ReGenESyS - Reborned GENeric and Expansible SYstem Simulator

2019.0507a

Generated by Doxygen 1.8.11

Contents

Chapter 1

GenESyS-Reborn

Generic and Expansible System Simulator

(Work in progress C++ port from the original in Pascal)

Developed by `rlcancian`

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

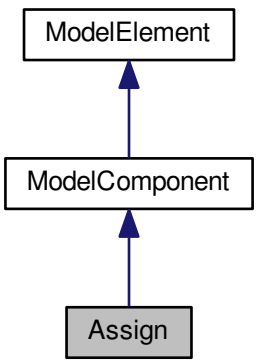
Chapter 5

Class Documentation

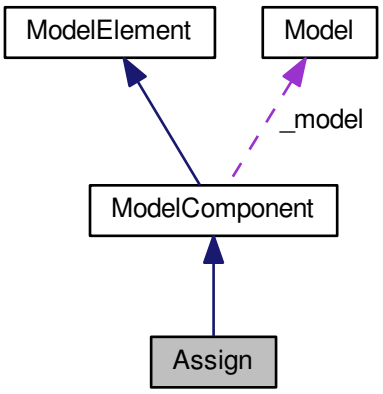
5.1 Assign Class Reference

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

Inheritance diagram for Assign:



Collaboration diagram for Assign:



Classes

- class Assignment

Public Types

- enum DestinationType : int { DestinationType::Attribute, DestinationType::Variable }

Public Member Functions

- Assign (Model *model)
- Assign (const Assign &orig)
- virtual ~Assign ()
- virtual std::string show ()
- List< Assignment * > * getAssignments () const

Protected Member Functions

- virtual void _execute (Entity *entity)
- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.1.1 Member Enumeration Documentation

5.1.1.1 enum Assign::DestinationType : int [strong]

Enumerator

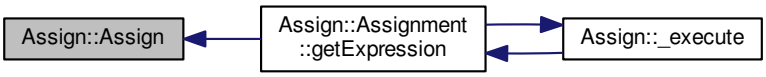
Attribute

Variable

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Assign::Assign (Model * *model*)

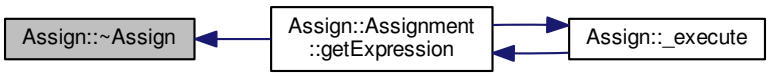
Here is the caller graph for this function:



5.1.2.2 Assign::Assign (const Assign & *orig*)

5.1.2.3 Assign::~Assign ( ) [virtual]

Here is the caller graph for this function:

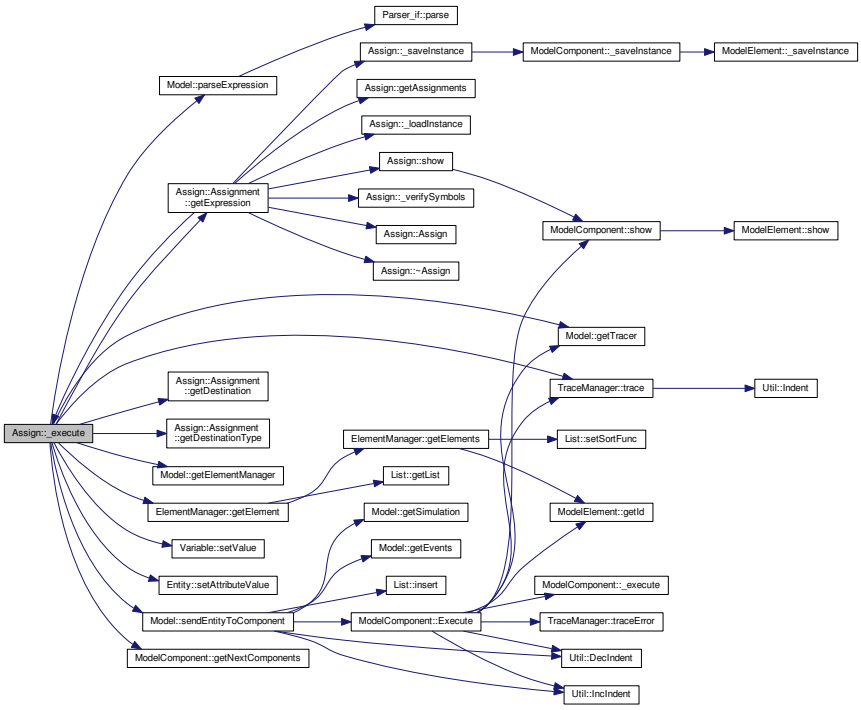


5.1.3 Member Function Documentation

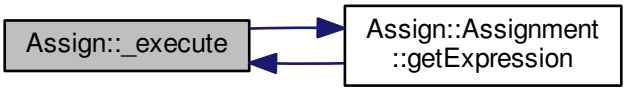
5.1.3.1 void Assign::_execute (Entity * *entity*) [protected], [virtual]

Implements ModelComponent.

Here is the call graph for this function:



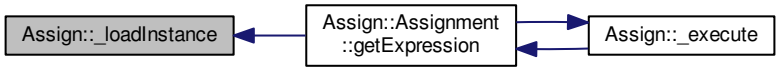
Here is the caller graph for this function:



5.1.3.2 void Assign::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

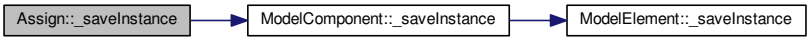
Here is the caller graph for this function:



5.1.3.3 std::list< std::string > * Assign::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:

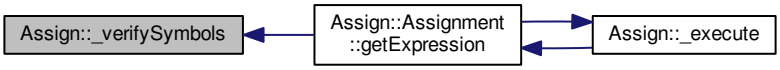


Here is the caller graph for this function:

5.1.3.4 bool Assign::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

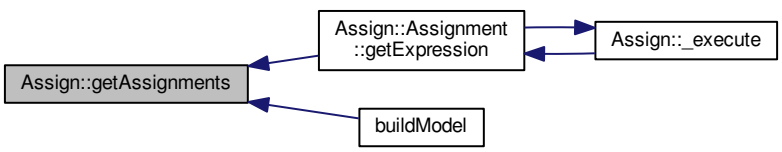
Implements ModelElement.

Here is the caller graph for this function:



5.1.3.5 List< Assign::Assignment * > * Assign::getAssignments ( ) const

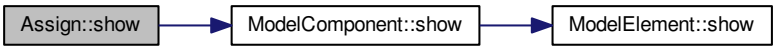
Here is the caller graph for this function:



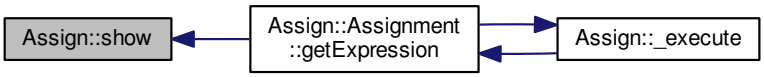
5.1.3.6 std::string Assign::show ( ) [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- Assign.h
- Assign.cpp

5.2 Assign::Assignment Class Reference

`#include <Assign.h>`

Public Member Functions

- Assignment (DestinationType destinationType, std::string destination, std::string expression)
- void setDestination (std::string _destination)
- std::string getDestination () const
- void setDestinationType (DestinationType _destinationType)
- DestinationType getDestinationType () const
- void setExpression (std::string _expression)
- std::string getExpression () const

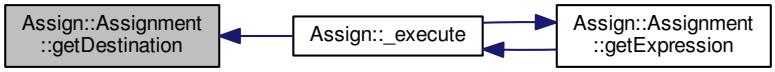
5.2.1 Constructor & Destructor Documentation

5.2.1.1 Assign::Assignment::Assignment (DestinationType *destinationType,* std::string *destination,* std::string *expression*) [inline]

5.2.2 Member Function Documentation

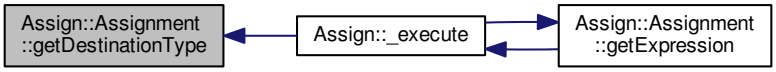
5.2.2.1 std::string Assign::Assignment::getDestination ( ) const [inline]

Here is the caller graph for this function:



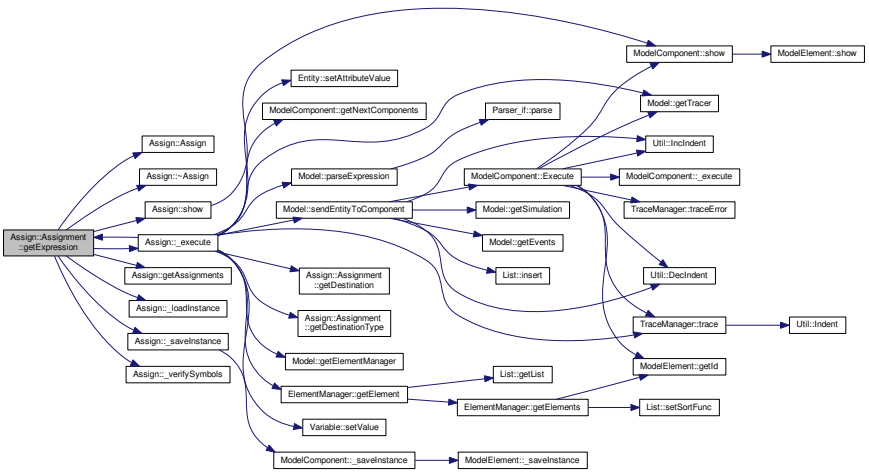
5.2.2.2 DestinationType Assign::Assignment::getDestinationType ( ) const [inline]

Here is the caller graph for this function:

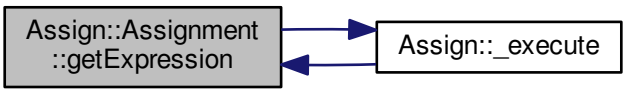


5.2.2.3 std::string Assign::Assignment::getExpression ( ) const [inline]

Here is the call graph for this function:



Here is the caller graph for this function:



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

5.2.2.5 void Assign::Assignment::setDestinationType (DestinationType *_destinationType*) [inline]

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

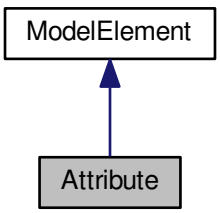
The documentation for this class was generated from the following file:

- Assign.h

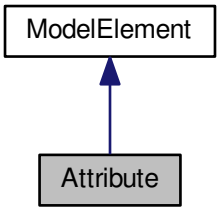
5.3 Attribute Class Reference

`#include <Attribute.h>`

Inheritance diagram for Attribute:



Collaboration diagram for Attribute:



Public Member Functions

- Attribute ()
- Attribute (std::string name)
- Attribute (const Attribute &orig)
- virtual ~Attribute ()
- virtual std::string show ()

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.3.1 Constructor & Destructor Documentation

5.3.1.1 Attribute::Attribute ( )

5.3.1.2 Attribute::Attribute (std::string *name*)

5.3.1.3 Attribute::Attribute (const Attribute & *orig*)

5.3.1.4 Attribute::~Attribute ( ) [virtual]

5.3.2 Member Function Documentation

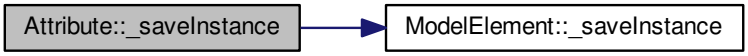
5.3.2.1 void Attribute::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.3.2.2 std::list< std::string > * Attribute::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



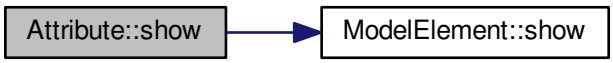
5.3.2.3 bool Attribute::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

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

Reimplemented from ModelElement.

Here is the call graph for this function:



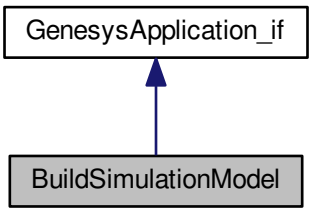
The documentation for this class was generated from the following files:

- Attribute.h
- Attribute.cpp

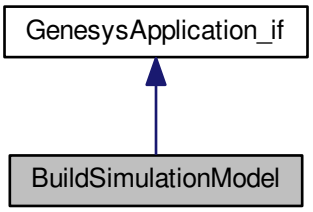
5.4 BuildSimulationModel Class Reference

`#include <BuildSimulationModel.h>`

Inheritance diagram for BuildSimulationModel:



Collaboration diagram for BuildSimulationModel:



Public Member Functions

- BuildSimulationModel ()
- int main (int argc, char **argv)

5.4.1 Constructor & Destructor Documentation

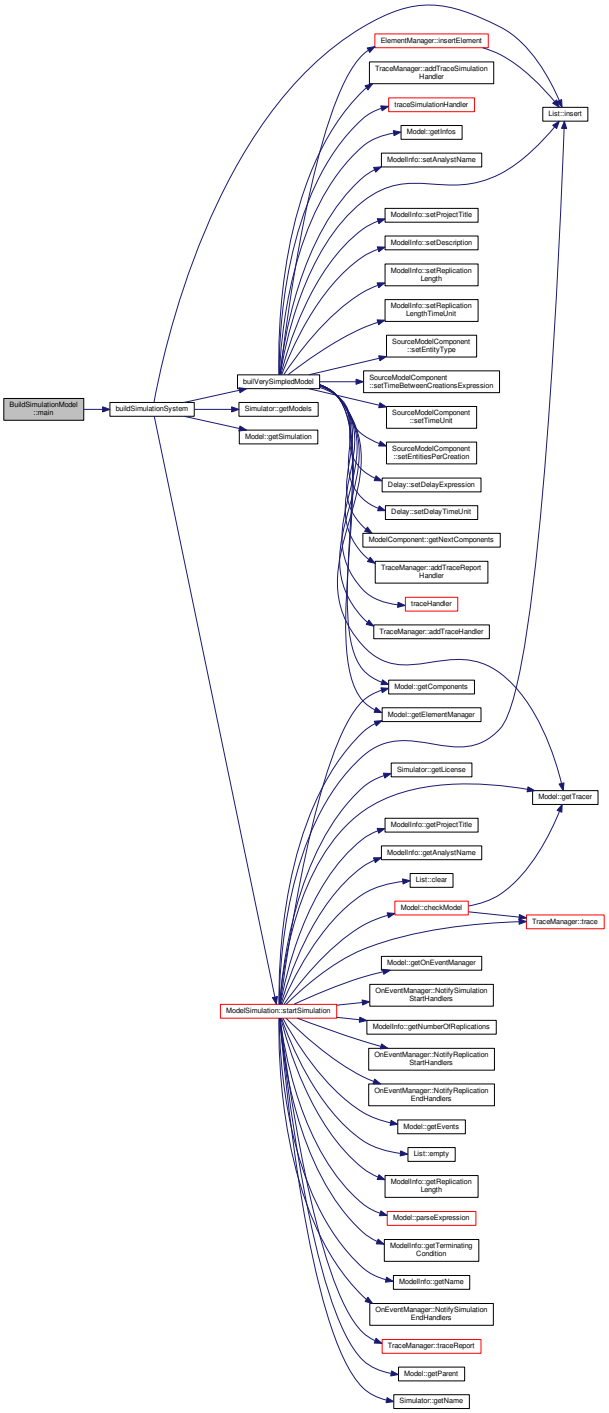
5.4.1.1 BuildSimulationModel::BuildSimulationModel ( )

5.4.2 Member Function Documentation

5.4.2.1 int BuildSimulationModel::main (int *argc,* char ** *argv*) [virtual]

Implements GenesysApplication_if.

