{
322   variables{
323     TheActor : actCoordinator
324     AdtCrisisID : dtCrisisID
325   }
326   constraints{
327     TheActor=TheSystem.rnactCoordinator
328     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
329     ->any2(true)
330     //and AdtCrisisID.value= '1'
331   }
332   test message{
333     out:TheActor sends to system actCoordinator.outactCoordinator.oeSetCrisisHandler(AdtCrisisID)
334   }
335   oracle{
336   variables{
337     AMesssage:ptString
338     AdtPhoneNumber:dtPhoneNumber
339     AdtSMS:dtSMS
340     ActAlert:ctAlert
341
342     TheComCompany: actComCompany
343     TheCoordinator:actCoordinator
344   }
345   constraints{
346     AMesssage = 'You are now considered as handling the crisis !'
347     AdtSMS.value = 'The handling of your alert by our services is in progress !'
348     TheComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
349     TheCoordinator.inactCoordinator.ieSendAnAlert(ActAlert)
350     TheActor.inactAuthenticated.ieMessage(AMesssage)
351   }
352 }
353 }

354 //-----
355 test step ts13oeSetClock04 order 13{
356   variables{
357     TheActor:actActivator
358     ACurrentClock:dtDateAndTime
359   }
360   constraints{
361     TheActor=TheSystem.rnactActivator->any2(true)
362     ACurrentClock.date.year.value = 2017
363     ACurrentClock.date.month.value = 11
364     ACurrentClock.date.day.value = 26
365     ACurrentClock.time.hour.value = 10
366     ACurrentClock.time.minute.value = 45
367     ACurrentClock.time.second.value = 00
368   }
369   test message{
370     out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)

```

```

371     }
372     oracle{
373       constraints{
374         true
375       }
376     }
377   }
378 /**
379 test step ts14oeValidateAlert order 14{
380   variables{
381     TheActor : actCoordinator
382     AdtAlertID : dtAlertID
383   }
384   constraints{
385     TheActor=TheSystem.rnactCoordinator
386     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
387     ->any2(true)
388     //and AdtAlertID.value= '1'
389   }
390   test message{
391     out:TheActor sends to system actCoordinator.outactCoordinator.oeValidateAlert(AdtAlertID)
392   }
393   oracle{
394     variables{
395       AMessage:ptString
396     }
397     constraints{
398       AMessage = 'The Alert is now declared as valid !'
399       TheActor.actAuthenticated.inactAuthenticated.ieMessage(AMessage)
400     }
401   }
402 }
403 /**
404 test step ts15oeAlert2 order 15{
405   variables{
406     TheActor : actComCompany
407     AetHumanKind:etHumanKind
408     AdtDate:dtDate
409     AdtTime:dtTime
410     AdtPhoneNumber:dtPhoneNumber
411     AdtGPSLocation:dtGPSLocation
412     AdtComment:dtComment
413   }
414   constraints{
415     TheActor = TheSystem.rnactComCompany->any2(true)
416     AetHumanKind = witness
417     AdtDate.year.value = 2017
418     AdtDate.month.value = 11
419     AdtDate.day.value = 26
420     AdtTime.hour.value = 10
421     AdtTime.minute.value = 20
422     AdtTime.second.value = 00
423     AdtPhoneNumber.value = '+3524666445000'
424     AdtGPSLocation.latitude.value = 49.627095
425     AdtGPSLocation.longitude.value = 6.160251
426     AdtComment.value = 'A car crash just happened.'
427   }
428   test message{
429     out:TheActor
430     sends to system actComCompany.outactComCompany.oeAlert( AetHumanKind,
431                               AdtDate,
432                               AdtTime,
433                               AdtPhoneNumber,
434                               AdtGPSLocation,
435                               AdtComment)
436   }
437   oracle{
438     variables{
439       AdtSMS:dtSMS
440     }

```

```

441     constraints{
442         AdtSMS.value = 'Your alert has been registered. We will handle it and keep you informed'
443         TheActor.actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
444     }
445 }
446 }
447 //-----
448 test step ts16oeSetClock05 order 16{
449     variables{
450         TheActor:actActivator
451         ACurrentClock:dtDateAndTime
452     }
453     constraints{
454         TheActor=TheSystem.rnactActivator->any2(true)
455         ACurrentClock.date.year.value = 2017
456         ACurrentClock.date.month.value = 11
457         ACurrentClock.date.day.value = 26
458         ACurrentClock.time.hour.value = 12
459         ACurrentClock.time.minute.value = 45
460         ACurrentClock.time.second.value = 00
461     }
462     test message{
463         out:TheActor sends to system actActivator.outactActivator.oeSetClock(ACurrentClock)
464     }
465     oracle{
466         constraints{
467             true
468         }
469     }
470 }
471 //-----
472 test step ts17oeSetCrisisStatus order 17{
473     variables{
474         TheActor : actCoordinator
475         AdtCrisisID : dtCrisisID
476         AetCrisisStatus : etCrisisStatus
477     }
478     constraints{
479         TheActor=TheSystem.rnactCoordinator
480         ->select(a | a.rnctCoordinator.login.value.eq('steve'))
481         ->any2(true)
482         //and AdtCrisisID.value= '1'
483         //and AetCrisisStatus = solved
484     }
485     test message{
486         out:TheActor sends to system actCoordinator.outactCoordinator.oeSetCrisisStatus(AdtCrisisID,
487         AetCrisisStatus)
488     }
489     oracle{
490         variables{
491             AMesssage:ptString
492         }
493         constraints{
494             AMesssage = 'The crisis status has been updated !'
495             TheActor.inactAuthenticated.ieMessage(AMesssage)
496         }
497     }
498 //-----
499 test step ts18oeReportOnCrisis order 18{
500     variables{
501         TheActor : actCoordinator
502         AdtCrisisID : dtCrisisID
503         AdtComment : dtComment
504     }
505     constraints{
506         TheActor=TheSystem.rnactCoordinator
507         ->select(a | a.rnctCoordinator.login.value.eq('steve'))
508         ->any2(true)
509         //and AdtCrisisID.value= '1'

```

```

510      //and AdtComment.value = '3 victims sent to hospital, 2 cars evacuated and 4 rescue unit
mobilized'
511    }
512  test message{
513    out:TheActor sends to system actCoordinator.outactCoordinator.oeReportOnCrisis(AdtCrisisID,
AdtComment)
514  }
515  oracle{
516    variables{
517      AMessage:ptString
518    }
519    constraints{
520      AMessage = 'The crisis comment has been updated !'
521      TheActor.inactAuthenticated.ieMessage(AMessage)
522    }
523  }
524 }
525 /-----
526 test step ts19oeCloseCrisis order 19{
527   variables{
528     TheActor : actCoordinator
529     AdtCrisisID : dtCrisisID
530   }
531   constraints{
532     TheActor=TheSystem.rnactCoordinator
533     ->select(a | a.rnctCoordinator.login.value.eq('steve'))
534     ->any2(true)
535     //and AdtCrisisID.value= '1'
536   }
537   test message{
538     out:TheActor sends to system actCoordinator.outactCoordinator.oeCloseCrisis(AdtCrisisID)
539   }
540   oracle{
541     variables {
542       AMessage:ptString
543     }
544     constraints{
545       AMessage = 'The crisis is now closed !'
546       TheActor.inactAuthenticated.ieMessage(AMessage)
547     }
548   }
549 }
550 }
551 }
552 }
```

Listing B.56: Messir Spec. file tc-testcase01.msr.

**B.57 File ./src-gen/messir-spec/test/tci-testcase01-instance01.msr**

```

1 package lu.uni.lassy.excalibur.examples.icrash.tests.testcase01.instance01 {
2
3 import lu.uni.lassy.messir.libraries.string
4 import lu.uni.lassy.messir.libraries.primitives
5 import lu.uni.lassy.messir.libraries.math
6 import lu.uni.lassy.messir.libraries.calendar
7
8 import icrash.concepts.primarytypes.associations
9 import icrash.concepts.primarytypes.classes
10 import icrash.concepts.primarytypes.datatypes
11 import lu.uni.lassy.excalibur.examples.icrash.tests.testcase01
12 import icrash.environment
13
14 Test Model {
15   test case instance instance01:testcase01{
16 /-----
17   test step instance tsi01:testcase01.ts01oeCreateSystemAndEnvironment{
18     variables {
19       theCreator:testcase01.ts01oeCreateSystemAndEnvironment.Creator = "theCreator"
```

```

20     AqtyComCompanies : testcase01.ts01oeCreateSystemAndEnvironment.AqtyComCompanies="4"
21   }
22   oracle {
23     satisfaction = "true"
24   }
25   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
26 }
27 //-----
28 test step instance tsi02: testcase01.ts02oeSetClock{
29   variables {
30     theClock:testcase01.ts02oeSetClock.TheActor = "theClock"
31     ACurrentClock : testcase01.ts02oeSetClock.ACurrentClock= "2017:11:24 - 03:20:00"
32   }
33   oracle {
34     satisfaction = "true"
35   }
36   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
37 }
38 //-----
39 test step instance tsi03: testcase01.ts03oeLogin{
40   variables {
41     bill:testcase01.ts03oeLogin.TheActor="bill"
42     AdtLogin : testcase01.ts03oeLogin.AdtLogin= "icrashadmin"
43     AdtPassword : testcase01.ts03oeLogin.AdtPassword= "7WXC1359"
44   }
45   oracle {
46     satisfaction = "true"
47     received message {
48       AMessag : testcase01.ts03oeLogin.AMessage= 'You are logged ! Welcome ...'
49       tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
50     }
51   }
52   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
53 }
54 //-----
55 test step instance tsi04: testcase01.ts04oeAddCoordinator{
56   variables {
57     reuse tsi03.bill as testcase01.ts04oeAddCoordinator.TheActor
58     AdtCoordinatorID : testcase01.ts04oeAddCoordinator.AdtCoordinatorID = "1"
59     AdtLogin : testcase01.ts04oeAddCoordinator.AdtLogin= "steve"
60     AdtPassword : testcase01.ts04oeAddCoordinator.AdtPassword = "pwdMessirExcalibur2017"
61   }
62   oracle {
63     satisfaction = "true"
64     received message {
65       tsi03.bill received from system actAdministrator.inactAdministrator.ieCoordinatorAdded()
66     }
67   }
68   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
69 }
70 //-----
71 test step instance tsi05: testcase01.ts05oeLogout{
72   variables {
73     reuse tsi03.bill as testcase01.ts05oeLogout.TheActor
74   }
75   oracle {
76     satisfaction = "true"
77     received message {
78       AMessag : testcase01.ts05oeLogout.AMessage= 'You are logged out ! Good Bye ...'
79       tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
80     }
81   }
82   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
83 }
84 //-----
85 test step instance tsi06: testcase01.ts06oeSetClock02{
86   variables {
87     reuse tsi02.theClock as testcase01.ts06oeSetClock02.TheActor
88     ACurrentClock : testcase01.ts06oeSetClock02.ACurrentClock= "2017:11:26 - 10:15:00"
89   }

```

```

90  oracle {
91      satisfaction = "true"
92  }
93  test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
94  }
95 //-----
96 test step instance tsi07: testcase01.ts07oeAlert1{
97  variables {
98      tango:testcase01.ts07oeAlert1.TheActor ="tango"
99      AetHumanKind : testcase01.ts07oeAlert1.AetHumanKind = "witness"
100     AdtDate : testcase01.ts07oeAlert1.AdtDate = "2017:11:26"
101     AdtTime : testcase01.ts07oeAlert1.AdtTime = "10:10:16"
102     AdtPhoneNumber : testcase01.ts07oeAlert1.AdtPhoneNumber = "+3524666445252"
103     AdtGPSLocation : testcase01.ts07oeAlert1.AdtGPSLocation = "49.627675:6.159590"
104     AdtComment : testcase01.ts07oeAlert1.AdtComment = "3 cars involved in an accident."
105  }
106 oracle {
107     satisfaction = "true"
108     received message {
109         AdtSMS : testcase01.ts07oeAlert1.AdtSMS= 'Your alert has been registered. We will handle it and
keep you informed'
110         tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
111     }
112  }
113  }
114  test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
115  }
116
117 //-----
118 test step instance tsi08: testcase01.ts08oeSetClock03{
119  variables {
120      reuse tsi02.theClock as testcase01.ts08oeSetClock03.ACurrrentClock
121      ACurrentClock : testcase01.ts08oeSetClock03.ACurrrentClock = "2017:11:26 - 10:30:00"
122  }
123 oracle {
124     satisfaction = "true"
125  }
126  test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
127  }
128 //-----
129 test step instance tsi09: testcase01.ts09oeSollicitateCrisisHandling{
130  variables {
131      reuse tsi02.theClock as testcase01.ts09oeSollicitateCrisisHandling.TheActor
132      steve:testcase01.ts09oeSollicitateCrisisHandling.TheCoordinator ="steve"
133      reuse tsi03.bill as testcase01.ts09oeSollicitateCrisisHandling.TheAdministrator
134  }
135 oracle {
136     satisfaction = "true"
137     received message {
138         AMesssageForCrisisHandlers : testcase01.ts09oeSollicitateCrisisHandling.
AMesssageForCrisisHandlers= 'There are alerts pending since more than the defined delay. Please
REACT !'
139
140         tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(
AMesssageForCrisisHandlers)
141         tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(
AMesssageForCrisisHandlers)
142     }
143  }
144  test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
145  }
146
147 //-----
148 test step instance tsi10: testcase01.ts10oeLogin02{
149  variables {
150      reuse tsi09.steve as testcase01.ts10oeLogin02.TheActor
151      AdtLogin : testcase01.ts10oeLogin02.AdtLogin = "steve"
152      AdtPassword : testcase01.ts10oeLogin02.AdtPassword= "pwdMessirExcalibur2017"
153  }
154 oracle {

```

```

155     satisfaction = "true"
156     received message {
157       AMessage : testcase01.ts10oeLogin02.AMessage= 'You are logged ! Welcome ...'
158       tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
159     }
160   }
161 }
162 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
163 }
164 //-----
165 test step instance ts11: testcase01.ts11oeGetCrisisSet{
166   variables {
167     reuse tsi09.steve as testcase01.ts11oeGetCrisisSet.TheActor
168     AetCrisisStatus : testcase01.ts11oeGetCrisisSet.AetCrisisStatus = "pending"
169   }
170   oracle {
171     satisfaction = "true"
172     received message {
173       ActCrisis : testcase01.ts11oeGetCrisisSet.ActCrisis= "crisis with ID 1 details"
174       tsi09.steve received from system actCoordinator.inactCoordinator.ieSendACrisis(ActCrisis)
175     }
176   }
177   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
178 }
179 //-----
180 test step instance ts12: testcase01.ts12oeSetCrisisHandler{
181   variables {
182     reuse tsi09.steve as testcase01.ts12oeSetCrisisHandler.TheActor
183     AdtCrisisID : testcase01.ts12oeSetCrisisHandler.AdtCrisisID = "1"
184
185     reuse tsi07.tango as testcase01.ts12oeSetCrisisHandler.TheComCompany
186   }
187   oracle {
188     satisfaction = "true"
189     received message {
190       AMessage : testcase01.ts12oeSetCrisisHandler.AMessage= 'You are now considered as handling the
191       crisis !'
192       AdtSMS : testcase01.ts12oeSetCrisisHandler.AdtSMS= 'The handling of your alert by our services
193         is in progress !'
194       AdtPhoneNumber : testcase01.ts12oeSetCrisisHandler.AdtPhoneNumber= "+3524666445252"
195
196       tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
197       tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
198     }
199   }
200   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
201 }
202 //-----
203 test step instance ts13: testcase01.ts13oeSetClock04{
204   variables {
205     reuse tsi02.theClock as testcase01.ts13oeSetClock04.TheActor
206     ACurrentClock : testcase01.ts13oeSetClock04.ACurrentClock = "2017:11:26 - 10:45:00"
207   }
208   oracle {
209     satisfaction = "true"
210   }
211   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
212 }
213 //-----
214 test step instance ts14: testcase01.ts14oeValidateAlert{
215   variables {
216     reuse tsi09.steve as testcase01.ts14oeValidateAlert.TheActor
217     AdtAlertID : testcase01.ts14oeValidateAlert.AdtAlertID = "1"
218   }
219   oracle {
220     satisfaction = "true"
221     received message {
222       AMessage : testcase01.ts14oeValidateAlert.AMessage= 'The Alert is now declared as valid !'
```

```

223     tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
224
225   }
226 }
227 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
228 }
229 //-----
230 test step instance tsi15: testcase01.ts15oeAlert2{
231   variables {
232     reuse tsi07.tango as testcase01.ts15oeAlert2.TheActor
233     AetHumanKind : testcase01.ts15oeAlert2.AetHumanKind ="witness"
234     AdtDate : testcase01.ts15oeAlert2.AdtDate= "2017:11:26"
235     AdtTime : testcase01.ts15oeAlert2.AdtTime= "10:20:00"
236     AdtPhoneNumber : testcase01.ts15oeAlert2.AdtPhoneNumber= "+3524666445000"
237     AdtGPSLocation : testcase01.ts15oeAlert2.AdtGPSLocation= "49.627095:6.160251"
238     AdtComment : testcase01.ts15oeAlert2.AdtComment= "A car crash just happened."
239   }
240   message {
241     tsi07.tango sent to system testcase01.ts15oeAlert2.out : actComCompany.outactComCompany.oeAlert(
242       AetHumanKind,AdtDate,AdtTime,AdtPhoneNumber,AdtGPSLocation,AdtComment)
243   }
244   oracle {
245     satisfaction = "true"
246     received message {
247       AdtSMS : testcase01.ts15oeAlert2.AdtSMS= 'Your alert has been registered. We will handle it and
248         keep you informed'
249       tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
250     }
251   }
252   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
253 }
254 //-----
255 test step instance tsi16: testcase01.ts16oeSetClock05{
256   variables {
257     reuse tsi02.theClock as testcase01.ts16oeSetClock05.TheActor
258     ACurrentClock : testcase01.ts16oeSetClock05.ACurrentClock = "2017:11:26 - 12:45:00"
259   }
260   oracle {
261     satisfaction = "true"
262     received message {
263
264   }
265   }
266   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
267 }
268 //-----
269 test step instance tsi17: testcase01.ts17oeSetCrisisStatus{
270   variables {
271     reuse tsi09.steve as testcase01.ts17oeSetCrisisStatus.TheActor
272     AdtCrisisID : testcase01.ts17oeSetCrisisStatus.AdtCrisisID = "1"
273     AetCrisisStatus : testcase01.ts17oeSetCrisisStatus.AetCrisisStatus= "solved"
274   }
275   oracle {
276     satisfaction = "true"
277     received message {
278       AMessage : testcase01.ts17oeSetCrisisStatus.AMessage= "The crisis status has been updated !"
279       tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
280     }
281   }
282   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
283 }
284 //-----
285 test step instance tsi18: testcase01.ts18oeReportOnCrisis{
286   variables {
287     reuse tsi09.steve as testcase01.ts18oeReportOnCrisis.TheActor
288     AdtCrisisID : testcase01.ts18oeReportOnCrisis.AdtCrisisID = "1"
289     AdtComment : testcase01.ts18oeReportOnCrisis.AdtComment= "3 victims sent to hospital, 2 cars
      evacuated and 4 rescue unit mobilized"

```

```

290     }
291   oracle {
292     satisfaction = "true"
293     received message {
294       AMessage : testcase01.ts18oeReportOnCrisis.AMessage= 'The crisis comment has been updated !'
295       tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
296     }
297   }
298 }
299 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
300 }
301 //-----
302 test step instance tsi19: testcase01.ts19oeCloseCrisis{
303 variables {
304   reuse tsi09.steve as testcase01.ts19oeCloseCrisis.TheActor
305   AdtCrisisID : testcase01.ts19oeCloseCrisis.AdtCrisisID = "1"
306 }
307 oracle {
308   satisfaction = "true"
309   received message {
310     AMessage : testcase01.ts19oeCloseCrisis.AMessage= 'The crisis is now closed !'
311   }
312   tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
313 }
314 }
315 }
316 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
317 }
318 }
319 }
320 //-----
321 //-----
322 //-----
323 test case instance instance01Part01:testcase01{
324 //-----
325 test step instance tsi01: testcase01.ts01oeCreateSystemAndEnvironment{
326 variables {
327   theCreator: testcase01.ts01oeCreateSystemAndEnvironment.Creator = "theCreator"
328   AqtyComCompanies : testcase01.ts01oeCreateSystemAndEnvironment.AqtyComCompanies="4"
329 }
330 oracle {
331   satisfaction = "true"
332 }
333 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
334 }
335 //-----
336 test step instance tsi02: testcase01.ts02oeSetClock{
337 variables {
338   theClock: testcase01.ts02oeSetClock.TheActor = "theClock"
339   ACurrentClock : testcase01.ts02oeSetClock.ACurrentClock= "2017:11:24 - 03:20:00"
340 }
341 oracle {
342   satisfaction = "true"
343 }
344 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
345 }
346 //-----
347 test step instance tsi03: testcase01.ts03oeLogin{
348 variables {
349   bill: testcase01.ts03oeLogin.TheActor="bill"
350   AdtLogin : testcase01.ts03oeLogin.AdtLogin= "icrashadmin"
351   AdtPassword : testcase01.ts03oeLogin.AdtPassword= "7WXC1359"
352 }
353 oracle {
354   satisfaction = "true"
355   received message {
356     AMessage : testcase01.ts03oeLogin.AMessage= 'You are logged ! Welcome ...'
357     tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
358   }
359 }

```

```

360     test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
361   }
362 //-----
363 test step instance tsi04: testcase01.ts04oeAddCoordinator{
364   variables {
365     reuse tsi03.bill as testcase01.ts04oeAddCoordinator.TheActor
366     AdtCoordinatorID : testcase01.ts04oeAddCoordinator.AdtCoordinatorID = "1"
367     AdtLogin : testcase01.ts04oeAddCoordinator.AdtLogin= "steve"
368     AdtPassword : testcase01.ts04oeAddCoordinator.AdtPassword = "pwdMessirExcalibur2017"
369   }
370   oracle {
371     satisfaction = "true"
372     received message {
373       tsi03.bill received from system actAdministrator.inactAdministrator.ieCoordinatorAdded()
374     }
375   }
376   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
377 }
378 //-----
379 test step instance tsi05: testcase01.ts05oeLogout{
380   variables {
381     reuse tsi03.bill as testcase01.ts05oeLogout.TheActor
382   }
383   oracle {
384     satisfaction = "true"
385     received message {
386       AMesssage : testcase01.ts05oeLogout.AMessage= 'You are logged out ! Good Bye ...'
387       tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
388     }
389   }
390   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
391 }
392 //-----
393 test step instance tsi06: testcase01.ts06oeSetClock02{
394   variables {
395     reuse tsi02.theClock as testcase01.ts06oeSetClock02.TheActor
396     ACurrentClock : testcase01.ts06oeSetClock02.ACurrentClock= "2017:11:26 - 10:15:00"
397   }
398   oracle {
399     satisfaction = "true"
400   }
401   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
402 }
403 //-----
404 test step instance tsi07: testcase01.ts07oeAlert1{
405   variables {
406     tango:testcase01.ts07oeAlert1.TheActor ="tango"
407     AetHumanKind : testcase01.ts07oeAlert1.AetHumanKind = "witness"
408     AdtDate : testcase01.ts07oeAlert1.AdtDate = "2017:11:26"
409     AdtTime : testcase01.ts07oeAlert1.AdtTime = "10:10:16"
410     AdtPhoneNumber : testcase01.ts07oeAlert1.AdtPhoneNumber = "+3524666445252"
411     AdtGPSLocation : testcase01.ts07oeAlert1.AdtGPSLocation = "49.627675:6.159590"
412     AdtComment : testcase01.ts07oeAlert1.AdtComment = "3 cars involved in an accident."
413   }
414   oracle {
415     satisfaction = "true"
416     received message {
417       AdtSMS : testcase01.ts07oeAlert1.AdtSMS= 'Your alert has been registered. We will handle it and keep you informed'
418       tsi07.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
419     }
420   }
421 }
422 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
423 }
424
425 //-----
426 test step instance tsi08: testcase01.ts08oeSetClock03{
427   variables {
428     reuse tsi02.theClock as testcase01.ts08oeSetClock03.ACurrentClock

```

```

429     ACurrentClock : testcase01.ts08oeSetClock03.ACurrentClock = "2017:11:26 - 10:30:00"
430   }
431   oracle {
432     satisfaction = "true"
433   }
434   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
435 }
436 //-----
437 test step instance tsi09: testcase01.ts09oeSollicitateCrisisHandling{
438   variables {
439     reuse tsi02.theClock as testcase01.ts09oeSollicitateCrisisHandling.TheActor
440     steve:testcase01.ts09oeSollicitateCrisisHandling.TheCoordinator ="steve"
441     reuse tsi03.bill as testcase01.ts09oeSollicitateCrisisHandling.TheAdministrator
442   }
443   oracle {
444     satisfaction = "true"
445     received message {
446       AMESSAGEForCrisisHandlers : testcase01.ts09oeSollicitateCrisisHandling.
447       AMESSAGEForCrisisHandlers= 'There are alerts pending since more than the defined delay. Please
448       REACT !'
449       tsi03.bill received from system actAuthenticated.inactAuthenticated.ieMessage(
450         AMESSAGEForCrisisHandlers)
451       tsi09.steve received from system actAuthenticated.inactAuthenticated.ieMessage(
452         AMESSAGEForCrisisHandlers)
453       }
454     }
455   }
456 //-----
457 //-----
458 //-----
459 test case instance instance01Part02: testcase01{
460
461   test step instance tsi10: testcase01.ts10oeLogin02{
462     variables {
463       steve : testcase01.ts10oeLogin02.TheActor
464       AdtLogin : testcase01.ts10oeLogin02.AdtLogin = "steve"
465       AdtPassword : testcase01.ts10oeLogin02.AdtPassword= "pwdMessirExcalibur2017"
466     }
467     oracle {
468       satisfaction = "true"
469       received message {
470         AMESSAGE : testcase01.ts10oeLogin02.AMESSAGE= 'You are logged ! Welcome ...'
471         steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMESSAGE)
472       }
473     }
474   }
475   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
476 }
477 //-----
478 test step instance ts11: testcase01.ts11oeGetCrisisSet{
479   variables {
480     reuse tsi10.steve as testcase01.ts11oeGetCrisisSet.TheActor
481     AetCrisisStatus : testcase01.ts11oeGetCrisisSet.AetCrisisStatus = "pending"
482   }
483   oracle {
484     satisfaction = "true"
485     received message {
486       ActCrisis : testcase01.ts11oeGetCrisisSet.ActCrisis= "crisis with ID 1 details"
487       tsi10.steve received from system actCoordinator.inactCoordinator.ieSendACrisis(ActCrisis)
488     }
489   }
490   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
491 }
492 //-----
493 test step instance tsi12: testcase01.ts12oeSetCrisisHandler{
494   variables {

```

```

495   reuse tsi10.steve as testcase01.ts12oeSetCrisisHandler.TheActor
496   AdtCrisisID : testcase01.ts12oeSetCrisisHandler.AdtCrisisID = "1"
497   tango : testcase01.ts12oeSetCrisisHandler.TheComCompany
498 }
499 oracle {
500   satisfaction = "true"
501   received message {
502     AMesssage : testcase01.ts12oeSetCrisisHandler.AMessage= 'You are now considered as handling the
503     crisis !'
504     AdtSMS : testcase01.ts12oeSetCrisisHandler.AdtSMS= 'The handling of your alert by our services
505     is in progress !'
506     AdtPhoneNumber : testcase01.ts12oeSetCrisisHandler.AdtPhoneNumber= "+3524666445252"
507   }
508   tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)
509   tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
510 }
511 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
512 }
513 //-----
514 test step instance tsi13: testcase01.ts13oeSetClock04{
515   variables {
516     theClock : testcase01.ts13oeSetClock04.TheActor
517     ACurrentClock : testcase01.ts13oeSetClock04.ACurrentClock = "2017:11:26 - 10:45:00"
518   }
519   oracle {
520     satisfaction = "true"
521   }
522   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
523 }
524 //-----
525 test step instance tsi14: testcase01.ts14oeValidateAlert{
526   variables {
527     reuse tsi10.steve as testcase01.ts14oeValidateAlert.TheActor
528     AdtAlertID : testcase01.ts14oeValidateAlert.AdtAlertID = "1"
529   }
530   oracle {
531     satisfaction = "true"
532     received message {
533       AMesssage : testcase01.ts14oeValidateAlert.AMessage= 'The Alert is now declared as valid !'
534       tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
535     }
536   }
537 }
538 test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
539 }
540 //-----
541 test step instance tsi15: testcase01.ts15oeAlert2{
542   variables {
543     reuse tsi12.tango as testcase01.ts15oeAlert2.TheActor
544     AetHumanKind : testcase01.ts15oeAlert2.AetHumanKind ="witness"
545     AdtDate : testcase01.ts15oeAlert2.AdtDate= "2017:11:26"
546     AdtTime : testcase01.ts15oeAlert2.AdtTime= "10:20:00"
547     AdtPhoneNumber : testcase01.ts15oeAlert2.AdtPhoneNumber= "+3524666445000"
548     AdtGPSLocation : testcase01.ts15oeAlert2.AdtGPSLocation= "49.627095:6.160251"
549     AdtComment : testcase01.ts15oeAlert2.AdtComment= "A car crash just happened."
550   }
551   message {
552     tsi12.tango sent to system testcase01.ts15oeAlert2.out : actComCompany.outactComCompany.oeAlert(
553       AetHumanKind,AdtDate,AdtTime,AdtPhoneNumber,AdtGPSLocation,AdtComment)
554   }
555   oracle {
556     satisfaction = "true"
557     received message {
558       AdtSMS : testcase01.ts15oeAlert2.AdtSMS= 'Your alert has been registered. We will handle it and
559       keep you informed'
560       tsi12.tango received from system actComCompany.inactComCompany.ieSmsSend(AdtPhoneNumber,AdtSMS)

```

```

561     }
562   }
563   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
564 }
565 //-----
566 test step instance tsi16: testcase01.ts16oeSetClock05{
567   variables {
568     reuse tsi13.theClock as testcase01.ts16oeSetClock05.TheActor
569     ACurrentClock : testcase01.ts16oeSetClock05.ACurrentClock = "2017:11:26 - 12:45:00"
570   }
571   oracle {
572     satisfaction = "true"
573     received message {
574       }
575     }
576   }
577   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
578 }
579 //-----
580 test step instance tsi17: testcase01.ts17oeSetCrisisStatus{
581   variables {
582     reuse tsi10.steve as testcase01.ts17oeSetCrisisStatus.TheActor
583     AdtCrisisID : testcase01.ts17oeSetCrisisStatus.AdtCrisisID = "1"
584     AetCrisisStatus : testcase01.ts17oeSetCrisisStatus.AetCrisisStatus= "solved"
585   }
586   oracle {
587     satisfaction = "true"
588     received message {
589       AMessage : testcase01.ts17oeSetCrisisStatus.AMessage= "The crisis status has been updated !"
590       tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
591     }
592   }
593   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
594 }
595 //-----
596 test step instance tsi18: testcase01.ts18oeReportOnCrisis{
597   variables {
598     reuse tsi10.steve as testcase01.ts18oeReportOnCrisis.TheActor
599     AdtCrisisID : testcase01.ts18oeReportOnCrisis.AdtCrisisID = "1"
600     AdtComment : testcase01.ts18oeReportOnCrisis.AdtComment= "3 victims sent to hospital, 2 cars
601     evacuated and 4 rescue unit mobilized"
602   }
603   oracle {
604     satisfaction = "true"
605     received message {
606       AMessage : testcase01.ts18oeReportOnCrisis.AMessage= 'The crisis comment has been updated !'
607       tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
608     }
609   }
610   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
611 }
612 //-----
613 test step instance tsi19: testcase01.ts19oeCloseCrisis{
614   variables {
615     reuse tsi10.steve as testcase01.ts19oeCloseCrisis.TheActor
616     AdtCrisisID : testcase01.ts19oeCloseCrisis.AdtCrisisID = "1"
617   }
618   oracle {
619     satisfaction = "true"
620     received message {
621       AMessage : testcase01.ts19oeCloseCrisis.AMessage= 'The crisis is now closed !'
622     }
623     tsi10.steve received from system actAuthenticated.inactAuthenticated.ieMessage(AMessage)
624   }
625 }
626   test results {pre-protocol = "true" pre-functional = "true" post-functional = "true"}
627 }
628

```

```

630 }
631 }
632 }
633 }
```

Listing B.57: Messir Spec. file tci-testcase01-instance01.msr.

## B.58 File . ./src-gen/messir-spec/usecases/usecase-suDeployAndRun.msr