Here is the call graph for this function:



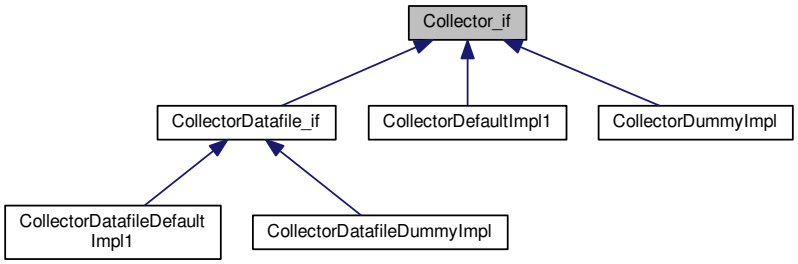
The documentation for this class was generated from the following files:

- BuildSimulationModel.h
- BuildSimulationModel.cpp

5.5 Collector_if Class Reference

`#include <Collector_if.h>`

Inheritance diagram for Collector_if:



Public Member Functions

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

5.5.1 Detailed Description

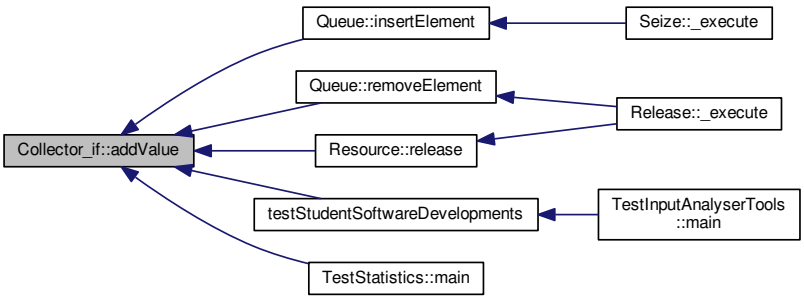
Interface for collecting values of a single stochastic variable. Values collected can be used as base for statistical analysis.

5.5.2 Member Function Documentation

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

Implemented in CollectorDatafileDefaultImpl1, CollectorDummyImpl, CollectorDatafileDummyImpl, and CollectorDefaultImpl1.

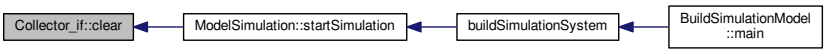
Here is the caller graph for this function:



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

Implemented in CollectorDatafileDefaultImpl1, CollectorDummyImpl, CollectorDatafileDummyImpl, and CollectorDefaultImpl1.

Here is the caller graph for this function:



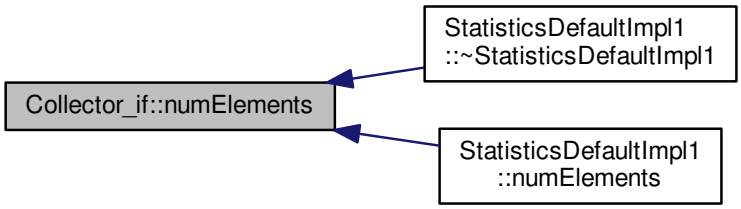
5.5.2.3 virtual double Collector_if::getLastValue ( ) [pure virtual]

Implemented in CollectorDatafileDefaultImpl1, CollectorDummyImpl, CollectorDatafileDummyImpl, and CollectorDefaultImpl1.

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

Implemented in CollectorDatafileDefaultImpl1, CollectorDummyImpl, CollectorDatafileDummyImpl, and CollectorDefaultImpl1.

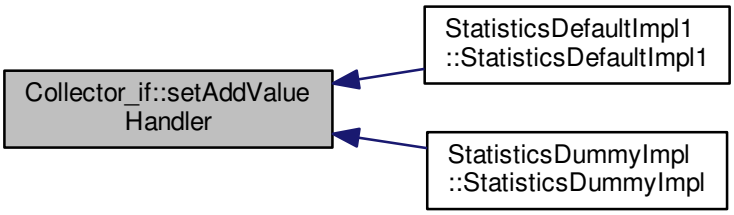
Here is the caller graph for this function:



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

Implemented in CollectorDatafileDefaultImpl1, CollectorDatafileDummyImpl, CollectorDummyImpl, and CollectorDefaultImpl1.

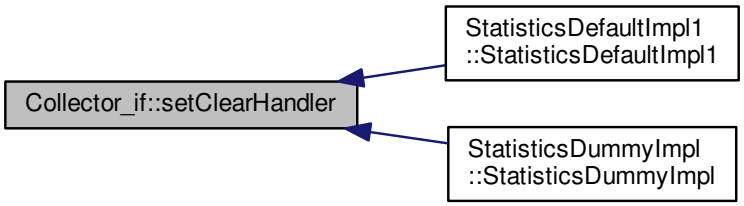
Here is the caller graph for this function:



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

Implemented in CollectorDatafileDefaultImpl1, CollectorDatafileDummyImpl, CollectorDummyImpl, and Collector↩ DefaultImpl1.

Here is the caller graph for this function:



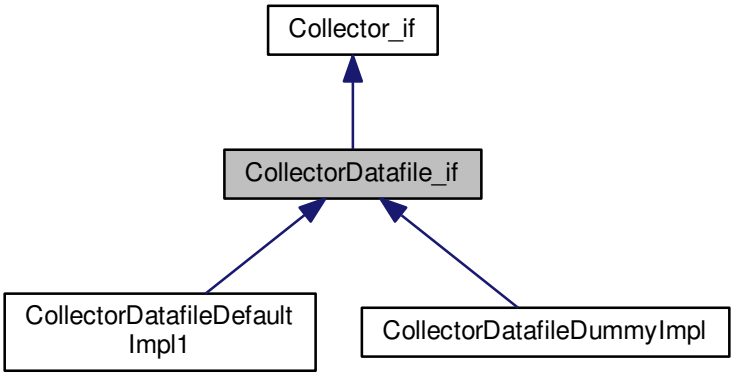
The documentation for this class was generated from the following file:

- Collector_if.h

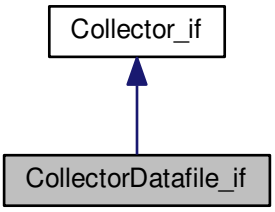
5.6 CollectorDatafile_if Class Reference

`#include <CollectorDatafile_if.h>`

Inheritance diagram for CollectorDatafile_if:



Collaboration diagram for CollectorDatafile_if:



Public Member Functions

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

5.6.1 Detailed Description

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

5.6.2 Member Function Documentation

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

Get the next value in the file and advances the pointer

Implemented in CollectorDatafileDefaultImpl1, and CollectorDatafileDummyImpl.

Here is the caller graph for this function:

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

Set the pointer to the first value in the file

Implemented in CollectorDatafileDefaultImpl1, and CollectorDatafileDummyImpl.

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDatafileDummyImpl.

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

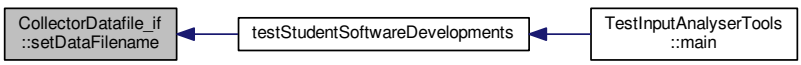
Get a value from a specific position

Implemented in CollectorDatafileDefaultImpl1, and CollectorDatafileDummyImpl.

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

Implemented in CollectorDatafileDefaultImpl1, and CollectorDatafileDummyImpl.

Here is the caller graph for this function:



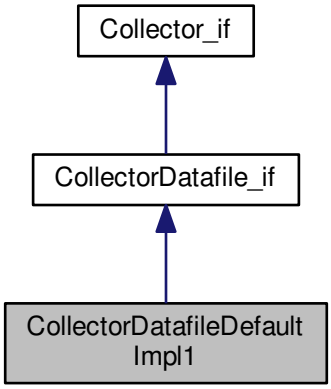
The documentation for this class was generated from the following file:

- CollectorDatafile_if.h

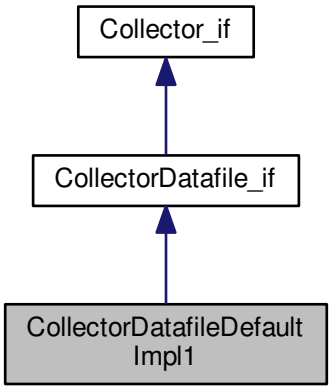
5.7 CollectorDatafileDefaultImpl1 Class Reference

`#include <CollectorDatafileDefaultImpl1.h>`

Inheritance diagram for CollectorDatafileDefaultImpl1:



Collaboration diagram for CollectorDatafileDefaultImpl1:



Public Member Functions

- CollectorDatafileDefaultImpl1 ()
- CollectorDatafileDefaultImpl1 (const CollectorDatafileDefaultImpl1 &orig)
- virtual ~CollectorDatafileDefaultImpl1 ()
- void clear ()

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

5.7.1 Constructor & Destructor Documentation

5.7.1.1 CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 ( )

5.7.1.2 CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 (const CollectorDatafileDefaultImpl1 & *orig*)

5.7.1.3 CollectorDatafileDefaultImpl1::~CollectorDatafileDefaultImpl1 ( ) [virtual]

5.7.2 Member Function Documentation

5.7.2.1 void CollectorDatafileDefaultImpl1::addValue (double *value*) [virtual]

Implements Collector_if.

5.7.2.2 void CollectorDatafileDefaultImpl1::clear ( ) [virtual]

Implements Collector_if.

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

Get the next value in the file and advances the pointer

Implements CollectorDatafile_if.

5.7.2.4 double CollectorDatafileDefaultImpl1::getLastValue ( ) [virtual]

Implements Collector_if.

5.7.2.5 double CollectorDatafileDefaultImpl1::getNextValue ( ) [virtual]

Set the pointer to the first value in the file

Implements CollectorDatafile_if.

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

Implements CollectorDatafile_if.

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

Implements Collector_if.

5.7.2.8 void CollectorDatafileDefaultImpl1::seekFirstValue ( ) [virtual]

Get a value from a specific position

Implements CollectorDatafile_if.

5.7.2.9 void CollectorDatafileDefaultImpl1::setAddValueHandler (CollectorAddValueHandler *addValueHandler*) [virtual]

Implements Collector_if.

5.7.2.10 void CollectorDatafileDefaultImpl1::setClearHandler (CollectorClearHandler *clearHandler*) [virtual]

Implements Collector_if.

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

Implements CollectorDatafile_if.

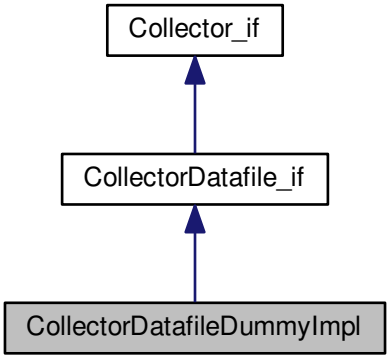
The documentation for this class was generated from the following files:

- CollectorDatafileDefaultImpl1.h
- CollectorDatafileDefaultImpl1.cpp

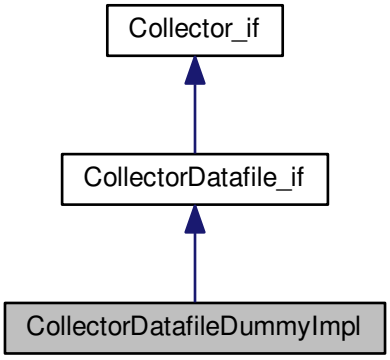
5.8 CollectorDatafileDummyImpl Class Reference

`#include <CollectorDatafileDummyImpl.h>`

Inheritance diagram for CollectorDatafileDummyImpl:



Collaboration diagram for CollectorDatafileDummyImpl:



Public Member Functions

- CollectorDatafileDummyImpl ()
- CollectorDatafileDummyImpl (const CollectorDatafileDummyImpl &orig)
- ~CollectorDatafileDummyImpl ()
- void clear ()
- void addValue (double value)

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

5.8.1 Constructor & Destructor Documentation

5.8.1.1 CollectorDatafileDummyImpl::CollectorDatafileDummyImpl ( )

5.8.1.2 CollectorDatafileDummyImpl::CollectorDatafileDummyImpl (const CollectorDatafileDummyImpl & *orig*)

5.8.1.3 CollectorDatafileDummyImpl::~CollectorDatafileDummyImpl ( )

5.8.2 Member Function Documentation

5.8.2.1 void CollectorDatafileDummyImpl::addValue (double *value*) [virtual]

Implements Collector_if.

5.8.2.2 void CollectorDatafileDummyImpl::clear ( ) [virtual]

Implements Collector_if.

5.8.2.3 std::string CollectorDatafileDummyImpl::getDataFilename ( ) [virtual]

Get the next value in the file and advances the pointer

Implements CollectorDatafile_if.

5.8.2.4 double CollectorDatafileDummyImpl::getLastValue ( ) [virtual]

Implements Collector_if.

5.8.2.5 double CollectorDatafileDummyImpl::getNextValue ( ) [virtual]

Set the pointer to the first value in the file

Implements CollectorDatafile_if.

5.8.2.6 double CollectorDatafileDummyImpl::getValue (unsigned int *num*) [virtual]

Implements CollectorDatafile_if.

5.8.2.7 unsigned long CollectorDatafileDummyImpl::numElements ( ) [virtual]

Implements Collector_if.

5.8.2.8 void CollectorDatafileDummyImpl::seekFirstValue ( ) [virtual]

Get a value from a specific position

Implements CollectorDatafile_if.

5.8.2.9 void CollectorDatafileDummyImpl::setAddValueHandler (CollectorAddValueHandler *addValueHandler*) [virtual]

Implements Collector_if.

5.8.2.10 void CollectorDatafileDummyImpl::setClearHandler (CollectorClearHandler *clearHandler*) [virtual]

Implements Collector_if.

5.8.2.11 void CollectorDatafileDummyImpl::setDataFilename (std::string *filename*) [virtual]

Implements CollectorDatafile_if.

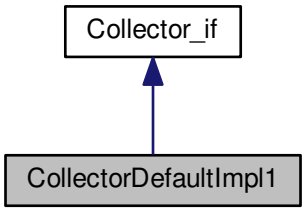
The documentation for this class was generated from the following files:

- CollectorDatafileDummyImpl.h
- CollectorDatafileDummyImpl.cpp

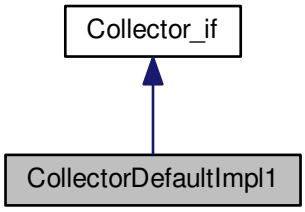
5.9 CollectorDefaultImpl1 Class Reference

`#include <CollectorDefaultImpl1.h>`

Inheritance diagram for CollectorDefaultImpl1:



Collaboration diagram for CollectorDefaultImpl1:



Public Member Functions

- CollectorDefaultImpl1 ()
- CollectorDefaultImpl1 (const CollectorDefaultImpl1 &orig)
- virtual ~CollectorDefaultImpl1 ()
- void clear ()
- void addValue (double value)
- double getLastValue ()
- unsigned long numElements ()
- void setAddValueHandler (CollectorAddValueHandler addValueHandler)
- void setClearHandler (CollectorClearHandler clearHandler)

5.9.1 Constructor & Destructor Documentation

5.9.1.1 CollectorDefaultImpl1::CollectorDefaultImpl1 ( )

5.9.1.2 CollectorDefaultImpl1::CollectorDefaultImpl1 (const CollectorDefaultImpl1 & *orig*)

5.9.1.3 CollectorDefaultImpl1::~CollectorDefaultImpl1 ( ) [virtual]

5.9.2 Member Function Documentation

5.9.2.1 void CollectorDefaultImpl1::addValue (double *value*) [virtual]

Implements Collector_if.

5.9.2.2 void CollectorDefaultImpl1::clear ( ) [virtual]

Implements Collector_if.

5.9.2.3 double CollectorDefaultImpl1::getLastValue ( ) [virtual]

Implements Collector_if.

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

Implements Collector_if.

5.9.2.5 void CollectorDefaultImpl1::setAddValueHandler (CollectorAddValueHandler *addValueHandler*) [virtual]

Implements Collector_if.

5.9.2.6 void CollectorDefaultImpl1::setClearHandler (CollectorClearHandler *clearHandler*) [virtual]

Implements Collector_if.

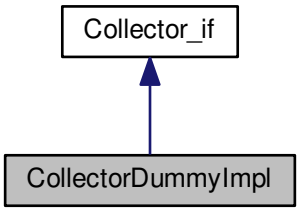
The documentation for this class was generated from the following files:

- CollectorDefaultImpl1.h
- CollectorDefaultImpl1.cpp

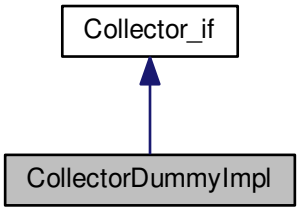
5.10 CollectorDummyImpl Class Reference

`#include <CollectorDummyImpl.h>`

Inheritance diagram for CollectorDummyImpl:



Collaboration diagram for CollectorDummyImpl:



Public Member Functions

- CollectorDummyImpl ()
- CollectorDummyImpl (const CollectorDummyImpl &orig)
- ~CollectorDummyImpl ()
- void clear ()
- void addValue (double value)
- double getLastValue ()
- unsigned long numElements ()
- void setAddValueHandler (CollectorAddValueHandler addValueHandler)
- void setClearHandler (CollectorClearHandler clearHandler)

5.10.1 Constructor & Destructor Documentation

5.10.1.1 CollectorDummyImpl::CollectorDummyImpl ( )

5.10.1.2 CollectorDummyImpl::CollectorDummyImpl (const CollectorDummyImpl & *orig*)

5.10.1.3 CollectorDummyImpl::~CollectorDummyImpl ( )

5.10.2 Member Function Documentation

5.10.2.1 void CollectorDummyImpl::addValue (double *value*) [virtual]

Implements Collector_if.

5.10.2.2 void CollectorDummyImpl::clear ( ) [virtual]

Implements Collector_if.

5.10.2.3 double CollectorDummyImpl::getLastValue ( ) [virtual]

Implements Collector_if.

5.10.2.4 unsigned long CollectorDummyImpl::numElements ( ) [virtual]

Implements Collector_if.

5.10.2.5 void CollectorDummyImpl::setAddValueHandler (CollectorAddValueHandler *addValueHandler*) [virtual]

Implements Collector_if.

5.10.2.6 void CollectorDummyImpl::setClearHandler (CollectorClearHandler *clearHandler*) [virtual]

Implements Collector_if.

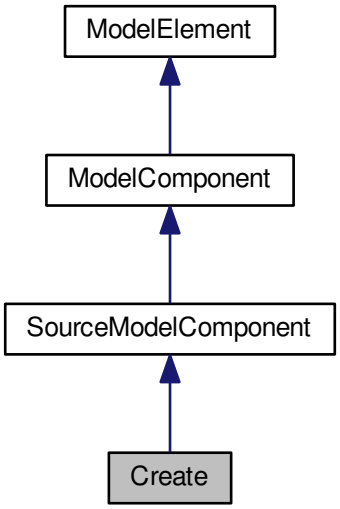
The documentation for this class was generated from the following files:

- CollectorDummyImpl.h
- CollectorDummyImpl.cpp

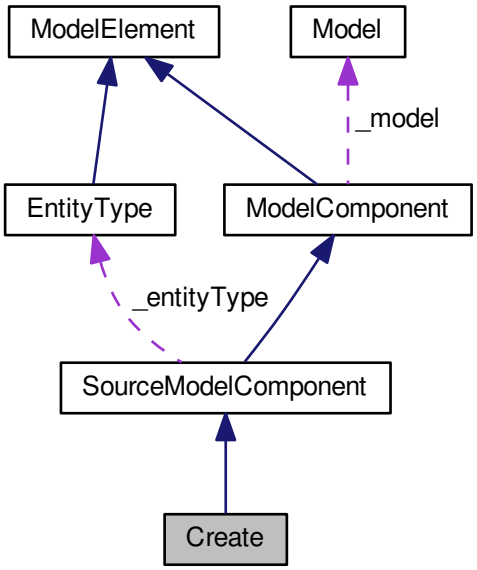
5.11 Create Class Reference

`#include <Create.h>`

Inheritance diagram for Create:



Collaboration diagram for Create:



Public Member Functions

- Create (Model ∗model)
- Create (const Create &orig)
- virtual ∼Create ()
- virtual std::string show ()

Protected Member Functions

- virtual void _execute (Entity ∗entity)
- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > ∗ _saveInstance ()
- virtual bool _verifySymbols (std::string ∗errorMessage)

Additional Inherited Members

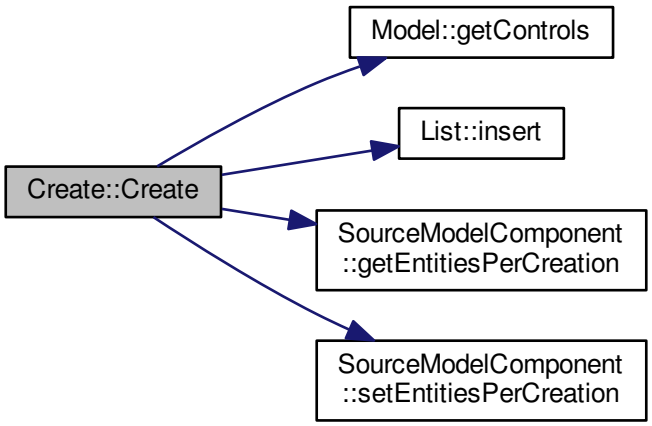
5.11.1 Detailed Description

Create is the most basic component to include the first entities into the model, and therefore is a source component (derived from SourceModelComponent)

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Create::Create (Model ∗ *model*)

Here is the call graph for this function:



5.11.2.2 Create::Create (const Create & *orig*)

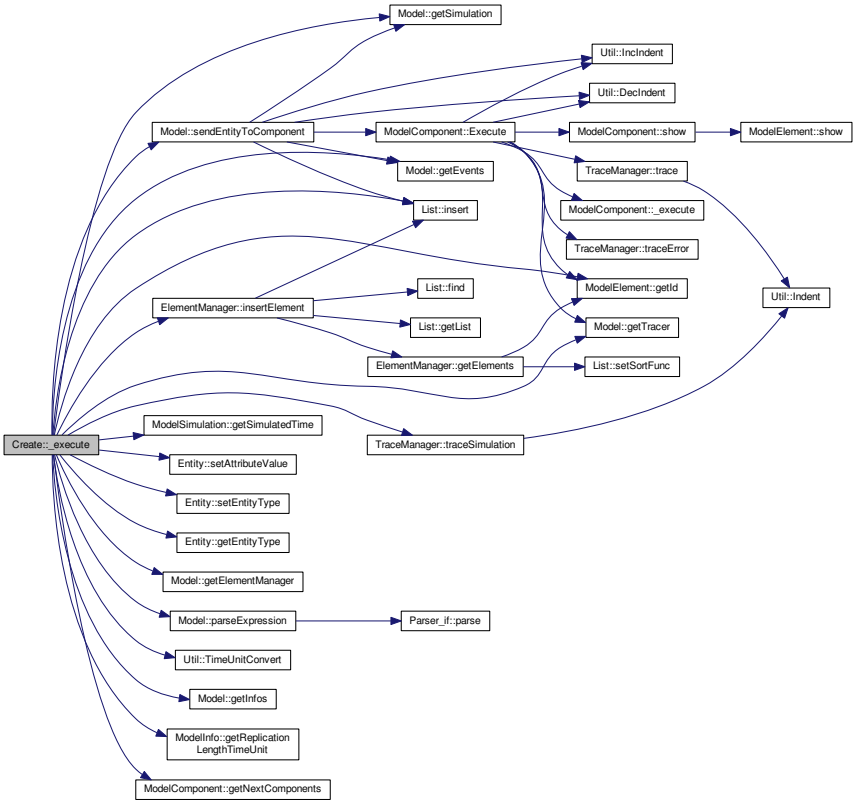
5.11.2.3 Create::~Create ( ) [virtual]

5.11.3 Member Function Documentation

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

Implements ModelComponent.

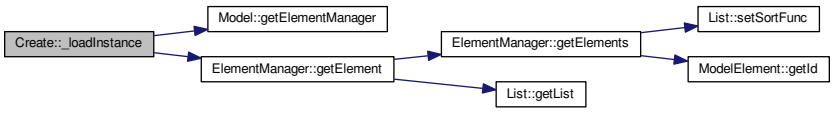
Here is the call graph for this function:



5.11.3.2 void Create::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

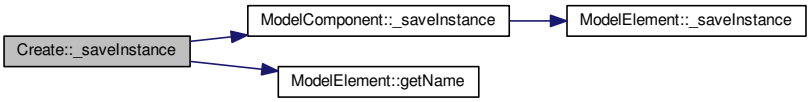
Here is the call graph for this function:



5.11.3.3 std::list< std::string > * Create::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelComponent.

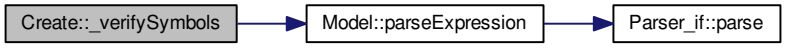
Here is the call graph for this function:



5.11.3.4 bool Create::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

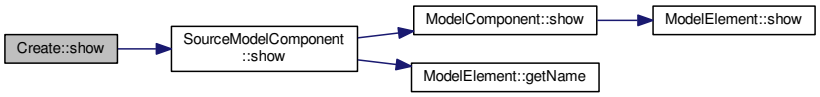
Here is the call graph for this function:



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

Reimplemented from SourceModelComponent.

Here is the call graph for this function:



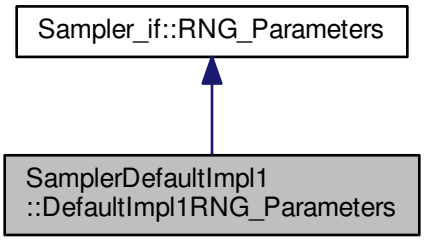
The documentation for this class was generated from the following files:

- Create.h
- Create.cpp

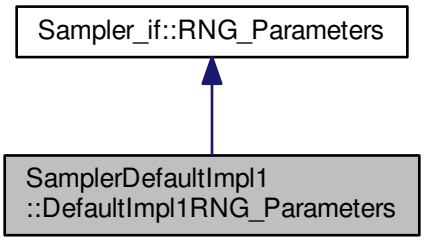
5.12 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Class Reference

`#include <SamplerDefaultImpl1.h>`

Inheritance diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:



Collaboration diagram for SamplerDefaultImpl1::DefaultImpl1RNG_Parameters:



Public Attributes

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

5.12.1 Member Data Documentation

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

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

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

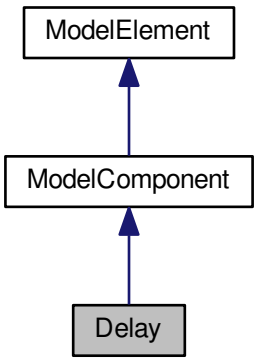
The documentation for this class was generated from the following file:

- SamplerDefaultImpl1.h

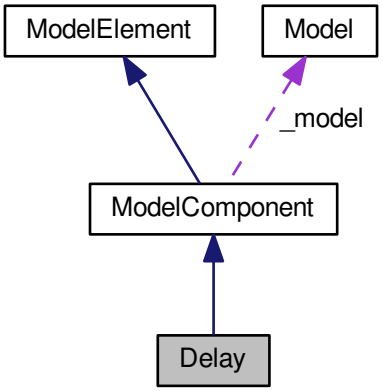
5.13 Delay Class Reference

`#include <Delay.h>`

Inheritance diagram for Delay:



Collaboration diagram for Delay:



Public Member Functions

- Delay (Model *model)
- Delay (const Delay &orig)
- virtual ~Delay ()
- void setDelayExpression (std::string _delayExpression)
- std::string getDelayExpression () const
- void setDelayTimeUnit (Util::TimeUnit _delayTimeUnit)
- Util::TimeUnit getDelayTimeUnit () const
- virtual std::string show ()

Protected Member Functions

- virtual void _execute (Entity *entity)
- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.13.1 Constructor & Destructor Documentation

5.13.1.1 Delay::Delay (Model * *model*)

5.13.1.2 Delay::Delay (const Delay & *orig*)

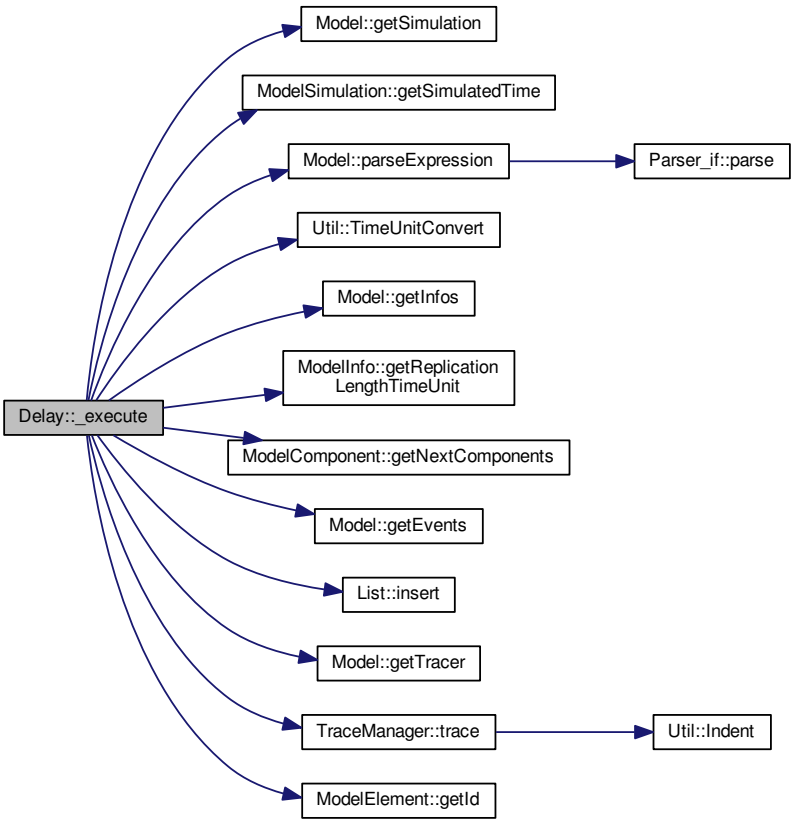
5.13.1.3 Delay::~Delay ( ) [virtual]

5.13.2 Member Function Documentation

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

Implements ModelComponent.

Here is the call graph for this function:



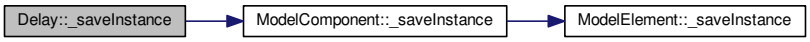
5.13.2.2 void Delay::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.13.2.3 std::list< std::string > * Delay::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelComponent.

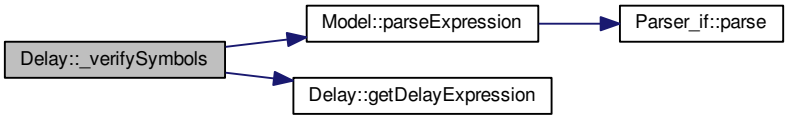
Here is the call graph for this function:



5.13.2.4 bool Delay::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

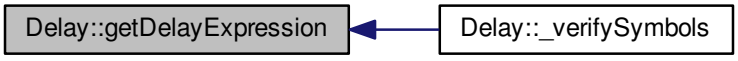
Implements ModelElement.