```

1 package icrash.usecases.suDeployAndRun {
2   import icrash.concepts.primarytypes.datatypes
3   import icrash.environment
4   import icrash.usecases.suGlobalCrisisHandling
5   import icrash.usecases.ugAdministrateTheSystem
6   import icrash.usecases.subfunctions
7
8   Use Case Model {
9     use case system summary suDeployAndRun() {
10    actor actAdministrator[primary,active]
11    actor actMsrCreator[secondary,active]
12    actor actCoordinator[secondary,active,multiple]
13    actor actPolice[secondary,active,multiple]
14    actor actActivator[secondary,proactive]
15    actor actComCompany[secondary,active]
16
17    reuse oeCreateSystemAndEnvironment[1..1]
18    reuse ugAdministrateTheSystem[1...*]
19    reuse suGlobalCrisisHandling[1...*]
20    reuse oeSetClock[1...*]
21    reuse oeSollicitateCrisisHandling[0...*]
22    reuse oeAlert[1...*]
23
24    step a: actMsrCreator executes oeCreateSystemAndEnvironment
25    step b: actAdministrator executes ugAdministrateTheSystem
26    step c: actComCompany executes oeAlert
27    step d: actActivator executes oeSetClock
28    step ^e: actActivator executes oeSollicitateCrisisHandling
29    step f: actCoordinator executes suGlobalCrisisHandling
30    step h: actPolice executes suGlobalCrisisHandling
31
32    ordering constraint
33      "step (a) must be always the first step."
34    ordering constraint
35      "step (f) can be executed by different actCoordinator actors."
36    ordering constraint
37      "step (h) can be executed by different actPolice actors."
38    ordering constraint
39      "if (e) then previously (d)."
40  }
41 //-----
42 //-----
43 //-----
44 use case instance uciSimpleAndComplete : suDeployAndRun {
45   actors {
46     theCreator : actMsrCreator
47     theClock : actActivator
48     bill : actAdministrator
49     tango : actComCompany
50     steve : actCoordinator
51     kitty : actPolice
52   }
53   use case steps {
54 //-----
55   theCreator
56   executed instanceof subfunction
57     oeCreateSystemAndEnvironment("4")
58 //-----
```

```

59     theClock
60     executed instanceof subfunction
61         oeSetClock("2017:11:24 - 03:20:00"){}
62 //-----
63     bill
64     executed instanceof subfunction
65         oeLogin("icrashadmin", "7WXC1359"){
66             ieMessage('You are logged ! Welcome ...') returned to bill
67         }
68 //-----
69     bill
70     executed instanceof subfunction
71         oeAddCoordinator("1", "steve", "pwdMessirExcalibur2017"){
72             ieCoordinatorAddedreturned returned to bill
73         }
74 //-----
75     bill
76     executed instanceof subfunction
77         oeAddPolice("2", "kitty", "pwdPoliceKitty"){
78             iePoliceAddedreturned returned to bill
79         }
80 //-----
81     bill
82     executed instanceof subfunction
83         oeLogout{
84             ieMessage('You are logged out ! Good Bye ...') returned to bill
85         }
86 //-----
87     theClock
88     executed instanceof subfunction
89         oeSetClock("2017:11:26 - 10:15:00"){}
90 //-----
91     tango
92     executed instanceof subfunction
93         oeAlert("witness", "2017:11:26", "10:10:16", "+3524666445252",
94             "49.627675:6.159590", "3 cars involved in an accident."){
95             ieSmsSend("+3524666445252", "Your alert has been registered. We will handle it and keep you
informed") returned to tango
96         }
97 //-----
98 //-----
99     tango
100    executed instanceof subfunction
101        oeAlert("victim", "2017:11:15", "10:07:16", "+3524666466666",
102            "69.66675:61.166690", "A car and a bicycle involved in an accident."){
103            ieSmsSend("+3524666466666", "Your alert has been registered. We will handle it and keep you
informed") returned to tango
104        }
105 //-----
106    theClock
107    executed instanceof subfunction
108        oeSetClock("2017:11:26 - 10:30:00"){}
109 //-----
110    theClock
111    executed instanceof subfunction
112        oeSollicitateCrisisHandling{
113            ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
returned to bill
114            ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
returned to steve
115            ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
returned to kitty
116        }
117 //-----
118     steve
119     executed instanceof subfunction
120         oeLogin("steve", "pwdMessirExcalibur2017"){
121             ieMessage('You are logged ! Welcome ...') returned to steve
122         }
123 //-----
124 //-----
```

```

127     kitty
128     executed instanceof subfunction
129     oeLogin("kitty", "pwdPoliceKitty"){
130         ieMessage('You are logged ! Welcome ...') returned to kitty
131     }
132 //-----
133     steve
134     executed instanceof subfunction
135     oeGetCrisisSet("pending"){
136         ieSendACrisis("crisis with ID 1 details") returned to steve
137         ieSendACrisis("crisis with ID 2 details") returned to steve
138     }
139 //-----
140     kitty
141     executed instanceof subfunction
142     oeGetCrisisSet("pending"){
143         ieSendACrisis("crisis with ID 1 details") returned to kitty
144         ieSendACrisis("crisis with ID 2 details") returned to kitty
145     }
146 //-----
147     steve
148     executed instanceof subfunction
149     oeSetCrisisHandler("1"){
150         ieSmsSend("+3524666445252", "The handling of your alert by our services is in progress !")
151         returned to tango
152         ieMessage("You are now considered as handling the crisis !")
153         returned to steve
154     }
155 //-----
156     kitty
157     executed instanceof subfunction
158     oeSetCrisisHandler("2"){
159         ieSmsSend("+3524666466666", "The handling of your alert by our services is in progress !")
160         returned to tango
161         ieMessage("You are now considered as handling the crisis !")
162         returned to kitty
163     }
164 //-----
165     theClock
166     executed instanceof subfunction
167     oeSetClock("2017:11:26 - 10:45:00"){}
168 //-----
169     steve
170     executed instanceof subfunction
171     oeValidateAlert("1"){
172         ieMessage('The Alert is now declared as valid !')
173         returned to steve
174     }
175 //-----
176     steve
177     executed instanceof subfunction
178     oeValidateAlert("2"){
179         ieMessage('The Alert is now declared as valid !')
180         returned to steve
181     }
182 //-----
183     tango
184     executed instanceof subfunction
185     oeAlert("witness", "2017:11:26", "10:20:00", "+3524666445000",
186         "49.627095:6.160251", "A car crash just happened."){
187         ieSmsSend("+3524666445000", "Your alert has been registered. We will handle it and keep you
188         informed") returned to tango
189     }
190 //-----
191     theClock
192     executed instanceof subfunction
193     oeSetClock("2017:11:26 - 12:45:00"){}
194 //-----
195     steve
196     executed instanceof subfunction

```

```

196     oeSetCrisisStatus("1","solved"){
197         ieMessage('The crisis status has been updated !')
198         returned to steve
199     }
200 //-----
201     kitty
202     executed instanceof subfunction
203     oeSetCrisisStatus("2","solved"){
204         ieMessage('The crisis status has been updated !')
205         returned to kitty
206     }
207 //-----
208     steve
209     executed instanceof subfunction
210     oeReportOnCrisis("1","3 victims sent to hospital, 2 cars evacuated and 4 rescue unit
mobilized"){
211         ieMessage('The crisis comment has been updated !')
212         returned to steve
213     }
214 //-----
215     kitty
216     executed instanceof subfunction
217     oeReportOnCrisis("2","2 victims in hospital, 1 car and 1 bicycle evacuated"){
218         ieMessage('The crisis comment has been updated !')
219         returned to kitty
220     }
221 //-----
222     steve
223     executed instanceof subfunction
224     oeCloseCrisis("1"){
225         ieMessage('The crisis is now closed !')
226         returned to steve
227     }
228 //-----
229     kitty
230     executed instanceof subfunction
231     oeCloseCrisis("2"){
232         ieMessage('The crisis is now closed !')
233         returned to kitty
234     }
235
236 }
237 }
238 //-----
239 //-----
240 //-----
241 use case instance uciSimpleAndCompletePart01 : suDeployAndRun{
242
243     actors {
244         theCreator : actMsrCreator
245         theClock : actActivator
246         bill : actAdministrator
247         tango : actComCompany
248         steve : actCoordinator
249         kitty : actPolice
250     }
251     use case steps {
252 //-----
253         theCreator
254         executed instanceof subfunction
255         oeCreateSystemAndEnvironment("4")
256 //-----
257         theClock
258         executed instanceof subfunction
259         oeSetClock("2017:11:24 - 03:20:00")
260 //-----
261         bill
262         executed instanceof subfunction
263         oeLogin("icrashadmin", "7WXC1359")
264         ieMessage('You are logged ! Welcome ...') returned to bill

```

```

265     }
266 //-----
267     bill
268     executed instanceof subfunction
269     oeAddCoordinator("1", "steve", "pwdMessirExcalibur2017"){
270         ieCoordinatorAddedreturned returned to bill
271     }
272 //-----
273     bill
274     executed instanceof subfunction
275     oeAddPolice("2", "kitty", "pwdPoliceKitty"){
276         iePoliceAddedreturned returned to bill
277     }
278 //-----
279     bill
280     executed instanceof subfunction
281     oeLogout{
282         ieMessage('You are logged out ! Good Bye ...') returned to bill
283     }
284 //-----
285     theClock
286     executed instanceof subfunction
287     oeSetClock("2017:11:26 - 10:15:00){}
288 //-----
289     tango
290     executed instanceof subfunction
291     oeAlert("witness", "2017:11:26", "10:10:16", "+3524666445252",
292         "49.627675:6.159590", "3 cars involved in an accident."){
293         ieSmsSend("+3524666445252", "Your alert has been registered. We will handle it and keep you
informed") returned to tango
294     }
295 //-----
296     tango
297     executed instanceof subfunction
298     oeAlert("victim", "2017:11:15", "10:04:16", "+3524666466666",
299         "49.626665:16.159590", "A car and a bicycle involved in an accident."){
300         ieSmsSend("+3524666466666", "Your alert has been registered. We will handle it and keep you
informed") returned to tango
301     }
302 //-----
303     theClock
304     executed instanceof subfunction
305     oeSetClock("2017:11:26 - 10:30:00){}
306 //-----
307     theClock
308     executed instanceof subfunction
309     oeSollicitateCrisisHandling{
310         ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
311         returned to bill
312         ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
313         returned to steve
314         ieMessage("There are alerts pending since more than the defined delay. Please REACT !")
315         returned to kitty
316     }
317 }
318 }
319 //-----
320 //-----
321 //-----
322 use case instance uciSimpleAndCompletePart02 : suDeployAndRun{
323     actors {
324         theCreator : actMsrCreator
325         theClock : actActivator
326         bill : actAdministrator
327         tango : actComCompany
328         steve : actCoordinator
329         kitty : actPolice
330     }
331     use case steps {
332

```

```

333 //-----
334     steve
335     executed instanceof subfunction
336         oeLogin("steve", "pwdMessirExcalibur2017"){
337             ieMessage('You are logged ! Welcome ...') returned to steve
338         }
339 //-----
340     kitty
341     executed instanceof subfunction
342         oeLogin("kitty", "pwdPoliceKitty"){
343             ieMessage('You are logged ! Welcome ...') returned to kitty
344         }
345 //-----
346     steve
347     executed instanceof subfunction
348         oeGetCrisisSet("pending"){
349             ieSendACrisis("crisis with ID 1 details") returned to steve
350             ieSendACrisis("crisis with ID 2 details") returned to steve
351         }
352 //-----
353     kitty
354     executed instanceof subfunction
355         oeGetCrisisSet("pending"){
356             ieSendACrisis("crisis with ID 1 details") returned to kitty
357             ieSendACrisis("crisis with ID 2 details") returned to kitty
358         }
359 //-----
360     steve
361     executed instanceof subfunction
362         oeSetCrisisHandler("1"){
363             ieSmsSend("+3524666445252", "The handling of your alert by our services is in progress !")
364             returned to tango
365             ieMessage("You are now considered as handling the crisis !")
366             returned to steve
367         }
368 //-----
369     kitty
370     executed instanceof subfunction
371         oeSetCrisisHandler("2"){
372             ieSmsSend("+3524666466666", "The handling of your alert by our services is in progress !")
373             returned to tango
374             ieMessage("You are now considered as handling the crisis !")
375             returned to kitty
376         }
377 //-----
378     theClock
379     executed instanceof subfunction
380         oeSetClock("2017:11:26 - 10:45:00"){}
381 //-----
382     steve
383     executed instanceof subfunction
384         oeValidateAlert("1"){
385             ieMessage('The Alert is now declared as valid !')
386             returned to steve
387         }
388 //-----
389     steve
390     executed instanceof subfunction
391         oeValidateAlert("2"){
392             ieMessage('The Alert is now declared as valid !')
393             returned to steve
394         }
395 //-----
396     tango
397     executed instanceof subfunction
398         oeAlert("witness", "2017:11:26", "10:20:00", "+3524666445000",
399             "49.627095:6.160251", "A car crash just happened."){
400             ieSmsSend("+3524666445000", "Your alert has been registered. We will handle it and keep you
401             informed") returned to tango
402         }

```

## B.59. FILE /SRC-GEN/MESSIR-SPEC/USECASES/USECASE-SUGLOBALCRISISHANDLING.MSR251

```
402 //-----
403     theClock
404     executed instanceof subfunction
405         oeSetClock("2017:11:26 - 12:45:00"){}
406 //-----
407     steve
408     executed instanceof subfunction
409         oeSetCrisisStatus("1","solved"){
410             ieMessage('The crisis status has been updated !')
411             returned to steve
412         }
413 //-----
414     kitty
415     executed instanceof subfunction
416         oeSetCrisisStatus("2","solved"){
417             ieMessage('The crisis status has been updated !')
418             returned to kitty
419         }
420 //-----
421     steve
422     executed instanceof subfunction
423         oeReportOnCrisis("1","3 victims sent to hospital, 2 cars evacuated and 4 rescue unit
mobilized"){
424             ieMessage('The crisis comment has been updated !')
425             returned to steve
426         }
427 //-----
428     kitty
429     executed instanceof subfunction
430         oeReportOnCrisis("2","2 victims sent to hospital, 1 car and 1 bicycle evacuated"){
431             ieMessage('The crisis comment has been updated !')
432             returned to kitty
433         }
434 //-----
435     steve
436     executed instanceof subfunction
437         oeCloseCrisis("1"){
438             ieMessage('The crisis is now closed !')
439             returned to steve
440         }
441 //-----
442     kitty
443     executed instanceof subfunction
444         oeCloseCrisis("2"){
445             ieMessage('The crisis is now closed !')
446             returned to kitty
447         }
448     }
449   }
450 }
451 }
452 }
```

Listing B.58: Messir Spec. file usecase-suDeployAndRun.msr.

## B.59 File [./src-gen/messir-spec/usecases/usecase-suGlobalCrisisHandling.msr](#)

```
1 package icrash.usecases.suGlobalCrisisHandling {
2 import lu.uni.lassy.messir.libraries.primitives
3 import icrash.environment
4 import icrash.usecases.subfunctions
5 import icrash.usecases.ugSecurelyUseSystem
6 import icrash.usecases.ugManageCrisis
7 import icrash.usecases.ugMonitor
8
9 Use Case Model {
10 use case system summary
```

```

11  suGlobalCrisisHandling() {
12    actor actCoordinator[primary, active]
13    actor actPolice[primary, active]
14
15    reuse ugSecurelyUseSystem[1...*]
16    reuse ugMonitor[1...*]
17    reuse ugManageCrisis[1...*]
18
19    step a: actCoordinator
20      executes ugSecurelyUseSystem
21    step b: actCoordinator
22      executes ugMonitor
23    step c: actCoordinator
24      executes ugManageCrisis
25
26    step d: actPolice
27      executes ugSecurelyUseSystem
28    step e: actPolice
29      executes ugMonitor
30    step f: actPolice
31      executes ugManageCrisis
32
33  ordering constraint
34    "steps (a) (b) and (c) executions are interleaved
35    (steps (b) and (c) have their protocol constrained by steps of (a)).
36    steps (d) (e) and (f) executions are interleaved
37    (steps (e) and (f) have their protocol constrained by steps of (d))."
38  ordering constraint
39    "steps (a) (b) and (c) can be executed multiple times.
40    steps (d) (e) and (f) can be executed multiple times."
41
42 }
43 }

```

Listing B.59: Messir Spec. file usecase-suGlobalCrisisHandling.msr.

## B.60 File [./src-gen/messir-spec/usecases/usecase-ugAdministrateTheSystem.msr](#)

```

1 package icrash.usecases.ugAdministrateTheSystem {
2
3   import icrash.environment
4   import icrash.usecases.ugSecurelyUseSystem
5   import icrash.usecases.subfunctions
6
7   Use Case Model {
8
9     use case system usergoal
10    ugAdministrateTheSystem() {
11      actor actAdministrator[primary, active]
12
13      reuse ugSecurelyUseSystem[1...*]
14      reuse oeAddCoordinator[1...*]
15      reuse oeDeleteCoordinator[0...*]
16      reuse oeAddPolice[1...*]
17      reuse oeDeletePolice[0...*]
18
19      step a: actAdministrator
20        executes ugSecurelyUseSystem
21      step b: actAdministrator
22        executes oeAddCoordinator
23      step c: actAdministrator
24        executes oeDeleteCoordinator
25
26      step d: actAdministrator
27        executes oeAddPolice
28      step f: actAdministrator
29        executes oeDeletePolice

```