Here is the call graph for this function:



5.13.2.5 std::string Delay::getDelayExpression ( ) const

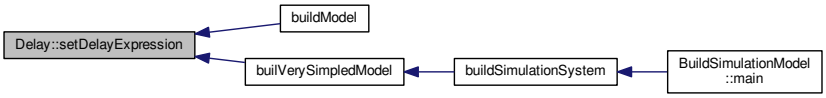
Here is the caller graph for this function:



5.13.2.6 Util::TimeUnit Delay::getDelayTimeUnit ( ) const

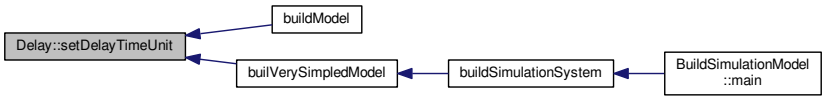
5.13.2.7 void Delay::setDelayExpression (std::string *_delayExpression*)

Here is the caller graph for this function:



5.13.2.8 void Delay::setDelayTimeUnit (Util::TimeUnit *_delayTimeUnit*)

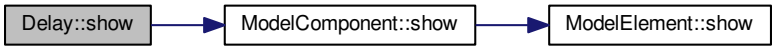
Here is the caller graph for this function:



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

Reimplemented from ModelComponent.

Here is the call graph for this function:



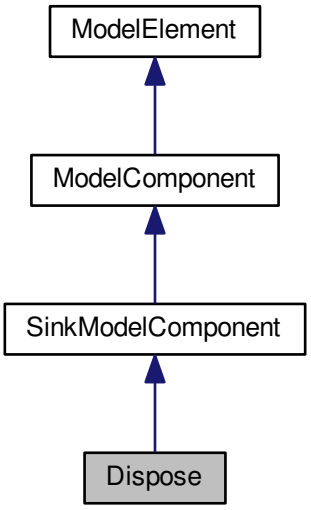
The documentation for this class was generated from the following files:

- Delay.h
- Delay.cpp

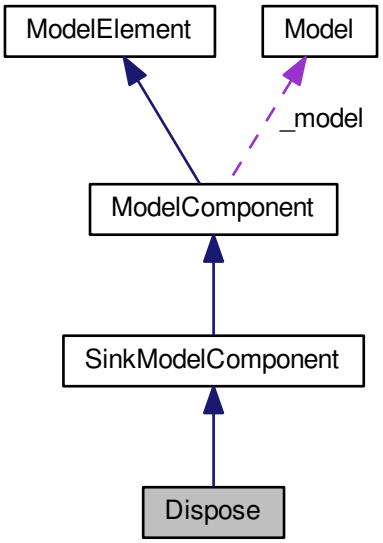
5.14 Dispose Class Reference

`#include <Dispose.h>`

Inheritance diagram for Dispose:



Collaboration diagram for Dispose:



Public Member Functions

- Dispose (Model *model)
- Dispose (const Dispose &orig)
- virtual ~Dispose ()
- virtual std::string show ()
- void setCollectStatistics (bool _collectStatistics)
- bool isCollectStatistics () const

Protected Member Functions

- virtual void _execute (Entity *entity)
- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.14.1 Constructor & Destructor Documentation

5.14.1.1 Dispose::Dispose (Model * *model*)

5.14.1.2 Dispose::Dispose (const Dispose & *orig*)

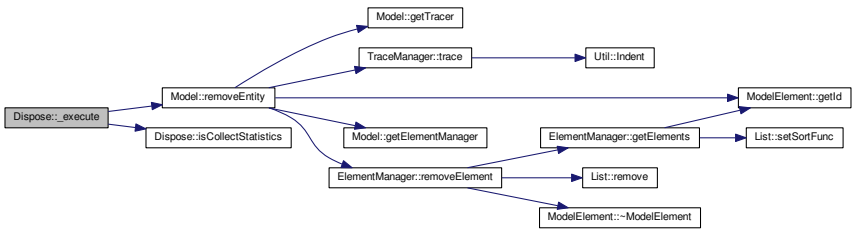
5.14.1.3 Dispose::~Dispose ( ) [virtual]

5.14.2 Member Function Documentation

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

Implements ModelComponent.

Here is the call graph for this function:



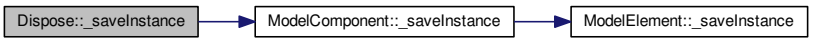
5.14.2.2 void Dispose::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.14.2.3 std::list< std::string > * Dispose::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:

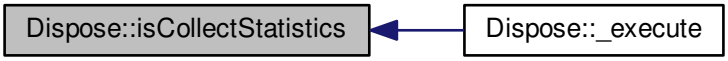


5.14.2.4 bool Dispose::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.14.2.5 bool Dispose::isCollectStatistics ( ) const

Here is the caller graph for this function:

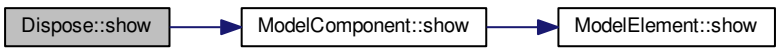


5.14.2.6 void Dispose::setCollectStatistics (bool *_collectStatistics*)

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

Reimplemented from ModelComponent.

Here is the call graph for this function:



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

- Dispose.h
- Dispose.cpp

5.15 ElementManager Class Reference

`#include <ElementManager.h>`

Public Member Functions

- ElementManager (Model *model)
- ElementManager (const ElementManager &orig)
- virtual ~ElementManager ()
- bool insertElement (std::string infraTypename, ModelElement *infra)
- void removeElement (std::string infraTypename, ModelElement *infra)
- List< ModelElement * > * getElements (std::string infraTypename) const
- ModelElement * getElement (std::string infraTypename, Util::identitifcation id)
- ModelElement * getElement (std::string infraTypename, std::string name)
- std::list< std::string > * getElementTypenames () const
- void show ()

5.15.1 Detailed Description

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

5.15.2 Constructor & Destructor Documentation

5.15.2.1 ElementManager::ElementManager (Model * *model*)

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

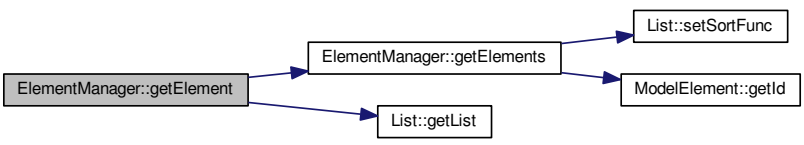
5.15.2.2 ElementManager::ElementManager (const ElementManager & *orig*)

5.15.2.3 ElementManager::~ElementManager ( ) [virtual]

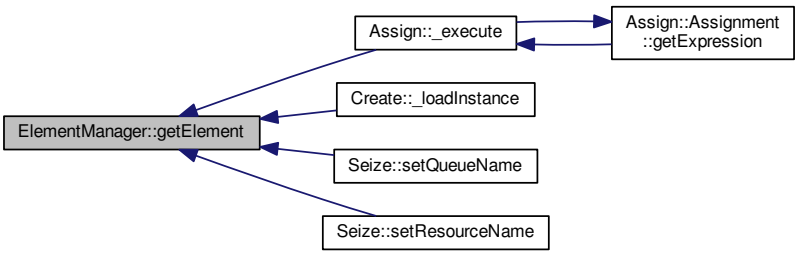
5.15.3 Member Function Documentation

5.15.3.1 ModelElement * ElementManager::getElement (std::string *infraTypename*, Util::identitifcation *id*)

Here is the call graph for this function:

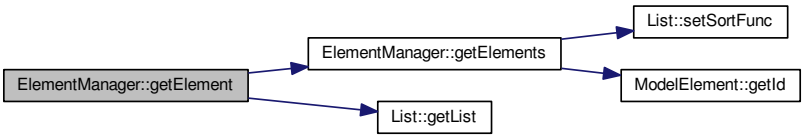


Here is the caller graph for this function:



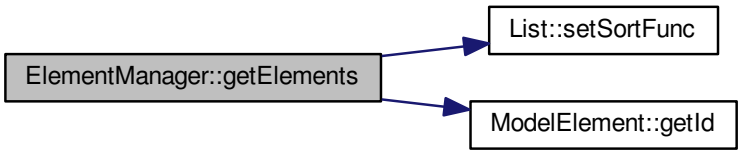
5.15.3.2 ModelElement * ElementManager::getElement (std::string *infraTypename*, std::string *name*)

Here is the call graph for this function:

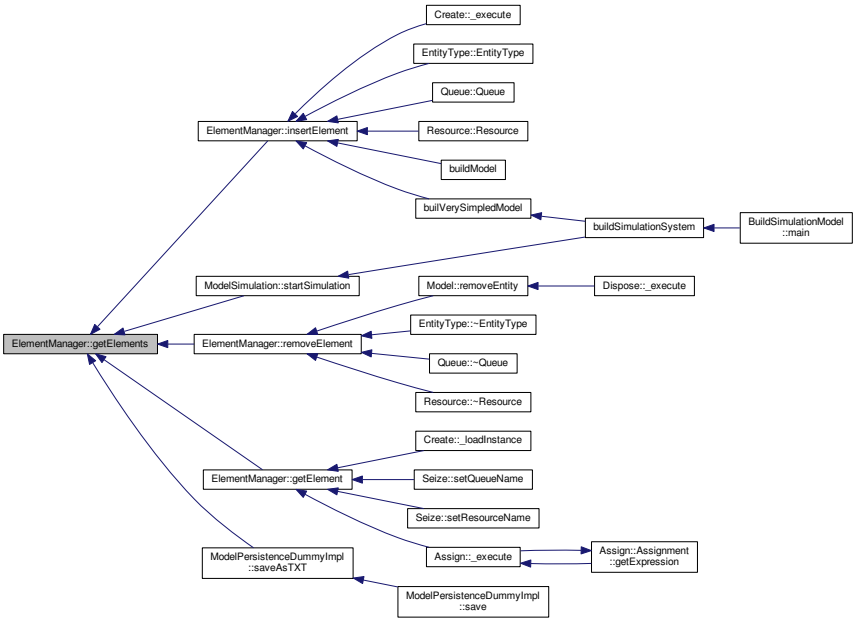


5.15.3.3 List< ModelElement * > * ElementManager::getElements (std::string *infraTypename*) const

Here is the call graph for this function:

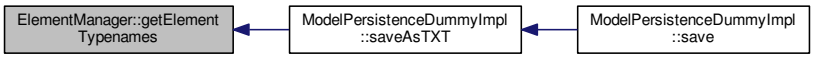


Here is the caller graph for this function:



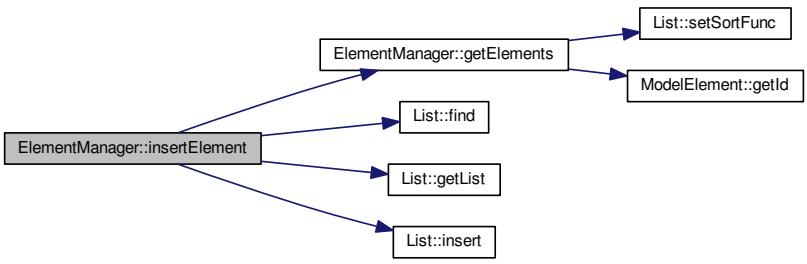
5.15.3.4 std::list< std::string > * ElementManager::getElementTypenames ( ) const

Here is the caller graph for this function:

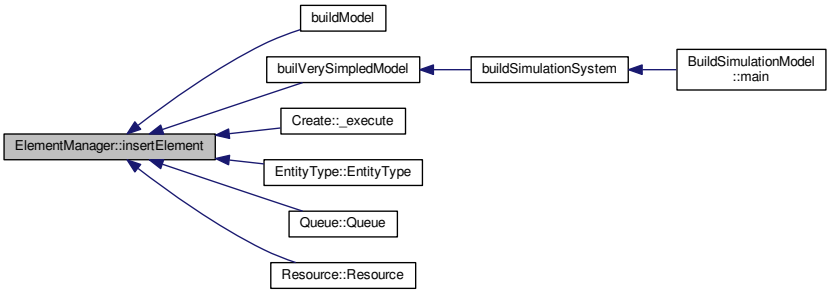


5.15.3.5 bool ElementManager::insertElement (std::string *infraTypename,* ModelElement * *infra*)

Here is the call graph for this function:

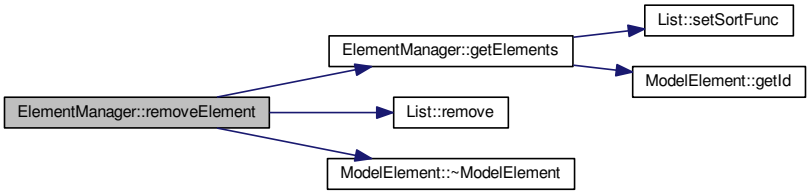


Here is the caller graph for this function:

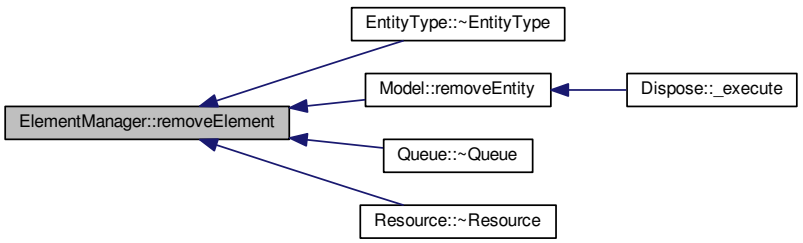


5.15.3.6 void ElementManager::removeElement (std::string *infraTypename,* ModelElement * *infra*)

Here is the call graph for this function:

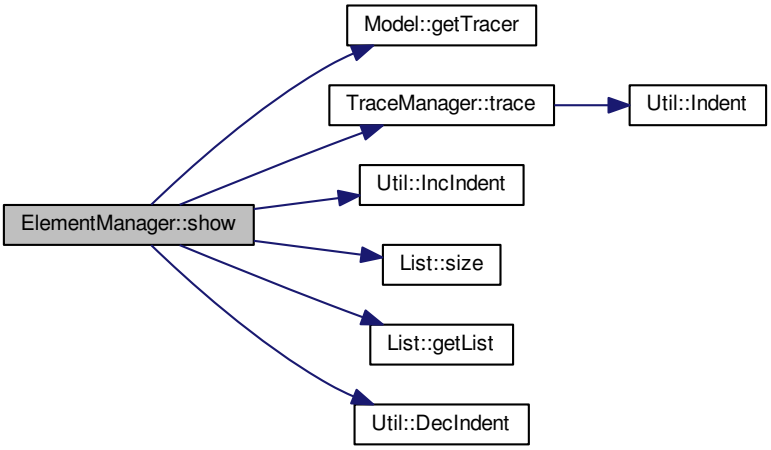


Here is the caller graph for this function:



5.15.3.7 void ElementManager::show ( )

Here is the call graph for this function:



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

- ElementManager.h
- ElementManager.cpp

5.16 ElementManager_if Class Reference

`#include <ElementManager_if.h>`

Public Member Functions

- ElementManager_if ()
- ElementManager_if (const ElementManager_if &orig)
- virtual ~ElementManager_if ()

5.16.1 Constructor & Destructor Documentation

5.16.1.1 ElementManager_if::ElementManager_if ( )

5.16.1.2 ElementManager_if::ElementManager_if (const ElementManager_if & *orig*)

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

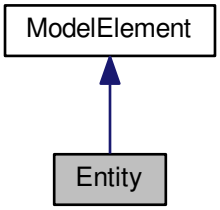
The documentation for this class was generated from the following file:

- ElementManager_if.h

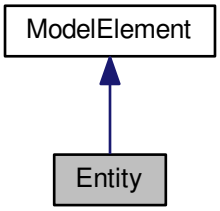
5.17 Entity Class Reference

`#include <Entity.h>`

Inheritance diagram for Entity:



Collaboration diagram for Entity:



Public Member Functions

- Entity ()
- Entity (const Entity &orig)
- virtual ~Entity ()
- virtual std::string show ()
- void setEntityType (EntityType *entityType)
- EntityType * getEntityType () const
- double getAttributeValue (std::string attributeName)
- void setAttributeValue (std::string attributeName, double value)

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.17.1 Constructor & Destructor Documentation

5.17.1.1 Entity::Entity ( )

5.17.1.2 Entity::Entity (const Entity & *orig*)

5.17.1.3 Entity::~Entity ( ) [virtual]

5.17.2 Member Function Documentation

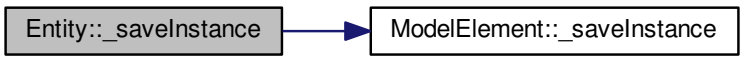
5.17.2.1 void Entity::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.17.2.2 std::list< std::string > * Entity::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



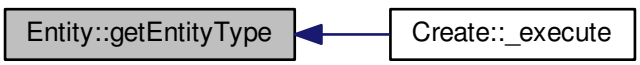
5.17.2.3 bool Entity::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.17.2.4 double Entity::getAttributeValue (std::string *attributeName*)

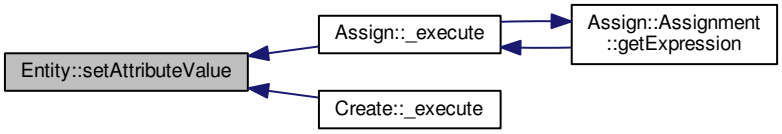
5.17.2.5 EntityType * Entity::getEntityType ( ) const

Here is the caller graph for this function:



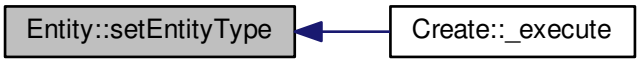
5.17.2.6 void Entity::setAttributeValue (std::string *attributeName,* double *value*)

Here is the caller graph for this function:



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

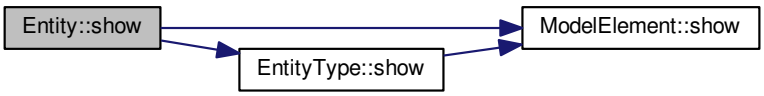
Here is the caller graph for this function:



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

Reimplemented from ModelElement.

Here is the call graph for this function:



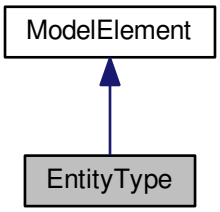
The documentation for this class was generated from the following files:

- Entity.h
- Entity.cpp

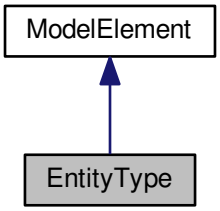
5.18 EntityType Class Reference

`#include <EntityType.h>`

Inheritance diagram for EntityType:



Collaboration diagram for EntityType:



Public Member Functions

- EntityType (ElementManager *elemManager)
- EntityType (ElementManager *elemManager, std::string name)
- EntityType (ElementManager *elemManager, std::string name, std::string initialPicture, double initialWaitingCost, double initialVACost, double initialNVACost, double initialOtherCost)
- EntityType (const EntityType &orig)
- virtual ~EntityType ()
- virtual std::string show ()
- void setInitialWaitingCost (double _initialWaitingCost)
- double getInitialWaitingCost () const
- void setInitialOtherCost (double _initialOtherCost)
- double getInitialOtherCost () const
- void setInitialNVACost (double _initialNVACost)
- double getInitialNVACost () const
- void setInitialVACost (double _initialVACost)

- double getInitialVACost () const
- void setInitialPicture (std::string _initialPicture)
- std::string getInitialPicture () const
- StatisticsCollector * getCstatTimeInSystem () const
- StatisticsCollector * getCstatNVATime () const
- StatisticsCollector * getCstatVATime () const
- StatisticsCollector * getCstatOtherTime () const
- StatisticsCollector * getCstatTransferTime () const
- StatisticsCollector * getCstatWaitingTime () const

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.18.1 Constructor & Destructor Documentation

5.18.1.1 EntityType::EntityType (ElementManager * *elemManager*)

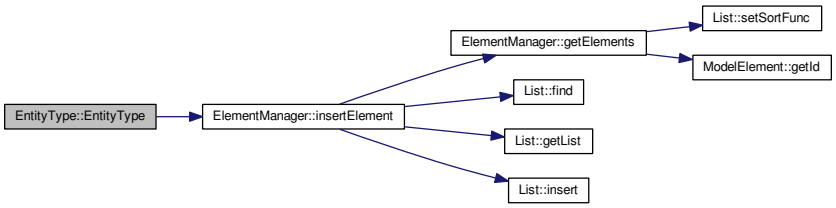
Here is the caller graph for this function:

5.18.1.2 EntityType::EntityType (ElementManager * *elemManager,* std::string *name*)

Here is the call graph for this function:

5.18.1.3 EntityType::EntityType (ElementManager * *elemManager,* std::string *name,* std::string *initialPicture,* double *initialWaitingCost,* double *initialVACost,* double *initialNVACost,* double *initialOtherCost*)

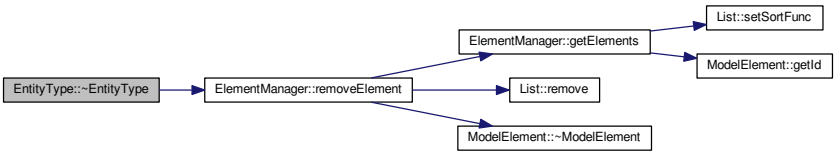
Here is the call graph for this function:



5.18.1.4 EntityType::EntityType (const EntityType & *orig*)

5.18.1.5 EntityType::~EntityType ( ) [virtual]

Here is the call graph for this function:



5.18.2 Member Function Documentation

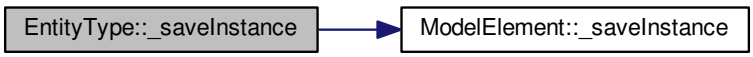
5.18.2.1 void EntityType::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.18.2.2 std::list< std::string > * EntityType::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



5.18.2.3 bool EntityType::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.18.2.4 StatisticsCollector * EntityType::getCstatNVATime ( ) const

5.18.2.5 StatisticsCollector * EntityType::getCstatOtherTime ( ) const

5.18.2.6 StatisticsCollector * EntityType::getCstatTimeInSystem ( ) const

5.18.2.7 StatisticsCollector * EntityType::getCstatTransferTime ( ) const

5.18.2.8 StatisticsCollector * EntityType::getCstatVATime ( ) const

5.18.2.9 StatisticsCollector * EntityType::getCstatWaitingTime ( ) const

5.18.2.10 double EntityType::getInitialNVACost ( ) const

5.18.2.11 double EntityType::getInitialOtherCost ( ) const

5.18.2.12 std::string EntityType::getInitialPicture ( ) const

5.18.2.13 double EntityType::getInitialVACost ( ) const

5.18.2.14 double EntityType::getInitialWaitingCost ( ) const

5.18.2.15 void EntityType::setInitialNVACost (double *_initialNVACost*)

5.18.2.16 void EntityType::setInitialOtherCost (double *_initialOtherCost*)

5.18.2.17 void EntityType::setInitialPicture (std::string *_initialPicture*)

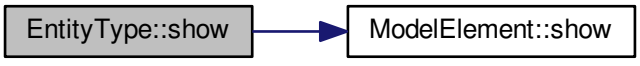
5.18.2.18 void EntityType::setInitialVACost (double *_initialVACost*)

5.18.2.19 void EntityType::setInitialWaitingCost (double *_initialWaitingCost*)

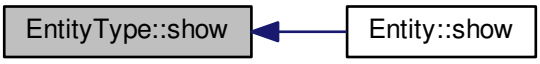
5.18.2.20 std::string EntityType::show ( ) [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- EntityType.h
- EntityType.cpp

5.19 Event Class Reference

`#include <Event.h>`

Public Member Functions

- Event (double time, Entity ∗entity, ModelComponent ∗component)
- Event (const Event &orig)
- virtual ∼Event ()
- double getTime () const
- ModelComponent ∗ getComponent () const
- Entity ∗ getEntity () const
- std::string show ()

5.19.1 Constructor & Destructor Documentation

5.19.1.1 Event::Event (double *time,* Entity ∗ *entity,* ModelComponent ∗ *component*)

5.19.1.2 Event::Event (const Event & *orig*)

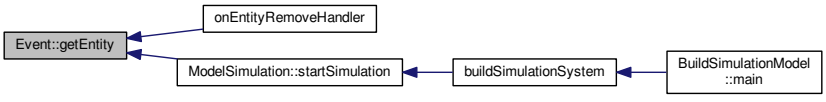
5.19.1.3 Event::∼Event ( ) [virtual]

5.19.2 Member Function Documentation

5.19.2.1 ModelComponent ∗ Event::getComponent ( ) const

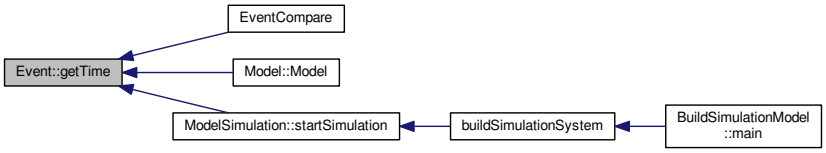
5.19.2.2 Entity ∗ Event::getEntity ( ) const

Here is the caller graph for this function:



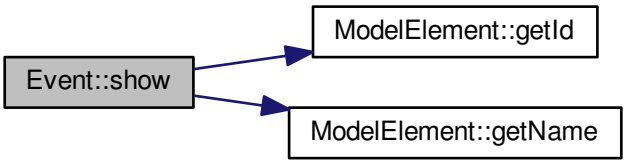
5.19.2.3 double Event::getTime ( ) const

Here is the caller graph for this function:

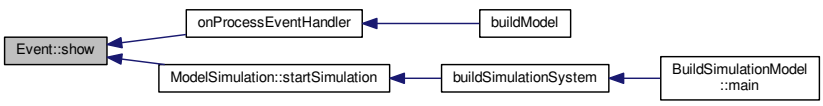


5.19.2.4 std::string Event::show ( )

Here is the call graph for this function:



Here is the caller graph for this function:



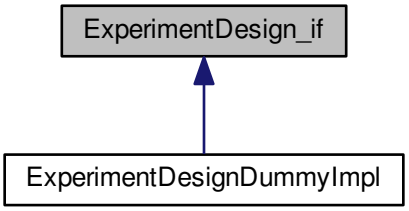
The documentation for this class was generated from the following files:

- Event.h
- Event.cpp

5.20 ExperimentDesign_if Class Reference

`#include <ExperimentDesign_if.h>`

Inheritance diagram for ExperimentDesign_if:



Public Member Functions

- virtual ProcessAnalyser_if * getProcessAnalyser () const =0
- virtual bool generate2krScenarioExperiments ()=0
- virtual bool calculateContributionAndCoefficients ()=0
- virtual std::list< FactorOrInteractionContribution * > * getContributions () const =0

5.20.1 Detailed Description

It designs a set of experiments (SimulationScenario) where que level of factors (SimulationControl) are set automatically to create a 2^k.r experiment design, and where the contributions of the factors and their interactions (just a set of SimulationControl) can be obtained.

5.20.2 Member Function Documentation

5.20.2.1 virtual bool ExperimentDesign_if::calculateContributionAndCoefficients ( ) [pure virtual]

Implemented in ExperimentDesignDummyImpl.

5.20.2.2 virtual bool ExperimentDesign_if::generate2krScenarioExperiments ( ) [pure virtual]

Implemented in ExperimentDesignDummyImpl.

5.20.2.3 virtual std::list<FactorOrInteractionContribution*>* ExperimentDesign_if::getContributions ( ) const [pure virtual]

Implemented in ExperimentDesignDummyImpl.

5.20.2.4 virtual ProcessAnalyser_if* ExperimentDesign_if::getProcessAnalyser ( ) const [pure virtual]

Implemented in ExperimentDesignDummyImpl.

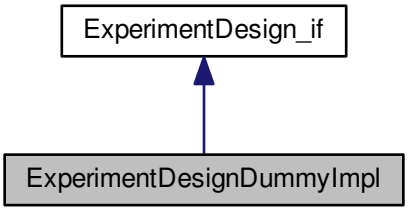
The documentation for this class was generated from the following file:

- ExperimentDesign_if.h

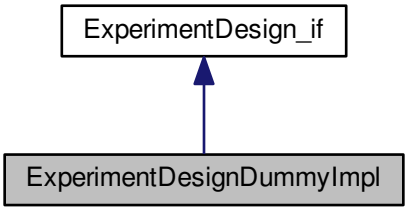
5.21 ExperimentDesignDummyImpl Class Reference

`#include <ExperimentDesignDummyImpl.h>`

Inheritance diagram for ExperimentDesignDummyImpl:



Collaboration diagram for ExperimentDesignDummyImpl:



Public Member Functions

- ExperimentDesignDummyImpl ()
- ExperimentDesignDummyImpl (const ExperimentDesignDummyImpl &orig)
- virtual ~ExperimentDesignDummyImpl ()
- ProcessAnalyser_if * getProcessAnalyser () const
- bool generate2krScenarioExperiments ()
- bool calculateContributionAndCoefficients ()
- std::list< FactorOrInteractionContribution * > * getContributions () const

5.21.1 Constructor & Destructor Documentation

5.21.1.1 ExperimentDesignDummyImpl::ExperimentDesignDummyImpl ( )

5.21.1.2 ExperimentDesignDummyImpl::ExperimentDesignDummyImpl (const ExperimentDesignDummyImpl & *orig*)

5.21.1.3 ExperimentDesignDummyImpl::∼ExperimentDesignDummyImpl ( ) [virtual]

5.21.2 Member Function Documentation

5.21.2.1 bool ExperimentDesignDummyImpl::calculateContributionAndCoefficients ( ) [virtual]

Implements ExperimentDesign_if.

5.21.2.2 bool ExperimentDesignDummyImpl::generate2krScenarioExperiments ( ) [virtual]

Implements ExperimentDesign_if.

5.21.2.3 std::list< FactorOrInteractionContribution * > * ExperimentDesignDummyImpl::getContributions ( ) const [virtual]

Implements ExperimentDesign_if.

5.21.2.4 ProcessAnalyser_if * ExperimentDesignDummyImpl::getProcessAnalyser ( ) const [virtual]

Implements ExperimentDesign_if.