```

30
31     ordering constraint
32     "steps (a) (b) (c) (d) and (f) executions are interleaved
33     (steps (b) (c) (d) and (f) have their protocol constrained
34     by steps of (a))."
35     ordering constraint
36     "steps (a) (b) (c) (d) and (f) can be executed multiple times."
37   }
38 }
39 }
```

Listing B.60: Messir Spec. file usecase-ugAdministrateTheSystem.msr.

## B.61 File ../src-gen/messir-spec/usecases/usecase-ugManageCrisis.msr

```

1 package icrash.usecases.ugManageCrisis {
2
3   import icrash.environment
4   import icrash.usecases.subfunctions
5
6   Use Case Model {
7
8     use case system usergoal ugManageCrisis() {
9       actor actCoordinator[primary, active]
10      actor actPolice[primary, active]
11
12      reuse oeValidateAlert[0...*]
13      reuse oeSetCrisisStatus[0...*]
14      reuse oeSetCrisisHandler[0...*]
15      reuse oeReportOnCrisis[0...*]
16      reuse oeCloseCrisis[0...*]
17      reuse oeInvalidateAlert[0...*]
18
19      step a: actCoordinator executes oeValidateAlert
20      step b: actCoordinator executes oeSetCrisisStatus
21      step h: actCoordinator executes oeSetCrisisType
22      step c: actCoordinator executes oeSetCrisisHandler
23      step d: actCoordinator executes oeReportOnCrisis
24      step f: actCoordinator executes oeCloseCrisis
25      step g: actCoordinator executes oeInvalidateAlert
26
27      step k: actPolice executes oeSetCrisisStatus
28      step l: actPolice executes oeSetCrisisHandler
29      step m: actPolice executes oeReportOnCrisis
30      step n: actPolice executes oeCloseCrisis
31
32      ordering constraint "managing a crisis is doing one of the indicated use cases."
33
34   }
35
36 }
37 }
```

Listing B.61: Messir Spec. file usecase-ugManageCrisis.msr.

## B.62 File ../src-gen/messir-spec/usecases/usecase-ugMonitor.msr

```

1 package icrash.usecases.ugMonitor {
2
3   import icrash.environment
4   import icrash.usecases.subfunctions
5
6   Use Case Model {
7     use case system usergoal ugMonitor() {
8       actor icrash.environment.actCoordinator[primary,active]
9       actor icrash.environment.actPolice[primary,active]
```

```
10
11 reuse oeGetCrisisSet[0...*]
12 reuse oeGetAlertsSet[0...*]
13
14 step a: icrash.environment.actCoordinator executes oeGetAlertsSet
15 step b: icrash.environment.actCoordinator executes oeGetCrisisSet
16 step c: icrash.environment.actPolice executes oeGetCrisisSet
17 }
18 }
19 }
```

Listing B.62: Messir Spec. file usecase-ugMonitor.msr.

B.63 File .  
./src-gen/messir-spec/usecases/usecase-  
ugSecurelyUseSystem.msr

```
1 package icrash.usecases.ugSecurelyUseSystem {
2
3 import icrash.environment
4 import icrash.usecases.subfunctions
5
6 Use Case Model {
7
8 use case system usergoal
9 ugSecurelyUseSystem() {
10
11 actor actAuthenticated[primary,active]
12
13 reuse oeLogin[1..1]
14 reuse oeLogout[1..1]
15
16 step a: actAuthenticated
17     executes oeLogin
18 step b: actAuthenticated
19     executes oeLogout
20
21 ordering constraint
22     "step (a) must always precede step (b)."
23 }
24 }
25 }
```

Listing B.63: Messir Spec. file usecase-ugSecurelyUseSystem.msr.

B.64 File .  
./src-gen/messir-spec/usecases/usecaseinstance-  
ugSecurelyUseSystem-uciugSecurelyUseSystem.msr

```
1 package usecases.uciugSecurelyUseSystem {
2 import icrash.usecases.ugSecurelyUseSystem
3 import icrash.usecases.ugSecurelyUseSystem
4 import icrash.concepts.primarytypes.datatypes
5 import icrash.environment
6 import icrash.usecases.suGlobalCrisisHandling
7 import icrash.usecases.ugAdministrateTheSystem
8 import icrash.usecases.subfunctions
9
10 Use Case Model {
11
12 //-----
13 use case instance uciugSecurelyUseSystem : ugSecurelyUseSystem {
14   actors {
15     bill:actAuthenticated
16   }
17   use case steps {
18 //-----
19     bill
```

B.64. FILE /SRC-GEN.../USECASEINSTANCE-UGSECURELYUSESYSTEM-UCIUGSECURELYUSESYSTEM

```
20     executed instanceof subfunction
21         oeLogin("icrashadmin","7WXC1359"){
22             ieMessage('You are logged ! Welcome ...') returned to bill
23         }
24 //-----
25     bill
26     executed instanceof subfunction
27         oeLogout{
28             ieMessage('You are logged out ! Good Bye ...') returned to bill
29         }
30     }
31 }
32 }
33 }
```

Listing B.64: Messir Spec. file usecaseinstance-ugSecurelyUseSystem-uciugSecurelyUseSystem.msr.



## Appendix C

# Listing of the Prolog Files Referenced in the Operation Model Specification

### C.1 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactActi oeSetClock.pl

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 /* DISCONTIGUOUS PREDICATES */
3 :- multifile msrop/4.
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 %-----%
6 msrop(outactActivator,
7     oeSetClock,
8     [preProtocol,Self,
9      AcurrentClock
10     ],
11     []):-!
12 /* Pre Protocol:*/
13 /* PreP01 */
14 msrVar(ctState,TheSystem),
15 msrVar(ptBoolean,AvpStarted),
16
17 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
18
19 msrNav([Self],[rnActor,rnSystem,vpStarted],[AvpStarted]),
20 AvpStarted = [ptBoolean,true],
21
22 msrNav([TheSystem],
23     [clock,lt,[AcurrentClock]],
24     [[ptBoolean,true]])
25 .
26
27 msrop(outactActivator,
28     oeSetClock,
29     [preFunctional,Self,
30      AcurrentClock
31     ],
32     []):-!
33 /* Pre Functional:*/
34 /* PreF01 */
35 true.
36
37 msrop(outactActivator,
38     oeSetClock,
39     [post,Self,
40      AcurrentClock
41     ],
42     []):-!
```

```

44 msrVar(ctState,TheSystem),
45
46 /* Post Functional:*/
47
48 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
49
50 /* PostF01 */
51 msrNav([TheSystem],
52     [msmAtPost,clock],
53     [AcurrentClock]),
54
55 /* Post Protocol:*/
56 /* PostP01 */
57 true
58 .

```

Listing C.1: Prolog file outactActivator-oeSetClock.pl.

## C.2 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactActivator-oeSollicitateCrisisHandling.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6
7msrop(outactActivator,
8    oeSollicitateCrisisHandling,
9    [preProtocol,Self
10    ],
11    []):-!
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
15
16 msrVarCol(ctCrisis,_,ColctCrisisToHandle),
17
18/* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23/* PreP02 */
24 msrNav([TheSystem],
25     [rnctCrisis,msrSelect,
26      handlingDelayPassed,[]]
27    ],
28    ColctCrisisToHandle),
29
30 msrNav(ColctCrisisToHandle,
31     [msrSize,geq,[[ptInteger,1]]],
32     [[ptBoolean,true]]),
33 .
34
35msrop(outactActivator,
36    oeSollicitateCrisisHandling,
37    [preFunctional,Self
38    ],
39    []):-!
40/* Pre Functional:*/
41/* PreF01 */
42true.
43
44msrop(outactActivator,
45    oeSollicitateCrisisHandling,
46    [post,Self
47    ],

```

## C.2. FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTACTIVATOR-OESOLLICITATECRISISHANDLING.PL

```

48      []):-  
49  
50 msrVar(ctState,TheSystem),  
51 msrVar(dtComment,AMessageForCrisisHandlers),  
52 msrVar(dtDateAndTime, TheClock),  
53 msrVarCol(ctCrisis,_,ColctCrisisToAllocateIfPossible),  
54  
55/* Post Functional:*/  
56 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  
57  
58 /* PostF01 */  
59 msrNav([TheSystem],  
60   [rnctCrisis,msrSelect,  
61    maxHandlingDelayPassed,[]  
62  ],  
63 ColctCrisisToAllocateIfPossible),  
64  
65msrNav(ColctCrisisToAllocateIfPossible,  
66   [msrForAll,isAllocatedIfPossible,[],  
67   [[ptBoolean,true]]],  
68  
69 /* PostF02 */  
70 msrNav([TheSystem],  
71   [rnctCrisis,msrSelect,  
72    handlingDelayPassed,[]  
73  ],  
74 ColctCrisisToHandle),  
75  
76 msrNav(ColctCrisisToHandle,  
77   [msrColSubtract,[ColctCrisisToAllocateIfPossible]  
78  ],  
79 ColctCrisisToRemind),  
80  
81 (msrNav(ColctCrisisToRemind,  
82   [msrSize,geq,[[ptInteger,1]]],  
83   [[ptBoolean,true]])  
84 -> (msrNav([AMessageForCrisisHandlers],  
85   [value],  
86   [[ptString,'There are alerts pending since more than the defined delay. Please REACT !']])),  
87  
88 msrNav([TheSystem],  
89   [rnactAdministrator,rnInterfaceIN,  
90    ieMessage,[AMessageForCrisisHandlers]  
91  ],  
92   [[ptBoolean,true]]),  
93  
94 msrNav([TheSystem],  
95   [rnactCoordinator,msrForAll,rnInterfaceIN,  
96    ieMessage,[AMessageForCrisisHandlers]  
97  ],  
98   [[ptBoolean,true]]))  
99 )  
100 ; true  
101 ),  
102  
103/* Post Protocol:*/  
104/* PostP01 */  
105 msrNav([TheSystem],  
106   [clock],  
107   [TheClock]),  
108  
109 msrNav([TheSystem],  
110   [msmAtPost,vpLastReminder],  
111   [TheClock])  
112 .

```

Listing C.2: Prolog file outactActivator-oeSollicitateCrisisHandling.pl.

### C.3 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outact oeAddCoordinator.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%-----%
6msrop(outactAdministrator,
7    oeAddCoordinator,
8    [preProtocol,Self,
9     AdtCoordinatorID,
10    AdtLogin,
11    AdtPassword
12    ],
13    []):-!
14/* Pre Protocol:*/
15 msrVar(ctState,TheSystem),
16 msrVar(actAdministrator,TheActor),
17 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
18 msrNav([Self],[rnActor],[TheActor]),
19
20/* PreP01 */
21 msrNav([TheSystem],
22     [vpStarted],
23     [[ptBoolean,true]]),
24
25/* PreP02 */
26 msrNav([TheActor],
27     [rnctAuthenticated,vpIsLogged],
28     [[ptBoolean,true]])
29
30 .
31
32msrop(outactAdministrator,
33    oeAddCoordinator,
34    [preFunctional,Self,
35     AdtCoordinatorID,
36     AdtLogin,
37     AdtPassword
38     ],
39    []):-!
40/* Pre Functional:*/
41 msrVar(ctState,TheSystem),
42 msrVar(actAdministrator,TheActor),
43 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
44 msrNav([Self],[rnActor],[TheActor]),
45/* PreF01 */
46 msrNav([TheSystem],
47     [rnctCoordinator,
48      msrSelect,id.eq,[AdtCoordinatorID]],
49     ColctCoordinators),
50 msrNav(ColctCoordinators,
51     [msrIsEmpty],
52     [[ptBoolean,true]]))
53 .
54
55msrop(outactAdministrator,
56    oeAddCoordinator,
57    [post,Self,
58     AdtCoordinatorID,
59     AdtLogin,
60     AdtPassword
61     ],
62    []):-!
63
64/* Post Functional:*/
65 msrVar(ctState,TheSystem),
66 msrVar(actAdministrator,TheActor),

```

#### C.4 FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTADMINISTRATOR-OEDELETECOORDINATOR.PL

```

67 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
68 msrNav([Self],[rnActor],[TheActor]),
69
70 msrVar(actCoordinator,TheactCoordinator),
71 msrVar(ctCoordinator,ThectCoordinator),
72
73 /* PostF01 */
74 msrNav([TheactCoordinator],
75     [init,[]],
76     [[ptBoolean,true]]),
77
78 /* PostF02 */
79 msrNav([ThectCoordinator],
80     [init,[AdtCoordinatorID,AdtLogin,AdtPassword]],
81     [[ptBoolean,true]]),
82
83 /* PostF03 */
84 msrNav([TheactCoordinator],
85     [msmAtPost,rnctCoordinator],
86     [ThectCoordinator]),
87
88 /* PostF04 */
89 msrNav([ThectCoordinator],
90     [msmAtPost,rnactAuthenticated],
91     [TheactCoordinator]),
92
93 /* PostF05 */
94 msrNav([TheActor],
95     [rnInterfaceIN,
96      ieCoordinatorAdded,[]],
97     [[ptBoolean,true]]),
98
99 /* Post Protocol:*/
100 /* PostP01 */
101 true
102 .

```

Listing C.3: Prolog file outactAdministrator-oeAddCoordinator.pl.

#### C.4 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactAdministrator-oeDeleteCoordinator.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5%-----%
6msrop(outactAdministrator,
7    oeDeleteCoordinator,
8    [preProtocol,Self,
9     AdtCoordinatorID
10    ],
11    []):-%
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actAdministrator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]])
26.

```

## 262 APPENDIX C. LISTING OF THE PROLOG FILES REFERENCED IN THE OPERATION MODEL SPE

```

27
28 msrop(outactAdministrator,
29     oeDeleteCoordinator,
30     [preFunctional,Self,
31      AdtCoordinatorID
32      ],
33      []):-!
34 /* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actAdministrator,TheActor),
37 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
38 msrNav([Self],[rnActor],[TheActor]),
39
40 /* PreF01 */
41 msrNav([TheSystem],
42     [rnctCoordinator,
43      msrSelect,id,eq,[AdtCoordinatorID]],
44      ColctCoordinators),
45
46 msrNav(ColctCoordinators,
47     [msrSize,eq,[[ptInteger,1]]],
48     [[ptBoolean,true]]).
49
50 msrop(outactAdministrator,
51     oeDeleteCoordinator,
52     [post,Self,
53      AdtCoordinatorID
54      ],
55      []):-!
56
57 /* Post Functional:*/
58 msrVar(ctState,TheSystem),
59 msrVar(actAdministrator,TheActor),
60 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
61 msrNav([Self],[rnActor],[TheActor]),
62
63 /* PostF01 */
64 msrNav([TheSystem],
65     [rnctCoordinator,
66      msrSelect,id,eq,[AdtCoordinatorID]],
67      [ThectCoordinator]),
68
69 msrNav([ThectCoordinator],
70     [rnactCoordinator,msrForAll,msrIsKilled,
71     [[ptBoolean,true]]),
72
73 msrNav([ThectCoordinator],
74     [msrIsKilled,
75     [[ptBoolean,true]]),
76
77 /* PostF02 */
78 msrNav([TheActor],
79     [rnInterfaceIN,
80      ieCoordinatorDeleted,[]]
81     ],
82     [[ptBoolean,true]]),
83
84 /* Post Protocol:*/
85 /* PostP01 */
86 true
87 .

```

Listing C.4: Prolog file outactAdministrator-oeDeleteCoordinator.pl.

## C.5 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactAdministrator-oeLogin.pl

1%%%%%%%%%%%%%

## C.5. FILE /SRC-GEN/PROLOG-REF-SPEC/OPERATIONS.../OUTACTA AUTHENTICATED-OELOGIN.PL263