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

- ExperimentDesignDummyImpl.h
- ExperimentDesignDummyImpl.cpp

5.22 FactorOrInteractionContribution Class Reference

`#include <FactorOrInteractionContribution.h>`

Public Member Functions

- FactorOrInteractionContribution (double contribution, double modelCoefficient, std::list< SimulationControl * > *controls)
- FactorOrInteractionContribution (const FactorOrInteractionContribution &orig)
- ∼FactorOrInteractionContribution ()
- double getModelCoefficient () const
- std::list< SimulationControl * > * getControls () const
- double getContribution () const

5.22.1 Detailed Description

This simple class corresponds to a factor when it refers to just one SimulationControl, or to the interaction between two or more factors when it refers to more SimulationControl. It also encapsulates the contribution of the factor or interaction and its coefficient in the full model that estimates one specific SimulationResponse.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 FactorOrInteractionContribution::FactorOrInteractionContribution (double *contribution,* double *modelCoefficient,* std::list< SimulationControl * > * *controls*)

5.22.2.2 FactorOrInteractionContribution::FactorOrInteractionContribution (const FactorOrInteractionContribution & *orig*)

5.22.2.3 FactorOrInteractionContribution::~FactorOrInteractionContribution ( )

5.22.3 Member Function Documentation

5.22.3.1 double FactorOrInteractionContribution::getContribution ( ) const

5.22.3.2 std::list< SimulationControl * > * FactorOrInteractionContribution::getControls ( ) const

5.22.3.3 double FactorOrInteractionContribution::getModelCoefficient ( ) const

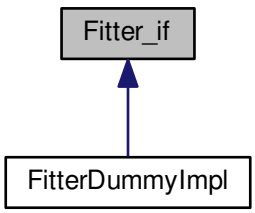
The documentation for this class was generated from the following files:

- FactorOrInteractionContribution.h
- FactorOrInteractionContribution.cpp

5.23 Fitter_if Class Reference

```
#include <Fitter_if.h>
```

Inheritance diagram for Fitter_if:



Public Member Functions

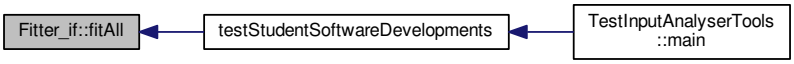
- virtual bool isNormalDistributed (double confidencelevel)=0
- virtual void fitUniform (double ∗sqrerror, double ∗min, double ∗max)=0
- virtual void fitTriangular (double ∗sqrerror, double ∗min, double ∗mo, double ∗max)=0
- virtual void fitNormal (double ∗sqrerror, double ∗avg, double ∗stddev)=0
- virtual void fitExpo (double ∗sqrerror, double ∗avg1)=0
- virtual void fitErlang (double ∗sqrerror, double ∗avg, double ∗m)=0
- virtual void fitBeta (double ∗sqrerror, double ∗alpha, double ∗beta, double ∗infLimit, double ∗supLimit)=0
- virtual void fitWeibull (double ∗sqrerror, double ∗alpha, double ∗scale)=0
- virtual void fitAll (double ∗sqrerror, std::string ∗name)=0
- virtual void setDataFilename (std::string dataFilename)=0
- virtual std::string getDataFilename ()=0

5.23.1 Member Function Documentation

5.23.1.1 virtual void Fitter_if::fitAll (double ∗ *sqrerror,* std::string ∗ *name*) [pure virtual]

Implemented in FitterDummyImpl.

Here is the caller graph for this function:



5.23.1.2 virtual void Fitter_if::fitBeta (double ∗ *sqrerror,* double ∗ *alpha,* double ∗ *beta,* double ∗ *infLimit,* double ∗ *supLimit*) [pure virtual]

Implemented in FitterDummyImpl.

5.23.1.3 virtual void Fitter_if::fitErlang (double ∗ *sqrerror,* double ∗ *avg,* double ∗ *m*) [pure virtual]

Implemented in FitterDummyImpl.

5.23.1.4 virtual void Fitter_if::fitExpo (double ∗ *sqrerror,* double ∗ *avg1*) [pure virtual]

Implemented in FitterDummyImpl.

5.23.1.5 virtual void Fitter_if::fitNormal (double * *sqrerror,* double * *avg,* double * *stddev*) [pure virtual]

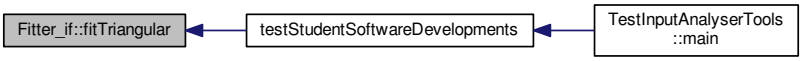
Implemented in FitterDummyImpl.

Here is the caller graph for this function:

5.23.1.6 virtual void Fitter_if::fitTriangular (double * *sqrerror,* double * *min,* double * *mo,* double * *max*) [pure virtual]

Implemented in FitterDummyImpl.

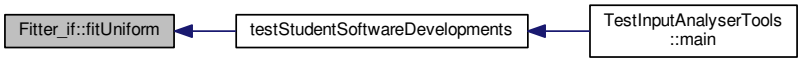
Here is the caller graph for this function:



5.23.1.7 virtual void Fitter_if::fitUniform (double * *sqrerror,* double * *min,* double * *max*) [pure virtual]

Implemented in FitterDummyImpl.

Here is the caller graph for this function:



5.23.1.8 virtual void Fitter_if::fitWeibull (double * *sqrerror,* double * *alpha,* double * *scale*) [pure virtual]

Implemented in FitterDummyImpl.

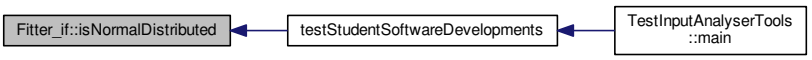
5.23.1.9 virtual std::string Fitter_if::getDataFilename ( ) [pure virtual]

Implemented in FitterDummyImpl.

5.23.1.10 virtual bool Fitter_if::isNormalDistributed (double *confidencelevel*) [pure virtual]

Implemented in FitterDummyImpl.

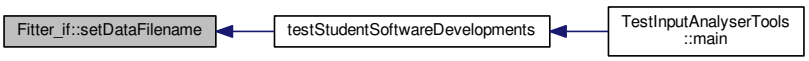
Here is the caller graph for this function:



5.23.1.11 virtual void Fitter_if::setDataFilename (std::string *dataFilename*) [pure virtual]

Implemented in FitterDummyImpl.

Here is the caller graph for this function:



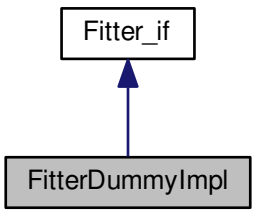
The documentation for this class was generated from the following file:

- Fitter_if.h

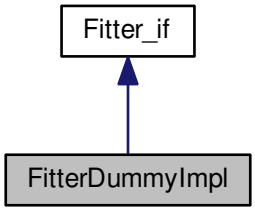
5.24 FitterDummyImpl Class Reference

`#include <FitterDummyImpl.h>`

Inheritance diagram for FitterDummyImpl:



Collaboration diagram for FitterDummyImpl:



Public Member Functions

- FitterDummyImpl ()
- FitterDummyImpl (const FitterDummyImpl &orig)
- ~FitterDummyImpl ()
- bool isNormalDistributed (double confidencelevel)
- void fitUniform (double *sqrerror, double *min, double *max)
- void fitTriangular (double *sqrerror, double *min, double *mo, double *max)
- void fitNormal (double *sqrerror, double *avg, double *stddev)
- void fitExpo (double *sqrerror, double *avg1)
- void fitErlang (double *sqrerror, double *avg, double *m)
- void fitBeta (double *sqrerror, double *alpha, double *beta, double *infLimit, double *supLimit)
- void fitWeibull (double *sqrerror, double *alpha, double *scale)
- void fitAll (double *sqrerror, std::string *name)
- void setDataFilename (std::string dataFilename)
- std::string getDataFilename ()

5.24.1 Constructor & Destructor Documentation

5.24.1.1 FitterDummyImpl::FitterDummyImpl ( )

5.24.1.2 FitterDummyImpl::FitterDummyImpl (const FitterDummyImpl & *orig*)

5.24.1.3 FitterDummyImpl::~FitterDummyImpl ( )

5.24.2 Member Function Documentation

5.24.2.1 void FitterDummyImpl::fitAll (double * *sqrerror,* std::string * *name*) [virtual]

Implements Fitter_if.

5.24.2.2 void FitterDummyImpl::fitBeta (double * *sqrerror,* double * *alpha,* double * *beta,* double * *infLimit,* double * *supLimit*) [virtual]

Implements Fitter_if.

5.24.2.3 void FitterDummyImpl::fitErlang (double * *sqrerror,* double * *avg,* double * *m*) [virtual]

Implements Fitter_if.

5.24.2.4 void FitterDummyImpl::fitExpo (double * *sqrerror,* double * *avg1*) [virtual]

Implements Fitter_if.

5.24.2.5 void FitterDummyImpl::fitNormal (double * *sqrerror,* double * *avg,* double * *stddev*) [virtual]

Implements Fitter_if.

5.24.2.6 void FitterDummyImpl::fitTriangular (double * *sqrerror,* double * *min,* double * *mo,* double * *max*) [virtual]

Implements Fitter_if.

5.24.2.7 void FitterDummyImpl::fitUniform (double * *sqrerror,* double * *min,* double * *max*) [virtual]

Implements Fitter_if.

5.24.2.8 void FitterDummyImpl::fitWeibull (double * *sqrerror,* double * *alpha,* double * *scale*) [virtual]

Implements Fitter_if.

5.24.2.9 std::string FitterDummyImpl::getDataFilename ( ) [virtual]

Implements Fitter_if.

5.24.2.10 bool FitterDummyImpl::isNormalDistributed (double *confidencelevel*) [virtual]

Implements Fitter_if.

5.24.2.11 void FitterDummyImpl::setDataFilename (std::string *dataFilename*) [virtual]

Implements Fitter_if.

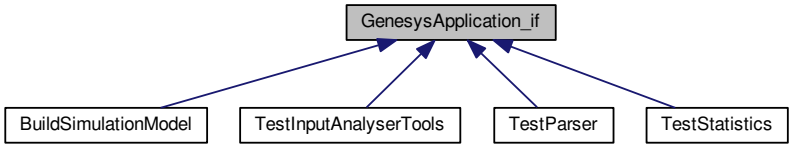
The documentation for this class was generated from the following files:

- FitterDummyImpl.h
- FitterDummyImpl.cpp

5.25 GenesysApplication_if Class Reference

`#include <GenesysApplication_if.h>`

Inheritance diagram for GenesysApplication_if:



Public Member Functions

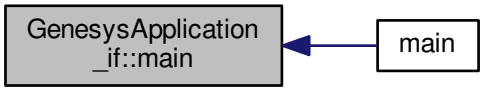
- virtual int main (int argc, char **argv)=0

5.25.1 Member Function Documentation

5.25.1.1 virtual int GenesysApplication_if::main (int *argc,* char ** *argv*) [pure virtual]

Implemented in TestInputAnalyserTools, TestParser, BuildSimulationModel, and TestStatistics.

Here is the caller graph for this function:



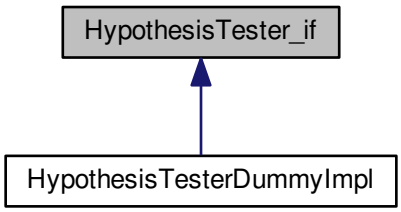
The documentation for this class was generated from the following file:

- GenesysApplication_if.h

5.26 HypothesisTester_if Class Reference

`#include <HypothesisTester_if.h>`

Inheritance diagram for HypothesisTester_if:



Public Types

- enum H1Comparition { DIFFERENT = 1, LESS_THAN = 2, GREATER_THAN = 3 }

Public Member Functions

- virtual double testAverage (double confidencelevel, double avg, H1Comparition comp)=0
- virtual double testProportion (double confidencelevel, double prop, H1Comparition comp)=0
- virtual double testVariance (double confidencelevel, double var, H1Comparition comp)=0
- virtual double testAverage (double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)=0
- virtual double testProportion (double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)=0
- virtual double testVariance (double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)=0
- virtual void setDataFilename (std::string dataFilename)=0
- virtual std::string getDataFilename ()=0

5.26.1 Detailed Description

Interface for parametric hypothesis tests based on a datafile.

5.26.2 Member Enumeration Documentation

5.26.2.1 enum HypothesisTester_if::H1Comparition

Enumerator

DIFFERENT

LESS_THAN

GREATER_THAN

5.26.3 Member Function Documentation

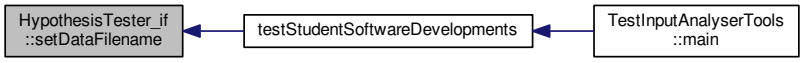
5.26.3.1 virtual std::string HypothesisTester_if::getDataFilename ( ) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

5.26.3.2 virtual void HypothesisTester_if::setDataFilename (std::string *dataFilename*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

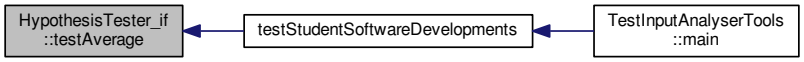
Here is the caller graph for this function:



5.26.3.3 virtual double HypothesisTester_if::testAverage (double *confidencelevel,* double *avg,* H1Comparition *comp*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

Here is the caller graph for this function:



5.26.3.4 virtual double HypothesisTester_if::testAverage (double *confidencelevel,* std::string *secondPopulationDataFilename,* H1Comparition *comp*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

5.26.3.5 virtual double HypothesisTester_if::testProportion (double *confidencelevel,* double *prop,* H1Comparition *comp*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

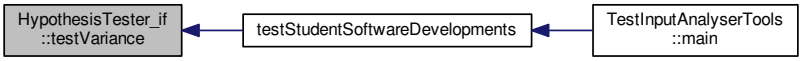
5.26.3.6 virtual double HypothesisTester_if::testProportion (double *confidencelevel,* std::string *secondPopulationDataFilename,* H1Comparition *comp*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

5.26.3.7 virtual double HypothesisTester_if::testVariance (double *confidencelevel,* double *var,* H1Comparition *comp*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

Here is the caller graph for this function:



5.26.3.8 virtual double HypothesisTester_if::testVariance (double *confidencelevel,* std::string *secondPopulationDataFilename,* H1Comparition *comp*) [pure virtual]

Implemented in HypothesisTesterDummyImpl.

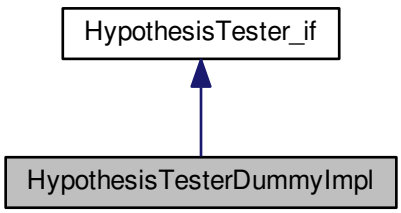
The documentation for this class was generated from the following file:

- HypothesisTester_if.h

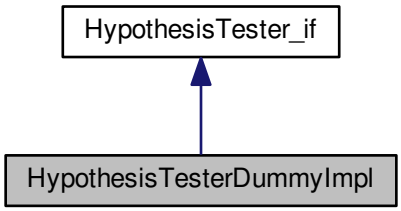
5.27 HypothesisTesterDummyImpl Class Reference

`#include <HypothesisTesterDummyImpl.h>`

Inheritance diagram for HypothesisTesterDummyImpl:



Collaboration diagram for HypothesisTesterDummyImpl:



Public Member Functions

- HypothesisTesterDummyImpl ()
- HypothesisTesterDummyImpl (const HypothesisTesterDummyImpl &orig)
- ~HypothesisTesterDummyImpl ()
- double testAverage (double confidencelevel, double avg, H1Comparition comp)
- double testProportion (double confidencelevel, double prop, H1Comparition comp)
- double testVariance (double confidencelevel, double var, H1Comparition comp)
- double testAverage (double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)
- double testProportion (double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)
- double testVariance (double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)
- void setDataFilename (std::string dataFilename)
- std::string getDataFilename ()

Additional Inherited Members

5.27.1 Constructor & Destructor Documentation

5.27.1.1 HypothesisTesterDummyImpl::HypothesisTesterDummyImpl ( )

5.27.1.2 HypothesisTesterDummyImpl::HypothesisTesterDummyImpl (const HypothesisTesterDummyImpl & *orig*)

5.27.1.3 HypothesisTesterDummyImpl::~HypothesisTesterDummyImpl ( )

5.27.2 Member Function Documentation

5.27.2.1 std::string HypothesisTesterDummyImpl::getDataFilename ( ) [virtual]

Implements HypothesisTester_if.

5.27.2.2 void HypothesisTesterDummyImpl::setDataFilename (std::string *dataFilename*) [virtual]

Implements HypothesisTester_if.

5.27.2.3 double HypothesisTesterDummyImpl::testAverage (double *confidencelevel*, double *avg*, H1Comparition *comp*) [virtual]

Implements HypothesisTester_if.

5.27.2.4 double HypothesisTesterDummyImpl::testAverage (double *confidencelevel*, std::string *secondPopulationDataFilename*, H1Comparition *comp*) [virtual]

Implements HypothesisTester_if.

5.27.2.5 double HypothesisTesterDummyImpl::testProportion (double *confidencelevel*, double *prop*, H1Comparition *comp*) [virtual]

Implements HypothesisTester_if.

5.27.2.6 double HypothesisTesterDummyImpl::testProportion (double *confidencelevel*, std::string *secondPopulationDataFilename*, H1Comparition *comp*) [virtual]

Implements HypothesisTester_if.

5.27.2.7 double HypothesisTesterDummyImpl::testVariance (double *confidencelevel*, double *var*, H1Comparition *comp*) [virtual]

Implements HypothesisTester_if.

5.27.2.8 double HypothesisTesterDummyImpl::testVariance (double *confidencelevel*, std::string *secondPopulationDataFilename*, H1Comparition *comp*) [virtual]

Implements HypothesisTester_if.

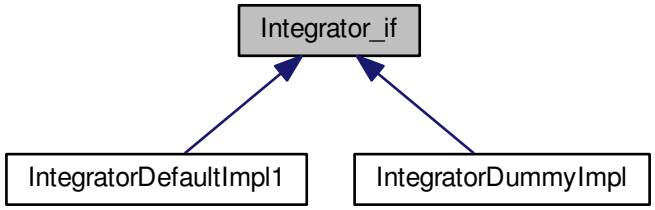
The documentation for this class was generated from the following files:

- HypothesisTesterDummyImpl.h
- HypothesisTesterDummyImpl.cpp

5.28 Integrator_if Class Reference

`#include <Integrator_if.h>`

Inheritance diagram for Integrator_if:



Public Member Functions

- virtual void setPrecision (double e)=0
- virtual double getPrecision ()=0
- virtual double integrate (double min, double max, double(∗f)(double, double), double p2)=0
- virtual double integrate (double min, double max, double(∗f)(double, double, double), double p2, double p3)=0
- virtual double integrate (double min, double max, double(∗f)(double, double, double, double), double p2, double p3, double p4)=0
- virtual double integrate (double min, double max, double(∗f)(double, double, double, double, double), double p2, double p3, double p4, double p5)=0

5.28.1 Member Function Documentation

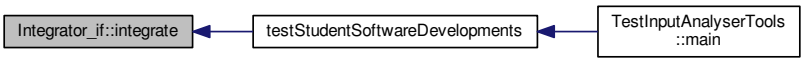
5.28.1.1 virtual double Integrator_if::getPrecision ( ) [pure virtual]

Implemented in IntegratorDefaultImpl1, and IntegratorDummyImpl.

5.28.1.2 virtual double Integrator_if::integrate (double *min*, double *max*, double(∗)(double, double) *f*, double *p2*) [pure virtual]

Implemented in IntegratorDefaultImpl1, and IntegratorDummyImpl.

Here is the caller graph for this function:



5.28.1.3 virtual double Integrator_if::integrate (double *min,* double *max,* double(∗)(double, double, double) *f,* double *p2,* double *p3*) [pure virtual]

Implemented in IntegratorDefaultImpl1, and IntegratorDummyImpl.

5.28.1.4 virtual double Integrator_if::integrate (double *min,* double *max,* double(∗)(double, double, double, double) *f,* double *p2,* double *p3,* double *p4*) [pure virtual]

Implemented in IntegratorDefaultImpl1, and IntegratorDummyImpl.

5.28.1.5 virtual double Integrator_if::integrate (double *min,* double *max,* double(∗)(double, double, double, double, double) *f,* double *p2,* double *p3,* double *p4,* double *p5*) [pure virtual]

Implemented in IntegratorDefaultImpl1, and IntegratorDummyImpl.

5.28.1.6 virtual void Integrator_if::setPrecision (double *e*) [pure virtual]

Implemented in IntegratorDefaultImpl1, and IntegratorDummyImpl.

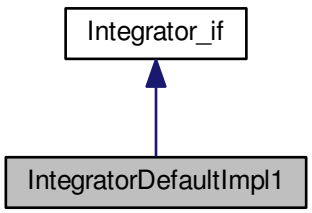
The documentation for this class was generated from the following file:

- Integrator_if.h

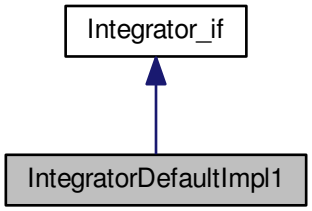
5.29 IntegratorDefaultImpl1 Class Reference

```
#include <IntegratorDefaultImpl1.h>
```

Inheritance diagram for IntegratorDefaultImpl1:



Collaboration diagram for IntegratorDefaultImpl1:



Public Member Functions

- IntegratorDefaultImpl1 ()
- IntegratorDefaultImpl1 (const IntegratorDefaultImpl1 &orig)
- virtual ~IntegratorDefaultImpl1 ()
- virtual void setPrecision (double e)
- virtual double getPrecision ()
- virtual double integrate (double min, double max, double(*f)(double, double), double p2)
- virtual double integrate (double min, double max, double(*f)(double, double, double), double p2, double p3)
- virtual double integrate (double min, double max, double(*f)(double, double, double, double), double p2, double p3, double p4)
- virtual double integrate (double min, double max, double(*f)(double, double, double, double, double), double p2, double p3, double p4, double p5)

5.29.1 Constructor & Destructor Documentation

5.29.1.1 IntegratorDefaultImpl1::IntegratorDefaultImpl1 ( )

5.29.1.2 IntegratorDefaultImpl1::IntegratorDefaultImpl1 (const IntegratorDefaultImpl1 & *orig*)

5.29.1.3 IntegratorDefaultImpl1::~IntegratorDefaultImpl1 ( ) [virtual]

5.29.2 Member Function Documentation

5.29.2.1 double IntegratorDefaultImpl1::getPrecision ( ) [virtual]

Implements Integrator_if.

5.29.2.2 double IntegratorDefaultImpl1::integrate (double *min*, double *max*, double(*)(double, double) *f*, double *p2*) [virtual]

Implements Integrator_if.

5.29.2.3 double IntegratorDefaultImpl1::integrate (double *min*, double *max*, double(∗)(double, double, double) *f*, double *p2*, double *p3*) [virtual]

Implements Integrator_if.

5.29.2.4 double IntegratorDefaultImpl1::integrate (double *min*, double *max*, double(∗)(double, double, double, double) *f*, double *p2*, double *p3*, double *p4*) [virtual]

Implements Integrator_if.

5.29.2.5 double IntegratorDefaultImpl1::integrate (double *min*, double *max*, double(∗)(double, double, double, double, double) *f*, double *p2*, double *p3*, double *p4*, double *p5*) [virtual]

Implements Integrator_if.

5.29.2.6 void IntegratorDefaultImpl1::setPrecision (double *e*) [virtual]

Implements Integrator_if.

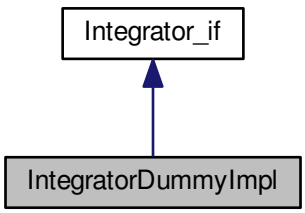
The documentation for this class was generated from the following files:

- IntegratorDefaultImpl1.h
- IntegratorDefaultImpl1.cpp

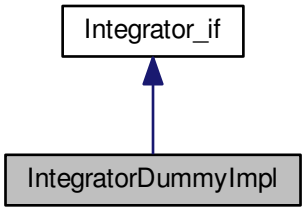
5.30 IntegratorDummyImpl Class Reference

`#include <IntegratorDummyImpl.h>`

Inheritance diagram for IntegratorDummyImpl:



Collaboration diagram for IntegratorDummyImpl:



Public Member Functions

- IntegratorDummyImpl ()
- IntegratorDummyImpl (const IntegratorDummyImpl &orig)
- ∼IntegratorDummyImpl ()
- void setPrecision (double e)
- double getPrecision ()
- double integrate (double min, double max, double(∗f)(double, double), double p2)
- double integrate (double min, double max, double(∗f)(double, double, double), double p2, double p3)
- double integrate (double min, double max, double(∗f)(double, double, double, double), double p2, double p3, double p4)
- double integrate (double min, double max, double(∗f)(double, double, double, double, double), double p2, double p3, double p4, double p5)

5.30.1 Constructor & Destructor Documentation

5.30.1.1 IntegratorDummyImpl::IntegratorDummyImpl ( )

5.30.1.2 IntegratorDummyImpl::IntegratorDummyImpl (const IntegratorDummyImpl & *orig*)

5.30.1.3 IntegratorDummyImpl::∼IntegratorDummyImpl ( )

5.30.2 Member Function Documentation

5.30.2.1 double IntegratorDummyImpl::getPrecision ( ) [virtual]

Implements Integrator_if.

5.30.2.2 double IntegratorDummyImpl::integrate (double *min*, double *max*, double(∗)(double, double) *f*, double *p2*) [virtual]

Implements Integrator_if.

5.30.2.3 double IntegratorDummyImpl::integrate (double *min*, double *max*, double(∗)(double, double, double) *f*, double *p2*, double *p3*) [virtual]

Implements Integrator_if.

5.30.2.4 double IntegratorDummyImpl::integrate (double *min*, double *max*, double(∗)(double, double, double, double) *f*, double *p2*, double *p3*, double *p4*) [virtual]

Implements Integrator_if.

5.30.2.5 double IntegratorDummyImpl::integrate (double *min*, double *max*, double(∗)(double, double, double, double, double) *f*, double *p2*, double *p3*, double *p4*, double *p5*) [virtual]

Implements Integrator_if.

5.30.2.6 void IntegratorDummyImpl::setPrecision (double *e*) [virtual]

Implements Integrator_if.

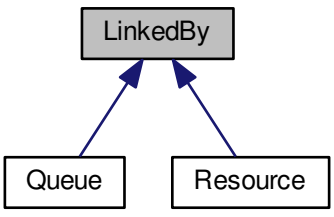
The documentation for this class was generated from the following files:

- IntegratorDummyImpl.h
- IntegratorDummyImpl.cpp

5.31 LinkedBy Class Reference

`#include <LinkedBy.h>`

Inheritance diagram for LinkedBy:



Public Member Functions

- LinkedBy ()
- LinkedBy (const LinkedBy &orig)
- virtual ~LinkedBy ()
- void addLink ()
- void removeLink ()
- bool isLinked ()

5.31.1 Constructor & Destructor Documentation

5.31.1.1 LinkedBy::LinkedBy ( )

5.31.1.2 LinkedBy::LinkedBy (const LinkedBy & *orig*)

5.31.1.3 LinkedBy::~LinkedBy ( ) [virtual]

5.31.2 Member Function Documentation

5.31.2.1 void LinkedBy::addLink ( )

5.31.2.2 bool LinkedBy::isLinked ( )

5.31.2.3 void LinkedBy::removeLink ( )

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

- LinkedBy.h
- LinkedBy.cpp

5.32 List< T > Class Template Reference

`#include <List.h>`

Public Types

- using CompFunct = std::function< bool(const T, const T) >

Public Member Functions

- List ()
- List (const List &orig)
- virtual ~List ()
- unsigned int size ()
- bool empty ()
- void clear ()
- void pop_front ()
- template<class Compare >
  void sort (Compare comp)
- std::list< T > * getList () const
- T create ()
- template<typename U >
  T create (U arg)
- std::string show ()
- std::list< T >::iterator find (T element)
- void insert (T element)
- void remove (T element)
- T next ()
- T first ()
- T last ()
- T previous ()
- T actual ()
- void setSortFunc (CompFunct _sortFunc)

5.32.1 Member Typedef Documentation

5.32.1.1 template<typename T> using List< T >::CompFunct = std::function<bool(const T, const T) >

5.32.2 Constructor & Destructor Documentation

5.32.2.1 template<typename T > List< T >::List ( )

5.32.2.2 template<typename T > List< T >::List (const List< T > & *orig*)

5.32.2.3 template<typename T > List< T >::~List ( ) [virtual]

5.32.3 Member Function Documentation

5.32.3.1 template<typename T > T List< T >::actual ( )

5.32.3.2 template<typename T > void List< T >::clear ( )

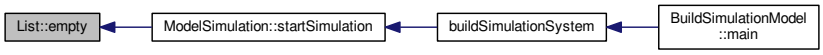
Here is the caller graph for this function:

5.32.3.3 template<typename T > T List< T >::create ( )

5.32.3.4 template<typename T > template<typename U > T List< T >::create (U *arg*)

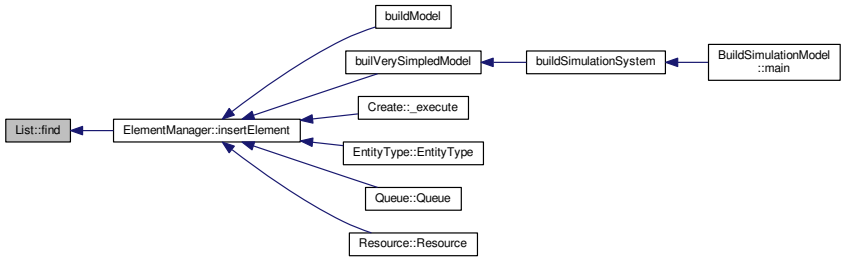
5.32.3.5 template<typename T > bool List< T >::empty ( )

Here is the caller graph for this function:



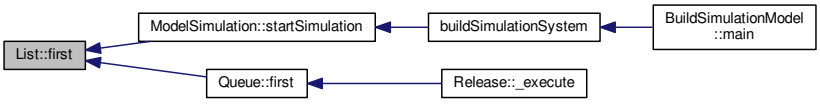
5.32.3.6 template<typename T> std::list< T >::iterator List< T >::find (T *element*)

Here is the caller graph for this function:



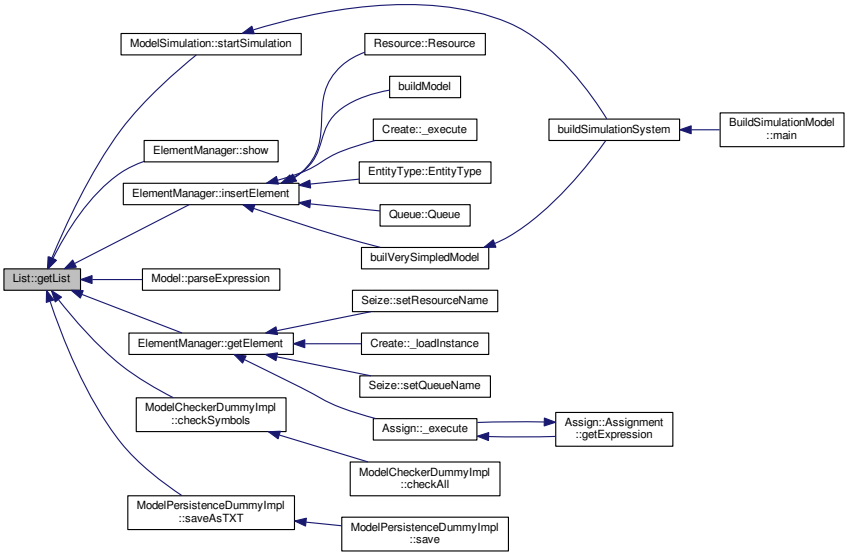
5.32.3.7 template<typename T > T List< T >::first ( )