```
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5%-----%
6msrop(outactAuthenticated,
7    oeLogin,
8    [preProtocol,Self,
9     AdtLogin,
10    AdtPassword
11    ],
12    []):-%
13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actAuthenticated,TheactAuthenticated),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheactAuthenticated]),
18 .
19 /* PreP01 */
20 msrNav([TheSystem],
21     [vpStarted,
22      [[ptBoolean,true]]]),
23 .
24 msrNav([TheactAuthenticated],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,false]])
27 .
28 .
29msrop(outactAuthenticated,
30    oeLogin,
31    [preFunctional,Self,
32     AdtLogin,
33     AdtPassword
34     ],
35    []):-%
36/* Pre Functional:*/
37/* PreF01 */
38true
39.
40.
41msrop(outactAuthenticated,
42    oeLogin,
43    [post,Self,
44     AdtLogin,
45     AdtPassword
46     ],
47    []):-%
48 .
49 msrVar(ctState,TheSystem),
50 msrVar(actAuthenticated,TheactAuthenticated),
51 .
52 msrVar(ptString,AptStringMessageForTheactAuthenticated),
53 msrVar(ptString,AptStringMessageForTheactAdministrator),
54 .
55/* Post Functional:*/
56 .
57 msrNav([Self],[rnActor],[TheactAuthenticated]),
58 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
59 .
60/* PostF01 */
61 .
62 ( (msrNav([TheactAuthenticated],
63     [rnctAuthenticated,pwd],
64     [AdtPassword]),
65   msrNav([TheactAuthenticated],
66     [rnctAuthenticated,login],
67     [AdtLogin])
68   )
69 -> ( msrNav([AptStringMessageForTheactAuthenticated],
70     [eq,[[ptString,'You are logged ! Welcome ...']]],
71     [[ptBoolean,true]]),
```

```

72     msrNav([TheactAuthenticated],
73         [rnInterfaceIN,
74             ieMessage,[AptStringMessageForTheactAuthenticated]],
75             [[ptBoolean,true]])
76     )
77 ; ( msrNav([AptStringMessageForTheactAuthenticated],
78             [eq,[[ptString,'Wrong identification information ! Please try again ...']]),
79             [[ptBoolean,true]]),
80     msrNav([TheactAuthenticated],
81         [rnInterfaceIN,
82             ieMessage,[AptStringMessageForTheactAuthenticated]],
83             [[ptBoolean,true]]),
84
85     msrNav([AptStringMessageForTheactAdministrator],
86             [eq,[[ptString,'Intrusion tentative !']]),
87             [[ptBoolean,true]]),
88     msrNav([TheSystem],
89         [rnactAdministrator,rnInterfaceIN,
90             ieMessage,[AptStringMessageForTheactAdministrator]],
91             [[ptBoolean,true]])
92     )
93 ),
94
95 /* Post Protocol:*/
96 /* PostP01 */
97 ( (msrNav([TheactAuthenticated],
98     [rnctAuthenticated,pwd],
99     [AdtPassword]),
100    msrNav([TheactAuthenticated],
101        [rnctAuthenticated,login],
102        [AdtLogin])
103    )
104    -> (msrNav([TheactAuthenticated],
105        [rnctAuthenticated,msmAtPost,vpIsLogged],
106        [[ptBoolean,true]])
107    )
108 ; true
109 )
110 .

```

Listing C.5: Prolog file outactAuthenticated-oeLogin.pl.

## C.6 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactAuthenticated-oeLogout.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactAuthenticated,
7    oeLogout,
8    [preProtocol,Self
9    ],
10   []):- 
11/* Pre Protocol:*/
12 msrVar(ctState,TheSystem),
13 msrVar(actAuthenticated,TheActor),
14 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
15 msrNav([Self],[rnActor],[TheActor]),
16
17/* PreP01 */
18 msrNav([TheSystem],
19     [vpStarted],
20     [[ptBoolean,true]]),
21
22 msrNav([TheActor],
23     [rnctAuthenticated,vpIsLogged],

```

## C.7 FILE /SRC-GEN/PROLOG-REF-SPEC/OPERATIONS.../OUTACTCOMCOMPANY-OEALERT.PL265

```

24     [[ptBoolean,true]]))
25 .
26
27msrop(outactAuthenticated,
28     oeLogout,
29     [preFunctional,Self
30     ],
31     []):-!
32/* Pre Functional:*/
33/* PreF01 */
34true
35.
36
37msrop(outactAuthenticated,
38     oeLogout,
39     [post,Self
40     ],
41     []):-!
42
43 msrVar(ctState,TheSystem),
44 msrVar(actAuthenticated,TheactAuthenticated),
45
46 msrVar(ptString,AptStringMessageForTheactAuthenticated),
47
48/* Post Functional:*/
49 msrNav([Self],[rnActor],[TheactAuthenticated]),
50 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
51
52/* PostF01 */
53 msrNav([AptStringMessageForTheactAuthenticated],
54     [eq,[[ptString,'You are logged out ! Good Bye ...']]],

55     [[ptBoolean,true]]),
56 msrNav([TheactAuthenticated],
57     [rnInterfaceIN,
58      ieMessage,[AptStringMessageForTheactAuthenticated]],
59     [[ptBoolean,true]]),
60
61 /* Post Protocol:*/
62/* PostP01 */
63msrNav([TheactAuthenticated],
64     [rnctAuthenticated,msmAtPost,vpIsLogged],
65     [[ptBoolean,false]])
66.
```

Listing C.6: Prolog file outactAuthenticated-oeLogout.pl.

## C.7 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactComoeAlert.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6nico(A):-
7 trace,
8 write('here'),
9 write('\n').
10
11msrop(outactComCompany,
12     oeAlert,
13     [preProtocol,Self,
14      AetHumanKind,
15      AdtDate,
16      AdtTime,
17      AdtPhoneNumber,
18      AdtGPSLocation,
19      AdtComment
```

266 APPENDIX C. LISTING OF THE PROLOG FILES REFERENCED IN THE OPERATION MODEL SPE

```

20      ],
21      []):-  

22 /* Pre Protocol:*/  

23 msrVar(ctState,TheSystem),  

24 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  

25 /* PreP01 */  

26 msrNav([TheSystem],  

27     [vpStarted],  

28     [[ptBoolean,true]]))  

29 .  

30  

31 msrop(outactComCompany,  

32     oeAlert,  

33     [preFunctional,Self,  

34     AetHumanKind,  

35     AdtDate,  

36     AdtTime,  

37     AdtPhoneNumber,  

38     AdtGPSLocation,  

39     AdtComment  

40     ],  

41     []):-  

42 /* Pre Functional:*/  

43 /* PreF01 */  

44 msrVar(ctState,TheSystem),  

45 msrNav([Self],  

46     [msmAtPre,rnActor,rnSystem],  

47     [TheSystem]),  

48  

49 ( msrNav([TheSystem],[clock,date,gt,[AdtDate]],[[ptBoolean,true]]))  

50 ; (msrNav([TheSystem],[clock,date,eq,[AdtDate]],[[ptBoolean,true]]))  

51 , msrNav([TheSystem],[clock,time,gt,[AdtTime]],[[ptBoolean,true]]))  

52 )  

53 )  

54 .  

55  

56 msrop(outactComCompany,  

57     oeAlert,  

58     [post,Self,  

59     AetHumanKind,  

60     AdtDate,  

61     AdtTime,  

62     AdtPhoneNumber,  

63     AdtGPSLocation,  

64     AdtComment  

65     ],  

66     []):-  

67  

68 msrVar(ctState,TheSystem),  

69 msrVar(ctHuman,ActHuman),  

70 msrVar(actComCompany,TheactComCompany),  

71 msrVar(ctAlert,ActAlert),  

72 msrVar(dtDateAndTime,AAlertInstant),  

73 msrVar(etAlertStatus,AetAlertStatus),  

74% msrVar(ctAlert,ActAlertNearBy),  

75 msrVar(ctCrisis,ActCrisis),  

76 msrVar(dtCrisisID,AdtCrisisID),  

77% msrVar(etCrisisType,AetCrisisType),  

78 msrVar(etCrisisStatus,AetCrisisStatus),  

79 msrVar(dtDateAndTime,ACrisisInstant),  

80 msrVar(dtComment,ACrisisdtComment),  

81% msrVar(ptString,AptStringMessage),  

82 msrVar(dtSMS,AdtSMS),  

83 msrVar(dtAlertID,AdtAlertID),  

84  

85% msrVar(ptInteger,TheNextptIntegerValue),  

86% msrVar(ptInteger,UpdatedNextptIntegerValue),  

87% msrVar(inactComCompany,TheComCompanyIN),  

88% msrVar(dtComment,TheCommentStored),  

89% msrVar(dtString,TheCommentStoreddtString),

```

## C.7. FILE /SRC-GEN/PROLOG-REF-SPEC/OPERATIONS.../OUTACTCOMCOMPANY-OEALERT.PL267

```

160   msrNav([ActCrisis],[init,[AdtCrisisID,
161             AdtCrisisType,
162             AetCrisisStatus,
163             AdtGPSLocation,
164             ACrisisInstant,
165             ACrisisdtComment]],,
166             [[ptBoolean,true]]))
167
168   )
169 ; (
170   msrNav(ColctAlertsNearBy,
171             [rnTheCrisis,msrAny,msrTrue],
172             [ActCrisis])
173   ),
174 ),
175
176 /* PostF04 */
177
178 msrNav([ActAlert,
179           [msmAtPost,rnTheCrisis],
180           [ActCrisis]),
181
182 /* PostF05 */
183
184 msrNav([TheSystem],
185           [rnctHuman,
186             msrSelect,id,eq,[AdtPhoneNumber]],
187             HumanColl),
188
189 msrNav(HumanColl,
190           [msrSelect,kind,etEq,[AetHumanKind]],
191             HumanCol2),
192
193 (msrNav(HumanCol2,[msrIsEmpty],[[ptBoolean,true]]))
194 -> (msrNav([ActHuman],
195             [init,[AdtPhoneNumber,AetHumanKind]],
196             [[ptBoolean,true]])),
197   msrNav([ActHuman],
198             [msmAtPost,rnactComCompany],
199             [TheactComCompany])
200   )
201 ; msrNav(HumanCol2,
202             [msrAny],
203             [ActHuman])
204 ),
205
206 msrNav([ActHuman],
207             [rnSignaled,msrIncluding,[ActAlert]],
208             ColAlerts),
209
210 msrNav([ActHuman],
211             [msmAtPost,rnSignaled],
212             ColAlerts),
213
214 /* PostF06 */
215 msrNav([AdtSMS],
216             [value],
217             [[ptString,'Your alert has been registered. We will handle it and keep you informed']])),
218 msrNav([TheactComCompany],
219             [rnInterfaceIN,
220               ieSmsSend,[AdtPhoneNumber,
221                           AdtSMS]],[[ptBoolean,true]]),
222
223 /*
224
225 */
226
227 /* Post Protocol:*/
228 /* PostP01 */
229 true

```

## C.8 FILE /SRC-GEN/PROLOG-REF-SPEC/OPERATIONS.../OUTACTCOORDINATOR-OECLOSECRISIS.PL

230 .

Listing C.7: Prolog file outactComCompany-oeAlert.pl.

## C.8 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeCloseCrisis.pl

```
1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeCloseCrisis,
8    [preProtocol,_Self,
9     AdtCrisisID
10    ],
11    []):-!
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([_Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([_Self],[rnActor],[TheActor]),
17 .
18/* PreP01 */
19 msrNav([TheSystem],
20        [vpStarted],
21        [[ptBoolean,true]]),
22 .
23/* PreP02 */
24 msrNav([TheActor],
25        [rnctAuthenticated,vpIsLogged],
26        [[ptBoolean,true]]),
27 .
28
29msrop(outactCoordinator,
30    oeCloseCrisis,
31    [preFunctional,_Self,
32     AdtCrisisID
33    ],
34    []):-!
35/* Pre Functional:*/
36 msrVar(ctState,TheSystem),
37 msrVar(actCoordinator,TheActor),
38 .
39 msrVar(dtCrisisID,AdtCrisisID),
40 .
41 msrNav([_Self],[rnActor,rnSystem],[TheSystem]),
42 msrNav([_Self],[rnActor],[TheActor]),
43 .
44/* PreF01 */
45 msrNav([TheSystem],
46        [rnctCrisis,
47         msrSelect,
48         id,eq,[AdtCrisisID]
49        ],
50        ColCrisis),
51 .
52 msrNav(ColCrisis,
53        [msrSize,eq,[[ptInteger,1]]],
54        [[ptBoolean,true]]),
55 .
56
57msrop(outactCoordinator,
58    oeCloseCrisis,
59    [post,_Self,
60     AdtCrisisID
61    ],
```

## 270 APPENDIX C. LISTING OF THE PROLOG FILES REFERENCED IN THE OPERATION MODEL SPE

```

62      []):-  

63  

64 /* Post Functional: */  

65 msrVar(ctState,TheSystem),  

66 msrVar(actCoordinator,TheActor),  

67  

68 msrVar(ctCrisis,TheCrisis),  

69 msrVar(dtCrisisID,AdtCrisisID),  

70  

71 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  

72 msrNav([Self],[rnActor],[TheActor]),  

73  

74 /* PostF01 */  

75 msrNav([TheSystem],  

76     [rnctCrisis,  

77      msrSelect,  

78      id,eq,[AdtCrisisID]],  

79      [TheCrisis]),  

80  

81 msrNav([TheCrisis],  

82     [msmAtPost,status],  

83     [[etCrisisStatus,closed]]),  

84  

85 /* PostF02 */  

86 msrNav([TheCrisis],  

87     [msmAtPost,rnHandler],  

88     []),  

89  

90 /* PostF03 */  

91 msrNav([TheCrisis],  

92     [rnAlerts,msrForAll,msrIsKilled],  

93     [[ptBoolean,true]]),  

94  

95 /* PostF04 */  

96 msrNav([TheActor],  

97     [rnInterfaceIN,  

98      ieMessage,[[ptString,'The crisis is now closed !']]  

99      ],  

100     [[ptBoolean,true]]),  

101  

102 /* Post Protocol: */  

103 /* PostP01 */  

104 true  

105 .

```

Listing C.8: Prolog file outactCoordinator-oeCloseCrisis.pl.

## C.9 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCoordinator-oeGetAlertsSet.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */  

3:- multifile msrop/4.  

4%%%%%%%%%%%%%%%
5-----  

6msrop(outactCoordinator,  

7    oeGetAlertsSet,  

8    [preProtocol,Self,  

9     AetAlertStatus  

10    ],  

11    []):-  

12 /* Pre Protocol: */  

13 msrVar(ctState,TheSystem),  

14 msrVar(actCoordinator,TheActor),  

15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  

16 msrNav([Self],[rnActor],[TheActor]),  

17  

18 /* PreP01 */

```

## C.10. FILE /SRC-GEN/PROLOG-REF-SPEC/OPERATIONS.../OUTACTCOORDINATOR-OEGETCRISISSET

```

19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]]))
26 .
27
28msrop(outactCoordinator,
29     oeGetAlertsSet,
30     [preFunctional,Self,
31     AetAlertStatus
32     ],
33     []):-!
34 /* Pre Functional:*/
35 /* PreF01 */
36 true
37 .
38
39msrop(outactCoordinator,
40     oeGetAlertsSet,
41     [post,Self,
42     AetAlertStatus
43     ],
44     []):-!
45
46 /* Post Functional:*/
47 msrVar(ctState,TheSystem),
48 msrVar(actCoordinator,TheActor),
49 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
50 msrNav([Self],[rnActor],[TheActor]),
51
52 /* PostF01 */
53 msrNav([TheSystem],
54     [rnctAlert,
55     msrSelect,
56     status,etEq,[AetAlertStatus]],
57     ColAlertSet),
58
59 msrNav(ColAlertSet,
60     [msrForAll,isSentToCoordinator,[TheActor]],
61     [[ptBoolean,true]]),
62
63 /* Post Protocol:*/
64 /* PostP01 */
65 true
66 .

```

Listing C.9: Prolog file outactCoordinator-oeGetAlertsSet.pl.

## C.10 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeGetCrisisSet.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7     oeGetCrisisSet,
8     [preProtocol,Self,
9     AetCrisisStatus
10    ],
11    []):-!
12 /* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),

```

```

15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18 /* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]])
26 .
27
28msrop(outactCoordinator,
29 oeGetCrisisSet,
30 [preFunctional,Self,
31 AetCrisisStatus
32 ],
33 []):-!
34 /* Pre Functional:*/
35 /* PreF01 */
36 true
37 .
38
39msrop(outactCoordinator,
40 oeGetCrisisSet,
41 [post,Self,
42 AetCrisisStatus
43 ],
44 []):-!
45
46 /* Post Functional:*/
47 msrVar(ctState,TheSystem),
48 msrVar(actCoordinator,TheActor),
49 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
50 msrNav([Self],[rnActor],[TheActor]),
51
52 /* PostF01 */
53 msrNav([TheSystem],
54     [rnctCrisis,
55      msrSelect,
56      status,etEq,[AetCrisisStatus]],
57     ColCrisisSet),
58
59 msrNav(ColCrisisSet,
60     [msrForAll,isSentToCoordinator,[TheActor]],
61     [[ptBoolean,true]]),
62
63 /* Post Protocol:*/
64 /* PostP01 */
65 true
66 .

```

Listing C.10: Prolog file outactCoordinator-oeGetCrisisSet.pl.

## C.11 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outac oeInvalidateAlert.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5%-----
6msrop(outactCoordinator,
7    oeInvalidateAlert,
8    [preProtocol,Self,
9     AdtAlertID
10    ],

```

## C.11. FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTCOORDINATOR-OEINVALIDATEALERT.PL273

```
11      []):-  
12 /* Pre Protocol:- */  
13 msrVar(ctState,TheSystem),  
14 msrVar(actCoordinator,TheActor),  
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  
16 msrNav([Self],[rnActor],[TheActor]),  
17  
18 /* PreP01 */  
19 msrNav([TheSystem],  
20     [vpStarted],  
21     [[ptBoolean,true]]),  
22  
23 /* PreP02 */  
24 msrNav([TheActor],  
25     [rnctAuthenticated,vpIsLogged],  
26     [[ptBoolean,true]])  
27.  
28  
29 msrop(outactCoordinator,  
30 oeInvalidateAlert,  
31 [preFunctional,Self,  
32 AdtAlertID  
33 ],  
34 []):-  
35 /* Pre Functional:- */  
36 msrVar(ctState,TheSystem),  
37 msrVar(actCoordinator,TheActor),  
38  
39 msrVar(dtAlertID,AdtAlertID),  
40  
41 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  
42 msrNav([Self],[rnActor],[TheActor]),  
43  
44 /* PreF01 */  
45 msrNav([TheSystem],  
46     [rnctAlert,  
47     msrSelect,  
48     id,eq,[AdtAlertID]  
49     ],  
50     ColAlert),  
51  
52 msrNav(ColAlert,  
53     [msrSize,eq,[[ptInteger,1]]],  
54     [[ptBoolean,true]])  
55 .  
56  
57 msrop(outactCoordinator,  
58 oeInvalidateAlert,  
59 [post,Self,  
60 AdtAlertID  
61 ],  
62 []):-  
63  
64 /* Post Functional:- */  
65 msrVar(ctState,TheSystem),  
66 msrVar(actCoordinator,TheActor),  
67  
68 msrVar(ctAlert,TheAlert),  
69 msrVar(dtAlertID,AdtAlertID),  
70  
71 msrNav([Self],[rnActor,rnSystem],[TheSystem]),  
72 msrNav([Self],[rnActor],[TheActor]),  
73  
74 /* PostF01 */  
75 msrNav([TheSystem],  
76     [rnctAlert,  
77     msrSelect,  
78     id,eq,[AdtAlertID]],  
79     [TheAlert]),  
80
```

```

81 msrNav( [TheAlert],
82     [ msmAtPost,status],
83     [[etAlertStatus,invalid]]),
84
85 /* PostF02 */
86 msrNav( [TheActor],
87     [ rnInterfaceIN,
88     ieMessage,[[ptString, 'The alert is now declared as invalid !']])
89 ],
90     [[ptBoolean,true]]),
91
92 /* Post Protocol:*/
93 /* PostP01 */
94 true
95 .

```

Listing C.11: Prolog file outactCoordinator-oeInvalidateAlert.pl.

## C.12 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCoordinator-oeReportOnCrisis.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeReportOnCrisis,
8    [preProtocol,Self,
9     AdtCrisisID,
10    AdtComment
11    ],
12    []):-!
13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actCoordinator,TheActor),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18
19/* PreP01 */
20 msrNav([TheSystem],
21     [vpStarted],
22     [[ptBoolean,true]]),
23
24 msrNav([TheActor],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,true]]))
27.
28
29msrop(outactCoordinator,
30    oeReportOnCrisis,
31    [preFunctional,Self,
32     AdtCrisisID,
33     AdtComment
34     ],
35    []):-!
36/* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39
40 msrVar(dtCrisisID,AdtCrisisID),
41
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44
45/* PreF01 */
46 msrNav([TheSystem],
47     [rnctCrisis,

```

### C.13. FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTCOORDINATOR-OESETCRISISHANDLER.PL275

```

48     msrSelect,
49     id,eq,[AdtCrisisID]
50   ],
51   ColCrisis),
52
53 msrNav(ColCrisis,
54   [msrSize,eq,[[ptInteger,1]],
55   [[ptBoolean,true]])
56 .
57
58msrop(outactCoordinator,
59   oeReportOnCrisis,
60   [post,Self,
61   AdtCrisisID,
62   AdtComment
63   ],
64   []):-!
65
66/* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(dtComment,AdtComment),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77/* PostP01 */
78 msrNav([TheSystem],
79   [rnctCrisis,
80   msrSelect,
81   id,eq,[AdtCrisisID]],
82   [TheCrisis]),
83
84 msrNav([TheCrisis],
85   [msmAtPost,comment],
86   [AdtComment]),
87
88 msrNav([TheActor],
89   [rnInterfaceIN,
90   ieMessage,[[ptString,'The crisis comment has been updated !']],
91   ],
92   [[ptBoolean,true]]),
93
94/* Post Protocol:*/
95/* PostP01 */
96 true
97 .

```

Listing C.12: Prolog file outactCoordinator-oeReportOnCrisis.pl.

### C.13 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeSetCrisisHandler.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7   oeSetCrisisHandler,
8   [preProtocol,Self,
9   AdtCrisisID
10  ],
11  []):-!
12/* Pre Protocol:*/

```

## 276 APPENDIX C. LISTING OF THE PROLOG FILES REFERENCED IN THE OPERATION MODEL SPE

```

13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
16 msrNav([Self],[rnActor],[TheActor]),
17
18 /* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]])
26 .
27
28 msrop(outactCoordinator,
29     oeSetCrisisHandler,
30     [preFunctional,Self,
31      AdtCrisisID
32      ],
33      []):-!
34 /* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtCrisisID,AdtCrisisID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43 /* PreF01 */
44 msrNav([TheSystem],
45     [rnctCrisis,
46      msrSelect,
47      id,eq,[AdtCrisisID]
48      ],
49      ColCrisis),
50
51 msrNav(ColCrisis,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]])
54 .
55
56 msrop(outactCoordinator,
57     oeSetCrisisHandler,
58     [post,Self,
59      AdtCrisisID
60      ],
61      []):-!
62
63 /* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66 msrVar(ctCoordinator,TheCoordinator),
67 msrVar(ctCoordinator,TheCurrentHandler),
68
69 msrVar(ctCrisis,TheCrisis),
70 msrVar(dtCrisisID,AdtCrisisID),
71
72 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
73 msrNav([Self],[rnActor],[TheActor]),
74
75 /* PostF01 */
76 msrNav([TheSystem],
77     [rnctCrisis,
78      msrSelect,
79      id,eq,[AdtCrisisID]],
80      [TheCrisis]),
81
82 msrNav([TheCrisis],

```

#### C.14. FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTCOORDINATOR-OESETCRISISSTATUS.PL277

```

83      [msmAtPost,status],
84      [[etCrisisStatus,handled]]),
85
86 msrNav([TheActor],
87     [rnctCoordinator],
88     [TheCoordinator]),
89 msrNav([TheCrisis],
90     [msmAtPost,rnHandler],
91     [TheCoordinator]),
92
93 msrNav([TheActor],
94     [rnInterfaceIN,
95     ieMessage,[[ptString,'You are now considered as handling the crisis !']],
96     ],
97     [[ptBoolean,true]]),
98
99 /* PostF02 */
100 msrNav([TheCrisis],
101     [rnAlerts,msrForAll,isSentToCoordinator,[TheActor]],
102     [[ptBoolean,true]]),
103
104 /* PostF03 */
105 ( msrNav([TheCrisis],
106     [rnHandler,msrSize,eq,[[ptInteger,1]]],
107     [[ptBoolean,true]])
108 -> (msrNav([TheCrisis],
109     [rnHandler],
110     [TheCurrentHandler]),
111     msrNav([TheCurrentHandler],
112     [rnactCoordinator,rnInterfaceIN,
113     ieMessage,[[ptString,'One of the crisis you were handling is now handled by one of your
114     colleagues!']]],
115     [[ptBoolean,true]])
116   )
117 ; true
118 ),
119
120 /* PostF04 */
121 msrNav([TheCrisis],
122     [rnAlerts,rnSignaler,msrForAll,isAcknowledged,[],],
123     [[ptBoolean,true]]),
124
125 /* Post Protocol:*/
126/* PostP01 */
127 true
128 .

```

Listing C.13: Prolog file outactCoordinator-oeSetCrisisHandler.pl.

#### C.14 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeSetCrisisStatus.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeSetCrisisStatus,
8    [preProtocol,Self,
9     AdtCrisisID,
10    AetCrisisStatus
11    ],
12    []):-!
13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actCoordinator,TheActor),

```

278 APPENDIX C. LISTING OF THE PROLOG FILES REFERENCED IN THE OPERATION MODEL SPE

```

16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18
19 /* PreP01 */
20 msrNav([TheSystem],
21     [vpStarted],
22     [[ptBoolean,true]]),
23
24 msrNav([TheActor],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,true]])
27 .
28
29 msrop(outactCoordinator,
30     oeSetCrisisStatus,
31     [preFunctional,Self,
32     AdtCrisisID,
33     AetCrisisStatus
34     ],
35     []):-!
36 /* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39
40 msrVar(dtCrisisID,AdtCrisisID),
41
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44
45 /* PreF01 */
46 msrNav([TheSystem],
47     [rnctCrisis,
48     msrSelect,
49     id,eq,[AdtCrisisID]
50     ],
51     ColCrisis),
52
53 msrNav(ColCrisis,
54     [msrSize,eq,[[ptInteger,1]]],
55     [[ptBoolean,true]])
56 .
57
58 msrop(outactCoordinator,
59     oeSetCrisisStatus,
60     [post,Self,
61     AdtCrisisID,
62     AetCrisisStatus
63     ],
64     []):-!
65
66 /* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisStatus,AetCrisisStatus),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77 /* PostF01 */
78 msrNav([TheSystem],
79     [rnctCrisis,
80     msrSelect,
81     id,eq,[AdtCrisisID]],
82     [TheCrisis]),
83
84 msrNav([TheCrisis],
85     [msmAtPost,status],

```

## C.15. FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTCOORDINATOR-OESETCRISISTYPE.PL279

```

86     [AetCrisisStatus]),
87
88 msrNav([TheActor],
89     [rnInterfaceIN,
90      ieMessage,[[ptString,'The crisis status has been updated !']],
91     ],
92     [[ptBoolean,true]]),
93
94 /* Post Protocol:*/
95 /* PostP01 */
96 true
97 .

```

Listing C.14: Prolog file outactCoordinator-oeSetCrisisStatus.pl.

## C.15 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCo oeSetCrisisType.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeSetCrisisType,
8    [preProtocol,Self,
9     AdtCrisisID,
10    AetCrisisType
11    ],
12    []):-!
13/* Pre Protocol:*/
14 msrVar(ctState,TheSystem),
15 msrVar(actCoordinator,TheActor),
16 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
17 msrNav([Self],[rnActor],[TheActor]),
18
19/* PreP01 */
20 msrNav([TheSystem],
21     [vpStarted],
22     [[ptBoolean,true]]),
23
24 msrNav([TheActor],
25     [rnctAuthenticated,vpIsLogged],
26     [[ptBoolean,true]]))
27.
28
29msrop(outactCoordinator,
30    oeSetCrisisType,
31    [preFunctional,Self,
32     AdtCrisisID,
33     AetCrisisType
34     ],
35    []):-!
36/* Pre Functional:*/
37 msrVar(ctState,TheSystem),
38 msrVar(actCoordinator,TheActor),
39
40 msrVar(dtCrisisID,AdtCrisisID),
41
42 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
43 msrNav([Self],[rnActor],[TheActor]),
44
45/* PreF01 */
46 msrNav([TheSystem],
47     [rnctCrisis,
48      msrSelect,
49      id,eq,[AdtCrisisID]
50     ],

```

```

51     ColCrisis),
52
53 msrNav(ColCrisis,
54     [msrSize,eq,[[ptInteger,1]]],
55     [[ptBoolean,true]])
56 .
57
58msrop(outactCoordinator,
59     oeSetCrisisType,
60     [post,Self,
61     AdtCrisisID,
62     AetCrisisType
63     ],
64     []):-.
65
66 /* Post Functional:*/
67 msrVar(ctState,TheSystem),
68 msrVar(actCoordinator,TheActor),
69
70 msrVar(ctCrisis,TheCrisis),
71 msrVar(dtCrisisID,AdtCrisisID),
72 msrVar(etCrisisType,AetCrisisType),
73
74 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
75 msrNav([Self],[rnActor],[TheActor]),
76
77 /* PostF01 */
78 msrNav([TheSystem],
79     [rnctCrisis,
80     msrSelect,
81     id,eq,[AdtCrisisID]],
82     [TheCrisis]),
83
84 msrNav([TheCrisis],
85     [msmAtPost,type],
86     [AetCrisisType]),
87
88 msrNav([TheActor],
89     [rnInterfaceIN,
90     ieMessage,[[ptString,'The crisis type has been updated !']]],
91     ),
92     [[ptBoolean,true]]),
93
94 /* Post Protocol:*/
95 /* PostP01 */
96 true
97 .

```

Listing C.15: Prolog file outactCoordinator-oeSetCrisisType.pl.

## C.16 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCoordinator-oeValidateAlert.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTINUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5-----
6msrop(outactCoordinator,
7    oeValidateAlert,
8    [preProtocol,Self,
9     AdtAlertID
10    ],
11    []):-.
12/* Pre Protocol:*/
13 msrVar(ctState,TheSystem),
14 msrVar(actCoordinator,TheActor),
15 msrNav([Self],[rnActor,rnSystem],[TheSystem]),

```

C.16. FILE /SRC-GEN/PROLOG-REF-SPEC.../OUTACTCOORDINATOR-OEVALIDATEALERT.PL281

```
16 msrNav([Self],[rnActor],[TheActor]),
17
18/* PreP01 */
19 msrNav([TheSystem],
20     [vpStarted],
21     [[ptBoolean,true]]),
22
23 msrNav([TheActor],
24     [rnctAuthenticated,vpIsLogged],
25     [[ptBoolean,true]])
26.
27
28msrop(outactCoordinator,
29    oeValidateAlert,
30    [preFunctional,Self,
31     AdtAlertID
32     ],
33     []):-!
34/* Pre Functional:*/
35 msrVar(ctState,TheSystem),
36 msrVar(actCoordinator,TheActor),
37
38 msrVar(dtAlertID,AdtAlertID),
39
40 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
41 msrNav([Self],[rnActor],[TheActor]),
42
43/* PreF01 */
44 msrNav([TheSystem],
45     [rnctAlert,
46      msrSelect,
47      id,eq,[AdtAlertID]
48      ],
49     ColAlerts),
50
51 msrNav(ColAlerts,
52     [msrSize,eq,[[ptInteger,1]]],
53     [[ptBoolean,true]]),
54 .
55
56msrop(outactCoordinator,
57    oeValidateAlert,
58    [post,Self,
59     AdtAlertID
60     ],
61     []):-!
62
63/* Post Functional:*/
64 msrVar(ctState,TheSystem),
65 msrVar(actCoordinator,TheActor),
66
67 msrVar(ctAlert,TheAlert),
68 msrVar(dtAlertID,AdtAlertID),
69
70 msrNav([Self],[rnActor,rnSystem],[TheSystem]),
71 msrNav([Self],[rnActor],[TheActor]),
72
73/* PostF01 */
74 msrNav([TheSystem],
75     [rnctAlert,
76      msrSelect,
77      id,eq,[AdtAlertID]],
78     [TheAlert]),
79
80 msrNav([TheAlert],
81     [msmAtPost,status],
82     [[etAlertStatus,valid]]),
83
84 msrNav([TheActor],
85     [rnInterfaceIN,
```

```

86      ieMessage,[[ptString,'The Alert is now declared as valid !']]
87    ],
88    [[ptBoolean,true]]),
89
90  /* Post Protocol:*/
91  /* PostP01 */
92  true
93 .

```

Listing C.16: Prolog file outactCoordinator-oeValidateAlert.pl.

### C.17 File ./src-gen/prolog-ref-spec/Operations/Environment/OUT/outactCoordinator-oeCreateSystemAndEnvironment.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5/*
6*****
7MSRCreatorActor
8*****
9
10/*** createSystemAndEnvironment ***/
11
12msrop(outactMsrCreator,
13      oeCreateSystemAndEnvironment,
14      [preFunctional,_Self,_AqtyComCompanies],
15      []):-_
16  true.
17
18msrop(outactMsrCreator,
19      oeCreateSystemAndEnvironment,
20      [preProtocol,_Self,_AqtyComCompanies],
21      []):-_
22  true.
23
24msrop(outactMsrCreator,
25      oeCreateSystemAndEnvironment,
26      [post,_Self,AqtyComCompanies],
27      []):-_
28
29 msrVar(ctState,TheSystem),
30 msrVar(actMsrCreator,AactMsrCreator),
31 msrVar(actAdministrator,AactAdministrator),
32
33 msrVar(dtInteger, AnextValueForAlertID),
34 msrVar(dtInteger, AnextValueForCrisisID),
35 msrVar(dtDateAndTime, Aclock),
36 msrVar(dtSecond, AcrisisReminderPeriod),
37 msrVar(dtSecond, AmaxCrisisReminderPeriod),
38 msrVar(ptBoolean, AvpStarted),
39
40 /* PostF01 -- MUST ALWAYS BE MADE FIRST -- */
41 msrNav([AnextValueForAlertID],
42        [value,eq,[[ptInteger,1]]],
43        [[ptBoolean,true]]),
44
45 msrNav([AnextValueForCrisisID],
46        [value,eq,[[ptInteger,1]]],
47        [[ptBoolean,true]]),
48
49msrNav([Aclock],
50       [date,year,value],
51       [[ptInteger,1970]]),
52msrNav([Aclock],
53       [date,month,value],
54       [[ptInteger,01]]),

```

C.17. FILE /SRC-GEN.../OUTACTMSRCREATOR-OECREATESYSTEMANDENVIRONMENT.PL283

```
55msrNav([Aclock],
56    [date,day,value],
57    [[ptInteger,01]]),
58
59msrNav([Aclock],
60    [time,hour,value],
61    [[ptInteger,00]]),
62msrNav([Aclock],
63    [time,minute,value],
64    [[ptInteger,00]]),
65msrNav([Aclock],
66    [time,second,value],
67    [[ptInteger,00]]),
68
69 msrNav([AcrisisReminderPeriod],
70    [value,eq,[[ptInteger,300]]],
71    [[ptBoolean,true]]),
72
73 msrNav([AmaxCrisisReminderPeriod],
74    [value,eq,[[ptInteger,1200]]],
75    [[ptBoolean,true]]),
76
77 msrNav([AvpStarted],
78    [],
79    [[ptBoolean,true]]),
80
81 msrNav([TheSystem],
82    [init,[AnextValueForAlertID,
83        AnextValueForCrisisID,
84        Aclock,
85        AcrisisReminderPeriod,
86        AmaxCrisisReminderPeriod,
87        Aclock,
88        AvpStarted]
89        ],
90    [[ptBoolean,true]]),
91
92/* PostF02*/
93 msrNav([AactMsrCreator],
94    [init,[],],
95    [[ptBoolean,true]]),
96
97 /* PostF03 */
98 msrVarCol(actComCompany,AqtyComCompanies,AactComCompanyCol),
99
100 msrNav(AactComCompanyCol,
101    [msrForAll,init,[],],
102    [[ptBoolean,true]]),
103
104 /* PostF04*/
105 msrNav([AactAdministrator],
106    [init,[],],
107    [[ptBoolean,true]]),
108
109 /* PostF05*/
110 msrVar(actActivator,AactActivator),
111 msrNav([AactActivator],
112    [init,[],],
113    [[ptBoolean,true]]),
114
115/* PostF06 */
116 msrVar(ctAdministrator,ActAdministrator),
117 msrVar(dtLogin,AdtLogin),
118 msrVar(dtPassword,AdtPassword),
119
120 msrNav([AdtLogin],
121    [value,eq,[[ptString,'icrashadmin']]],
122    [[ptBoolean,true]]),
123
124 msrNav([AdtPassword],
```

```

125      [value,eq,[[ptString,'7WXC1359']]],  

126      [[ptBoolean,true]]),  

127  

128 msrNav([ActAdministrator],  

129     [init,[AdtLogin,AdtPassword]],  

130     [[ptBoolean,true]]),  

131  

132 /* PostF07 */  

133 msrNav([ActAdministrator],  

134     [msmAtPost,rnactAuthenticated],  

135     [AactAdministrator]),  

136  

137 /* Post Protocol: */  

138 /* PostP01 */  

139 true  

140 .

```

Listing C.17: Prolog file outactMsrCreator-oeCreateSystemAndEnvironment.pl.

## C.18 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctAdministrator-init.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6msrop(ctAdministrator,init,[Self,
7           Alogin,
8           Apwd],
9       Result):-  

10 (
11msrVar(ctAdministrator,Self),
12
13/* Post F01 */
14msrNav([Self],[login],[Alogin]),
15msrNav([Self],[pwd],[Apwd]),
16msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),
17
18/* Post F02 */
19msrNav([Self],[msrIsNew],[Self])
20)
21-> Result = [ptBoolean,true]
22; Result = [ptBoolean,false]
23.

```

Listing C.18: Prolog file PrimaryTypesClasses-ctAdministrator-init.pl.

## C.19 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctAlert-init.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6msrop(ctAlert,init,[Self,
7           Aid,
8           Astatus,
9           Alocation,
10          Ainstant,
11          Acomment],
12       Result):-  

13
14/* Post F01 */
15(

```

## C.20. FILE /SRC-GEN.../PRIMARYTYPESCLASSES-CTALERT-ISSENTTOCOORDINATOR.PL285

```
16msrVar(ctAlert,Self),
17
18msrNav([Self],[id],[Aid]),
19msrNav([Self],[status],[Astatus]),
20msrNav([Self],[location],[Alocation]),
21msrNav([Self],[instant],[Ainstant]),
22msrNav([Self],[comment],[Acomment]),
23
24/* Post F02 */
25 msrNav([Self],[msrIsNew],[Self])
26)
27-> Result = [ptBoolean,true]
28; Result = [ptBoolean,false]
29.
```

Listing C.19: Prolog file PrimaryTypesClasses-ctAlert-init.pl.

## C.20 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl

```
1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctAlert,isSentToCoordinator,[Self,AactCoordinator],
7      Result):-_
8
9/* Post F01 */
10(
11 msrNav([AactCoordinator],
12       [rnInterfaceIN,ieSendAnAlert,[Self]],
13       [[ptBoolean,true]])
14)
15-> Result = [ptBoolean,true]
16; Result = [ptBoolean,false]
17.
```

Listing C.20: Prolog file PrimaryTypesClasses-ctAlert-isSentToCoordinator.pl.

## C.21 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctAuthenticated-init.pl

```
1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctAuthenticated,init,[Self,
7      Alogin,
8      Apwd],
9      Result):-_
10
11/* Post F01 */
12(
13msrVar(ctAuthenticated,Self),
14
15msrNav([Self],[login],[Alogin]),
16msrNav([Self],[pwd],[Apwd]),
17msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),
18
19/* Post F02 */
20 msrNav([Self],[msrIsNew],[Self])
21)
22-> Result = [ptBoolean,true]
23; Result = [ptBoolean,false]
```

24.

Listing C.21: Prolog file PrimaryTypesClasses-ctAuthenticated-init.pl.

## C.22 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctCoordinator-init.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCoordinator,init,[Self,
7      Aid,
8      Alogin,
9      Apwd],
10     Result):- 
11
12/* Post F01 */
13(
14msrVar(ctCoordinator,Self),
15
16msrNav([Self],[id],[Aid]),
17msrNav([Self],[login],[Alogin]),
18msrNav([Self],[pwd],[Apwd]),
19msrNav([Self],[vpIsLogged],[[ptBoolean,false]]),
20
21/* Post F02 */
22 msrNav([Self],[msrIsNew],[Self])
23)
24-> Result = [ptBoolean,true]
25; Result = [ptBoolean,false]
26.

```

Listing C.22: Prolog file PrimaryTypesClasses-ctCoordinator-init.pl.

## C.23 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctCrisis-handlingDelayPassed.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,handlingDelayPassed,[Self],
7     Result):-
8
9/* Post F01 */
10(
11 msrVar(ctState,TheSystem),
12 msrVar(dtInteger,CurrentClockSecondsQty),
13 msrVar(dtInteger,LastReminderSecondsQty),
14 msrVar(dtSecond,CrisisReminderPeriod),
15
16 msrNav([Self],[rnSystem],[TheSystem]),
17
18 msrNav([Self],
19      [status],
20      [[etCrisisStatus,pending]]),
21
22 msrNav([TheSystem],
23      [clock,toSecondsQty,[]],
24      [CurrentClockSecondsQty]),
25
26 msrNav([TheSystem],
27      [vpLastReminder,toSecondsQty,[]],

```

## C.24. FILE /SRC-GEN/PROLOG-REF-SPEC.../PRIMARYTYPESCLASSES-CTCRISIS-INIT.PL287

```

28     [LastReminderSecondsQty]),
29
30 msrNav([TheSystem],
31     [crisisReminderPeriod],
32     [CrisisReminderPeriod]),
33
34 msrNav([CurrentClockSecondsQty],
35     [sub,[LastReminderSecondsQty],
36         gt, [CrisisReminderPeriod]
37         ],
38     [[ptBoolean,true]])
39
40)
41-> Result = [ptBoolean,true]
42; Result = [ptBoolean,false]
43.
```

Listing C.23: Prolog file PrimaryTypesClasses-ctCrisis-handlingDelayPassed.pl.

## C.24 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctCrisis-init.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6msrop(ctCrisis,init,[Self,
7        Aid,
8        Atype,
9        Astatus,
10       Alocation,
11       Ainstant,
12       Acomment],
13   Result):-!
14
15/* Post F01 */
16(
17msrVar(ctCrisis,Self),
18
19msrNav([Self],[id],[Aid]),
20msrNav([Self],[type],[Atype]),
21msrNav([Self],[status],[Astatus]),
22msrNav([Self],[location],[Alocation]),
23msrNav([Self],[instant],[Ainstant]),
24msrNav([Self],[comment],[Acomment]),
25
26/* Post F02 */
27 msrNav([Self],[msrIsNew],[Self])
28)
29-> Result = [ptBoolean,true]
30; Result = [ptBoolean,false]
31.
```

Listing C.24: Prolog file PrimaryTypesClasses-ctCrisis-init.pl.

## C.25 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6msrop(ctCrisis,isAllocatedIfPossible,[Self],
7   Result):-!
```

```

8(
9  msrVar(ctState,TheSystem),
10 msrNav([Self],[rnSystem],[TheSystem]),
11
12 msrVar(actCoordinator,TheCoordinatorActor),
13 msrVar(ctCoordinator,TheCoordinator),
14 msrVar(ptString,TheMessage),
15 msrVar(ptString,TheCrisisIDptString),
16
17 (
18  /* Post F01 */
19 msrNav([Self],
20        [maxHandlingDelayPassed,[],[[ptBoolean,true]]]),
21
22 ( msrNav([TheSystem],
23          [rnactCoordinator,msrIsEmpty],
24          [[ptBoolean,false]]))
25 -> (
26
27  /* Post F02 */
28  msrNav([TheSystem],
29         [rnactCoordinator,msrAny,msrTrue],
30         [TheCoordinatorActor]),
31
32  msrNav([TheCoordinatorActor],
33         [rnctCoordinator],
34         [TheCoordinator]),
35
36  msrNav([Self],
37         [msmAtPost,rnHandler],
38         [TheCoordinator]),
39
40  msrNav([Self],
41         [msmAtPost,status],
42         [[etCrisisStatus,handled]]),
43
44  msrNav([Self],
45         [id,value],
46         [TheCrisisIDptString]),
47
48  msrNav([[ptString,'You are now considered as handling the crisis having ID: ']],
49          [ptStringConcat,[TheCrisisIDptString]],
50          [TheMessage]),
51
52  msrNav([TheCoordinatorActor],
53         [rnInterfaceIN,
54          ieMessage,[TheMessage]
55         ],
56         [[ptBoolean,true]]))
57 )
58 ; ( /* Post F03 */
59  msrNav([TheSystem],
60         [rnactAdministrator,msrForAll,rnInterfaceIN,
61          ieMessage,[[ptString,'Please add new coordinators to handle pending crisis !']]],
62          [[ptBoolean,true]]))
63 )
64 )
65 )
66 )
67-> Result = [ptBoolean,true]
68 ; Result = [ptBoolean,false]
69 .

```

Listing C.25: Prolog file PrimaryTypesClasses-ctCrisis-isAllocatedIfPossible.pl.

## C.26 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC ctCrisis-isSentToCoordinator.pl

## C.27. FILE /SRC-GEN.../PRIMARYTYPESCLASSES-CTCRISIS-MAXHANDLINGDELAYPASSED.PL289

```
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,isSentToCoordinator,[Self,AactCoordinator],
7      Result):-  

8
9/* Post F01 */
10(
11 msrNav([AactCoordinator],
12         [rnInterfaceIN,ieSendACrisis,[Self]],
13         [[ptBoolean,true]])
14)
15-> Result = [ptBoolean,true]
16; Result = [ptBoolean,false]
17.
```

Listing C.26: Prolog file PrimaryTypesClasses-ctCrisis-isSentToCoordinator.pl.

## C.27 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl

```
1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(ctCrisis,maxHandlingDelayPassed,[Self],
7      Result):-  

8
9/* Post F01 */
10(
11 msrVar(ctState,TheSystem),
12 msrVar(dtInteger,CurrentClockSecondsQty),
13 msrVar(dtInteger,CrisisInstantSecondsQty),
14 msrVar(dtSecond,MaxCrisisReminderPeriod),
15
16 msrNav([Self],[rnSystem],[TheSystem]),
17
18 msrNav([Self],
19         [status],
20         [[etCrisisStatus,pending]]),
21
22 msrNav([TheSystem],
23         [clock,toSecondsQty,[]],
24         [CurrentClockSecondsQty]),
25
26 msrNav([Self],
27         [instant,toSecondsQty,[]],
28         [CrisisInstantSecondsQty]),
29
30 msrNav([TheSystem],
31         [maxCrisisReminderPeriod],
32         [MaxCrisisReminderPeriod]),
33
34 msrNav([CurrentClockSecondsQty],
35         [sub,[CrisisInstantSecondsQty],
36          gt, [MaxCrisisReminderPeriod]
37          ],
38         [[ptBoolean,true]])
39
40)
41-> Result = [ptBoolean,true]
42; Result = [ptBoolean,false]
43.
```

Listing C.27: Prolog file PrimaryTypesClasses-ctCrisis-maxHandlingDelayPassed.pl.

### C.28 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctHuman-init.pl

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 /* DISCONTIGUOUS PREDICATES */
3 :- multifile msrop/4.
4 %%%%%%%%%%%%%%
5
6 msrop(ctHuman,init,[Self,
7     Aid,
8     Akind],
9     Result):-
10
11 /* Post F01 */
12 (
13 msrVar(ctHuman,Self),
14
15 msrNav([Self],[id],[Aid]),
16 msrNav([Self],[kind],[Akind]),
17
18 /* Post F02 */
19 msrNav([Self],[msrIsNew],[Self])
20 )
21 -> Result = [ptBoolean,true]
22 ; Result = [ptBoolean,false]
23 .

```

Listing C.28: Prolog file PrimaryTypesClasses-ctHuman-init.pl.

### C.29 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctHuman-isAcknowledged.pl

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 /* DISCONTIGUOUS PREDICATES */
3 :- multifile msrop/4.
4 %%%%%%%%%%%%%%
5
6 msrop(ctHuman,isAcknowledged,[Self],Result):-
7
8 /* Post F01 */
9 (msrVar(dtPhoneNumber,AdtPhoneNumber),
10 msrVar(dtSMS,AdtSMS),
11
12 msrNav([Self],
13     [id,eq,[AdtPhoneNumber]],
14     [[ptBoolean,true]]),
15 msrNav([AdtSMS],
16     [value,eq,[[ptString,'The handling of your alert by our services is in progress !']]],
17     [[ptBoolean,true]]),
18 msrNav([Self],
19     [rnactComCompany,rnInterfaceIN,ieSmsSend,[AdtPhoneNumber,AdtSMS]],
20     [[ptBoolean,true]]),
21 )
22 -> Result = [ptBoolean,true]
23 ; Result = [ptBoolean,false]
24 .

```

Listing C.29: Prolog file PrimaryTypesClasses-ctHuman-isAcknowledged.pl.

### C.30 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClasses-ctState-init.pl

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 /* DISCONTIGUOUS PREDICATES */
3 :- multifile msrop/4.

```

C.31. FILE /SRC-GEN/PROLOG-REF-SPEC.../PRIMARYTYPESDATATYPES-DTALERTID-IS.PL291

```

1%%%%%%%%%%%%%
2
3
4
5
6msrop(ctState,init,[Self,
7          AnextValueForAlertID,
8          AnextValueForCrisisID,
9          Aclock,
10         AcrisisReminderPeriod,
11         AmaxCrisisReminderPeriod,
12         AvpLastReminder,
13         AvpStarted],
14     Result):- 
15
16 /* Post F01 */
17(
18 msrVar(ctState,Self),
19
20 msrNav([Self],[nextValueForAlertID],[AnextValueForAlertID]),
21 msrNav([Self],[nextValueForCrisisID],[AnextValueForCrisisID]),
22 msrNav([Self],[clock],[Aclock]),
23 msrNav([Self],[crisisReminderPeriod],[AcrisisReminderPeriod]),
24 msrNav([Self],[maxCrisisReminderPeriod],[AmaxCrisisReminderPeriod]),
25 msrNav([Self],[vpLastReminder],[AvpLastReminder]),
26 msrNav([Self],[vpStarted],[AvpStarted]),
27
28 msrNav([Self],[msrIsNew],[Self])
29)
30-> Result = [ptBoolean,true]
31; Result = [ptBoolean,false]
32.

```

Listing C.30: Prolog file PrimaryTypesClasses-ctState-init.pl.

C.31 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData  
dtAlertID-is.pl

```

1%-----%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%-----%
5
6msrop(dtAlertID,is,[AdtValue],Result):-%
7% msd01
8msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,length,[],gt,[[ptInteger,0]]],
12   [[ptBoolean,true]]),
13   msrNav([AdtValue],
14   [value,length,[],leq,[[ptInteger,20]]],
15   [[ptBoolean,true]])
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20 TheResult = Result
21 .
22
23/*
24| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[[]]]],%
25msrNav([X],[is,[],[Result]]).
26
27X = [dtAlertID,[],[[dtString,[[value,[ptString,'0123456789']]],[[]]]],%
28Result = [ptBoolean,true] ?
29
30yes
31
32| ?- X = [dtAlertID,[],[[dtString,[[value,[ptString,'012345678901234567890123456789']]],[[]]]],%
33msrNav([X],[is,[],[Result]]).
```

Listing C.31: Prolog file PrimaryTypesDatatypes-dtAlertID-is.pl.

C.32 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesDtComment-is.pl

```

1%{
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%{
5%% dtComment
6
7%msd01
8msrop(dtComment,is,[AdtValue],Result):-%
9 msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MaxLength),
11 (
12 (
13 (
14     MaxLength = [ptInteger,160],
15     msrNav([AdtValue],
16         [value,length,[],leq,[MaxLength]],
17         [[ptBoolean,true]])
18 )
19     -> TheResult = [ptBoolean,true]
20     ; TheResult = [ptBoolean,false]
21 )
22),
23 Result = TheResult
24 .
25
26/*
27 | ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[[]]]],%
28msrNav([X],[is,[],[Result]]).
29X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg ! Please help ...']]],[[]]]],%
30Result = [ptBoolean,true] ?
31yes
32
33| ?- X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog
            to go to the skate park because my friends called me on my mobile phone and told me that a skate
            star was doing triple back flips.']]],[[]]]],%
34msrNav([X],[is,[],[Result]]).
35X = [dtComment,[],[[dtString,[[value,[ptString,'I broke my leg when I was running with my dog to go
            to the skate park because my friends called me on my mobile phone and told me that a skate star
            was doing triple back flips.']]],[[]]]],%
36Result = [ptBoolean,false] ?
37yes
38*/

```

Listing C.32: Prolog file PrimaryTypesDatatypes-dtComment-is.pl.

C.33 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesDtCoordinatorID-is.pl

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 /* DISCONTIGUOUS PREDICATES */
3 :- multifile msrop/4.
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6 msrop(dtCoordinatorID,js,[AdtValue],Result):-
```

C.34. FILE /SRC-GEN/PROLOG-REF-SPEC.../PRIMARYTYPESDATATYPES-DTCRISISID-IS.PL293

```
7% msd01
8 msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,length,[],gt,[[ptInteger,0]]]),
12   [[ptBoolean,true]]),
13 msrNav([AdtValue],
14   [value,length,[],leq,[[ptInteger,5]]]),
15   [[ptBoolean,true]]))
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20 TheResult = Result
21 .
```

Listing C.33: Prolog file PrimaryTypesDatatypes-dtCoordinatorID-is.pl.

C.34 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData  
dtCrisisID-is.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6msrop(dtCrisisID,is,[AdtValue],Result):-%
7% msd01
8 msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,length,[],gt,[[ptInteger,0]]],
12   [[ptBoolean,true]]),
13 msrNav([AdtValue],
14   [value,length,[],leq,[[ptInteger,10]]],
15   [[ptBoolean,true]])
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19 ),
20 TheResult = Result
21 .
22/*
23| ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]]],[],[]]].
24msrNav([X],[is,[],[Result]).
25X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789']]]],[],[]]],
26Result = [ptBoolean,true] ?
27yes
28
29| ?- X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]]],[],[]]].
30msrNav([X],[is,[],[Result]).
31X = [dtCrisisID,[],[[dtString,[[value,[ptString,'0123456789a']]]],[],[]]],
32Result = [ptBoolean,false] ?
33yes
34*/

```

Listing C.34: Prolog file PrimaryTypesDatatypes-dtCrisisID-is.pl.

C.35 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData  
dtGPSLocation-is.pl

```

6%% dtPhoneNumber
7
8% msd01
9msrop(dtGPSLocation,is,[AdtValue],Result):-
10msrVar(ptBoolean,TheResult),
11(
12  (
13    msrNav([AdtValue],
14      [latitude,is,[]],
15      [[ptBoolean,true]]),
16    msrNav([AdtValue],
17      [longitude,is,[]],
18      [[ptBoolean,true]])
19  )
20 -> TheResult = [ptBoolean,true]
21 ; TheResult = [ptBoolean,false]
22),
23
24 Result = TheResult
25.

```

Listing C.35: Prolog file PrimaryTypesDatatypes-dtGPSLocation-is.pl.

### C.36 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesDtGPSLocation-isNearTo.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% dtGPSLocation
7
8msrop(dtGPSLocation,isNearTo,[Self,AdtValue],Result):-
9msrVar(ptBoolean,TheResult),
10msrVar(dtReal,EarthRadius),
11msrVar(dtReal,MaxDistance),
12
13msrVar(dtLatitude,ComparedLatitude),
14msrVar(dtLongitude,ComparedLongitude),
15
16msrVar(dtReal,R1),msrVar(dtReal,R1a),
17msrVar(dtReal,R2),msrVar(dtReal,R2a),
18
19(
20  (
21    (
22      % msd01
23      msrNav([EarthRadius],[value],[[ptReal,6371]]),
24      msrNav([MaxDistance],[value],[[ptReal,100]]),
25
26      msrNav([AdtValue],[latitude],[ComparedLatitude]),
27      msrNav([AdtValue],[longitude],[ComparedLongitude]),
28
29      msrNav([Self],[latitude,sin,[],[R1a]]),
30      msrNav([AdtValue],[latitude,sin,[],mul,[R1a]],[],[R1]),
31
32      msrNav([Self],[latitude,cos,[],[R2a]]),
33      msrNav([AdtValue],[latitude,cos,[],mul,[R2a]],[],[R2]),
34
35      msrNav([AdtValue],[longitude],[ComparedLongitude]),
36      msrNav([Self],[longitude,sub,[ComparedLongitude],cos,[],mul,[R2],
37        add,[R1],
38        acos,[]],
39        mul,[EarthRadius],
40        sub,[MaxDistance],
41        value,leq,[[ptReal,0]]]),
42        [[ptBoolean,true]])

```

### C.37 FILE /SRC-GEN/PROLOG-REF-SPEC.../PRIMARYTYPESDATATYPES-DTLATITUDE-IS.PL295

```
43      )
44      -> TheResult = [ptBoolean,true]
45      ; TheResult = [ptBoolean,false]
46    )
47),
48 Result = TheResult
49.
```

Listing C.36: Prolog file PrimaryTypesDatatypes-dtGPSLocation-isNearTo.pl.

### C.37 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtLatitude-is.pl

```
1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6% msd01
7msrop(dtLatitude,is,[AdtValue],Result):-%
8msrVar(ptBoolean,TheResult),
9(
10 ( msrNav([AdtValue],
11   [value,geq,[[ptReal,-90.0]]],
12   [[ptBoolean,true]]),
13   msrNav([AdtValue],
14   [value,leq,[[ptReal,+90.0]]],
15   [[ptBoolean,true]])
16 )
17 -> (TheResult = [ptBoolean,true])
18 ; (TheResult = [ptBoolean,false])
19),
20Result = TheResult
21.
```

Listing C.37: Prolog file PrimaryTypesDatatypes-dtLatitude-is.pl.

### C.38 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtLogin-is.pl

```
1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5% dtComment
6
7msd01
8msrop(dtLogin,is,[AdtValue],Result):-
9 msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MaxLength),
11(
12 (
13 (
14   MaxLength = [ptInteger,20],
15   msrNav([AdtValue],
16   [value,length,[],leq,[MaxLength]],
17   [[ptBoolean,true]])
18 )
19 -> TheResult = [ptBoolean,true]
20 ; TheResult = [ptBoolean,false]
21 )
22),
23 Result = TheResult
24.
25/*
26| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[[]]]],
```

```

27msrNav([X],[is,[],[Result]).  

28X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567']]],[[]]]],  

29Result = [ptBoolean,true] ?  

30yes  

31  

32| ?- X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[[]]]],  

33msrNav([X],[is,[],[Result]).  

34X = [dtLogin,[],[[dtString,[[value,[ptString,'01234567a']]],[[]]]],  

35Result = [ptBoolean,false] ?  

36yes  

37*/

```

Listing C.38: Prolog file PrimaryTypesDatatypes-dtLogin-is.pl.

### C.39 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesDtLongitude-is.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% dtPhoneNumber
7
8% msd01
9msrop(dtLongitude,is,[AdtValue],Result):-  

10msrVar(ptBoolean,TheResult),  

11(  

12( msrNav([AdtValue],  

13    [value,geq,[[ptReal,-180.0]]],  

14    [[ptBoolean,true]]),  

15  msrNav([AdtValue],  

16    [value,leq,[[ptReal,+180.0]]],  

17    [[ptBoolean,true]]))  

18)
19 -> (TheResult = [ptBoolean,true])
20 ; (TheResult = [ptBoolean,false])
21),
22
23 Result = TheResult
24.

```

Listing C.39: Prolog file PrimaryTypesDatatypes-dtLongitude-is.pl.

### C.40 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesDtPassword-is.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%% dtComment
6
7%msd01
8msrop(dtPassword,is,[AdtValue],Result):-  

9 msrVar(ptBoolean,TheResult),  

10 msrVar(ptInteger,MinLength),  

11(  

12(
13(
14  MinLength = [ptInteger,6],
15  msrNav([AdtValue],
16    [value,length,[],geq,[MinLength]],
17    [[ptBoolean,true]]))  

18)
19 -> TheResult = [ptBoolean,true]

```

#### C.41. FILE /SRC-GEN/PROLOG-REF-SPEC.../PRIMARYTYPESDATATYPES-DTPHONENUMBER-IS.PL29

```

20      ; TheResult = [ptBoolean,false]
21    )
22),
23 Result = TheResult
24.
25/*
26| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[[]]]],
27msrNav([X],[is[],[Result]]).
28X = [dtPassword,[],[[dtString,[[value,[ptString,'012345']]],[[]]]],
29Result = [ptBoolean,true] ?
30yes
31
32| ?- X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[[]]]],
33msrNav([X],[is[],[Result]]).
34X = [dtPassword,[],[[dtString,[[value,[ptString,'01234']]],[[]]]],
35Result = [ptBoolean,false] ?
36yes
37*/

```

Listing C.40: Prolog file PrimaryTypesDatatypes-dtPassword-is.pl.

#### C.41 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesData dtPhoneNumber-is.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%
5
6%% dtPhoneNumber
7
8% msd01
9msrop(dtPhoneNumber,is,[AdtValue],Result):-!
10msrVar(ptBoolean,TheResult),
11(
12 ( msrNav([AdtValue],
13   [value,length,[],gt,[[ptInteger,4]]],
14   [[ptBoolean,true]]),
15   msrNav([AdtValue],
16   [value,length,[],leq,[[ptInteger,30]]],
17   [[ptBoolean,true]])
18 )
19
20 -> TheResult = [ptBoolean,true]
21 ; TheResult = [ptBoolean,false]
22),
23 Result = TheResult
24.
25/*
26| ?- X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00']]],[[]]]],
27msrNav([X],[is[],[Result]]).
28
29X = [dtPhoneNumber,[],[[dtString,[[value,[ptString,'(+352) 46 66 44 60 00']]],[[]]]],
30
31Result = [ptBoolean,true] ?
32
33yes
34*/

```

Listing C.41: Prolog file PrimaryTypesDatatypes-dtPhoneNumber-is.pl.

#### C.42 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass etAlertStatus-is.pl

```

1%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */

```

```

3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% etAlertStatus
7
8% msd01
9msrop(etAlertStatus,is,[AdtValue],Result):-  

10msrVar(ptBoolean,TheResult),  

11(  

12  (  

13    member(AdtValue,[pending, valid, invalid])  

14  )  

15 -> TheResult = [ptBoolean,true]  

16 ; TheResult = [ptBoolean,false]  

17),  

18 Result = TheResult  

19.

```

Listing C.42: Prolog file PrimaryTypesDatatypes-etAlertStatus-is.pl.

#### C.43 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC etCrisisStatus-is.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% etCrisisStatus
7
8% msd01
9msrop(etCrisisStatus,is,[AdtValue],Result):-  

10msrVar(ptBoolean,TheResult),  

11(  

12  (  

13    member(AdtValue,[pending, handled, solved, closed])  

14  )  

15 -> TheResult = [ptBoolean,true]  

16 ; TheResult = [ptBoolean,false]  

17),  

18 Result = TheResult  

19.

```

Listing C.43: Prolog file PrimaryTypesDatatypes-etCrisisStatus-is.pl.

#### C.44 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesC etCrisisType-is.pl

```

1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% etCrisisType
7
8% msd01
9msrop(etCrisisType,is,[AdtValue],Result):-  

10msrVar(ptBoolean,TheResult),  

11(  

12  (  

13    member(AdtValue,[small, medium, huge])  

14  )  

15 -> TheResult = [ptBoolean,true]  

16 ; TheResult = [ptBoolean,false]  

17),  

18 Result = TheResult

```

#### C.45. FILE /SRC-GEN/PROLOG-REF-SPEC.../PRIMARYTYPESDATATYPES-ETHUMANKIND-IS.PL299

19.

Listing C.44: Prolog file PrimaryTypesDatatypes-etCrisisType-is.pl.

#### C.45 File ./src-gen/prolog-ref-spec/Operations/Concepts/PrimaryTypesClass etHumanKind-is.pl

```
1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5
6%% etHumanKind
7
8% msd01
9msrop(etHumanKind,is,[AdtValue],Result):-  
10msrVar(ptBoolean,TheResult),  
11(  
12 (
13     member(AdtValue,[witness,victim,anonymous])
14 )
15 -> TheResult = [ptBoolean,true]
16 ; TheResult = [ptBoolean,false]
17),
18 Result = TheResult
19.
```

Listing C.45: Prolog file PrimaryTypesDatatypes-etHumanKind-is.pl.

#### C.46 File ./src-gen/prolog-ref-spec/Operations/Concepts/SecondaryTypesDa dtSMS-is.pl

```
1%%%%%%%%%%%%%%%
2/* DISCONTIGUOUS PREDICATES */
3:- multifile msrop/4.
4%%%%%%%%%%%%%%%
5%% dtComment
6
7%msd01
8msrop(dtSMS,is,[AdtValue],Result):-  
9 msrVar(ptBoolean,TheResult),
10 msrVar(ptInteger,MaxLength),
11(
12 (
13 (
14     MaxLength = [ptInteger,160],
15     msrNav([AdtValue],
16         [value,length,[],leq,[MaxLength]],
17         [[ptBoolean,true]])
18 )
19 -> TheResult = [ptBoolean,true]
20 ; TheResult = [ptBoolean,false]
21 )
22),
23 Result = TheResult
24.
```

Listing C.46: Prolog file SecondaryTypesDatatypes-dtSMS-is.pl.



# Glossary

<b><i>abstract actor</i></b> an actor that is not . . . . .	22
<b><i>actor</i></b> An actor is a person, organization, or external system that plays a role in one or more interactions with the system . . . . .	18
<b><i>direct actor</i></b> an actor that interacts directly with the system. It thus belongs to the environment.	22
<b><i>indirect actor</i></b> an actor that interacts indirectly with the system through a direct actor. It thus belongs the domain but not to the environment. . . . .	22
<b><i>system operation</i></b> a functionality of the system that can be triggered by a message sent by an actor belonging to the environment. . . . .	18



# Bibliography

- [1] Guelfi, N.: Messir: A Scientific Method for the Software Engineer. to be published (2017)
- [2] Armour, F., Miller, G.: Advanced Use Case Modeling: Software Systems. Addison-Wesley (2001)
- [3] ISO/IEC: ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. (2011) ISO/IEC 13211-1.