Here is the caller graph for this function:



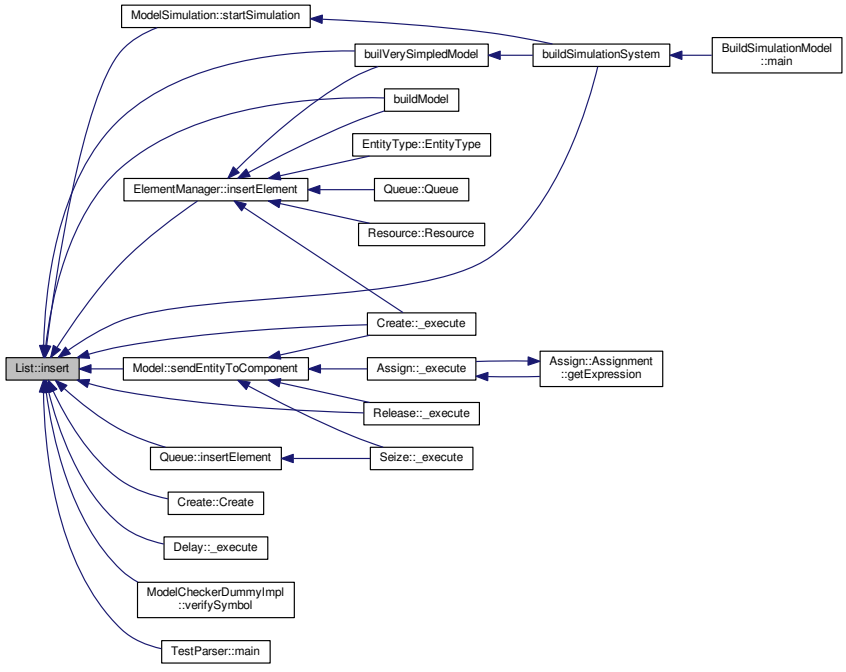
5.32.3.8 template<typename T > std::list< T > * List< T >::getList ( ) const

Here is the caller graph for this function:



5.32.3.9 template<typename T> void List< T >::insert (T *element*)

Here is the caller graph for this function:

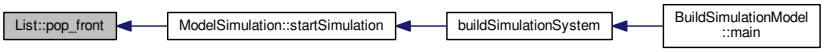


5.32.3.10 template<typename T > T List< T >::last ( )

5.32.3.11 template<typename T > T List< T >::next ( )

5.32.3.12 template<typename T > void List< T >::pop_front ( )

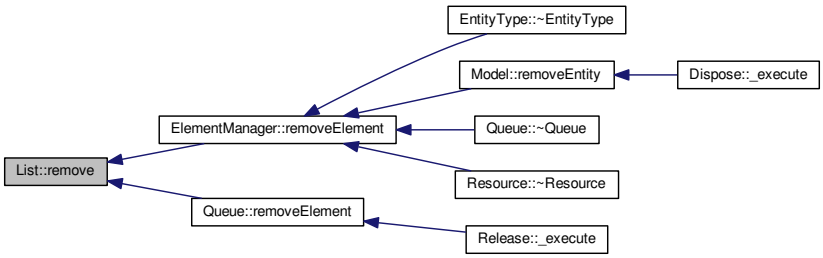
Here is the caller graph for this function:



5.32.3.13 template<typename T > T List< T >::previous ( )

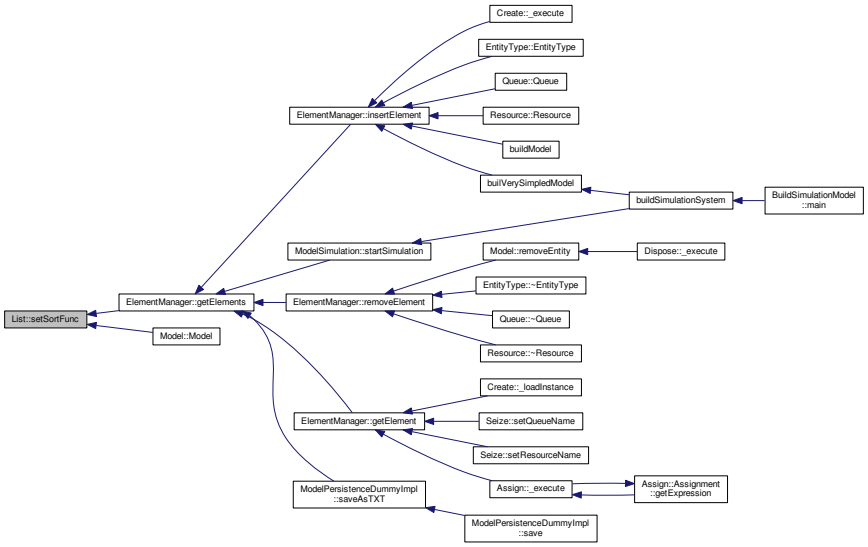
5.32.3.14 template<typename T> void List< T >::remove (T *element*)

Here is the caller graph for this function:



5.32.3.15 template<typename T > void List< T >::setSortFunc (CompFunct *_sortFunc*)

Here is the caller graph for this function:

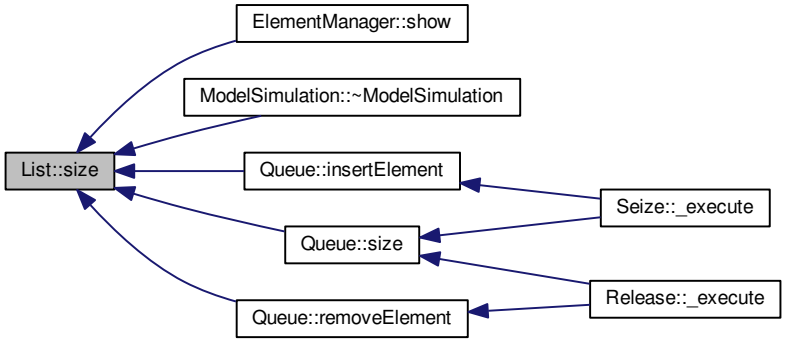


5.32.3.16 template<typename T > std::string List< T >::show ( )

Here is the caller graph for this function:

5.32.3.17 template<typename T > unsigned int List< T >::size ( )

Here is the caller graph for this function:



5.32.3.18 template<typename T > template<class Compare > void List< T >::sort (Compare *comp*)

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

- List.h

5.33 Model Class Reference

`#include <Model.h>`

Public Member Functions

- Model (Simulator *simulator)
- Model (const Model &orig)
- virtual ~Model ()
- void showReports ()
- bool saveModel (std::string filename)
- bool loadModel (std::string filename)
- bool checkModel ()

  Checks the integrity and consistency of the model, possibly corrects some inconsistencies, and returns if the model is in position to the simulated.
- bool verifySymbol (std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)

  Verifies if a symbol defined in a component (ModelComponent) or element is syntactically valid and addresses existing components or elements. It's used only by and directed by the component that defines the symbol.
- void removeEntity (Entity *entity, bool collectStatistics)
- void sendEntityToComponent (Entity *entity, ModelComponent *component, double timeDelay)

Used by components (ModelComponent) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event (Event).

- double parseExpression (const std::string expression)
- double parseExpression (const std::string expression, bool *success, std::string *errorMessage)
- Util::identitifcation getId () const
- List< SimulationControl * > * getControls () const

  Returns a list of values that can be externally controlled (changed). They usually correspond to input parameters in the simulation model that must be changed for an experimental design.
- List< SimulationResponse * > * getResponses () const

  Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.
- TraceManager * getTracer () const

  Provides access to the class that performs the trace of simulation and replications.
- OnEventManager * getOnEventManager () const
- ElementManager * getElementManager () const

  Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).
- ModelInfo * getInfos () const
- Simulator * getParent () const
- ModelSimulation * getSimulation () const

  Provides access to the class that manages the model simulation.
- List< ModelComponent * > * getComponents () const

  Returns the list of components (such as Create, Delay, Dispose, etc.) that make up the simulation model.
- List< Event * > * getEvents () const

  The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components (SourceComponentModel).

5.33.1 Detailed Description

Model is probably the most important class of Genesys kernel. It represents a discrete event-driven simulation model. Each model is responsible for controlling its own simulation, ie, for sequentially processing events and collecting statistical results. A model is mainly represented by a collection of components (ModelComponent), adequately configurated and connected, and a collection of under layered element (ModelElement).

5.33.2 Constructor & Destructor Documentation

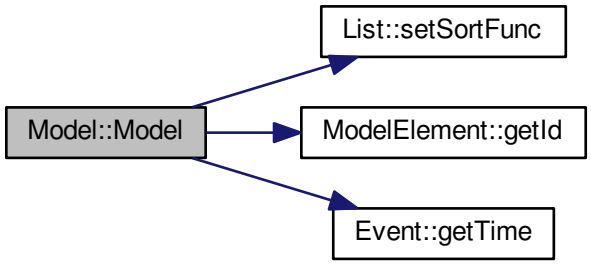
5.33.2.1 Model::Model (Simulator * *simulator*)

Components are sorted by ID

The future events list must be chronologicaly sorted

Events are sorted chronologically

Here is the call graph for this function:



5.33.2.2 Model::Model (const Model & *orig*)

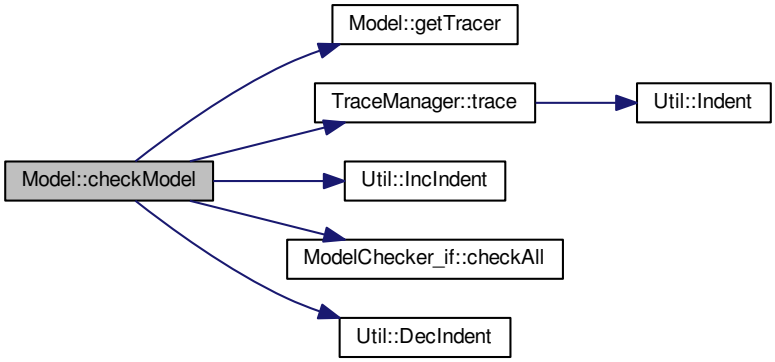
5.33.2.3 Model::~Model ( ) [virtual]

5.33.3 Member Function Documentation

5.33.3.1 bool Model::checkModel ( )

Checks the integrity and consistency of the model, possibly corrects some inconsistencies, and returns if the model is in position to the simulated.

Here is the call graph for this function:

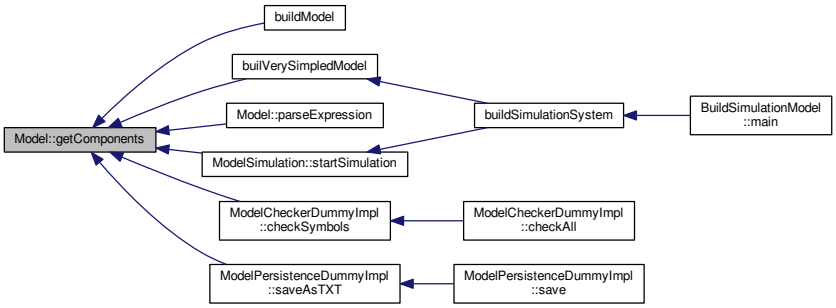


Here is the caller graph for this function:

5.33.3.2 List< ModelComponent * > * Model::getComponents ( ) const

Returns the list of components (such as Create, Delay, Dispose, etc.) that make up the simulation model.

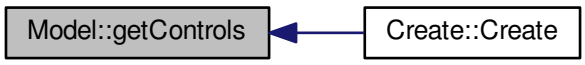
Here is the caller graph for this function:



5.33.3.3 List< SimulationControl * > * Model::getControls ( ) const

Returns a list of values that can be externally controlled (changed). They usually correspond to input parameters in the simulation model that must be changed for an experimental design.

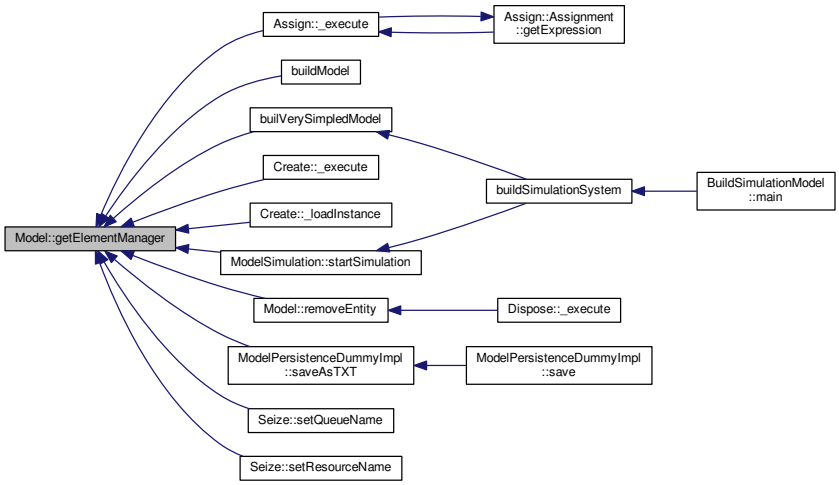
Here is the caller graph for this function:



5.33.3.4 ElementManager * Model::getElementManager ( ) const

Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).

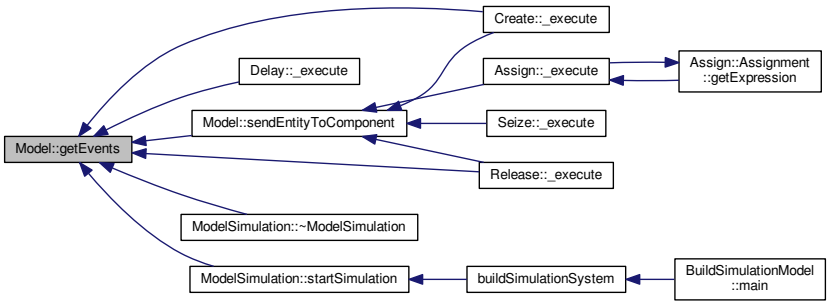
Here is the caller graph for this function:



5.33.3.5 List< Event * > * Model::getEvents ( ) const

The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components (SourceComponentModel).

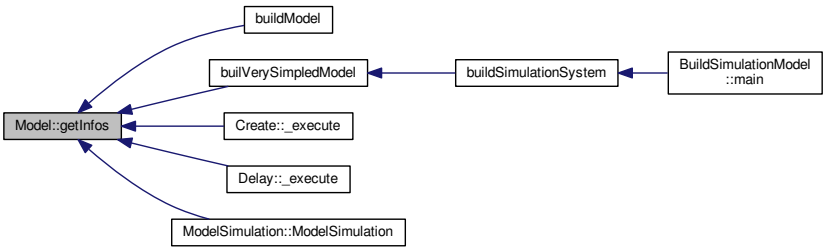
Here is the caller graph for this function:



5.33.3.6 Util::identitifcation Model::getId ( ) const

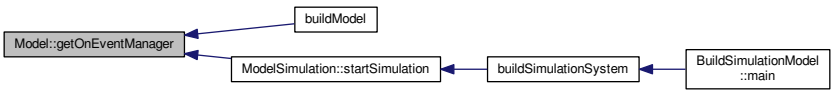
5.33.3.7 ModelInfo * Model::getInfos ( ) const

Here is the caller graph for this function:



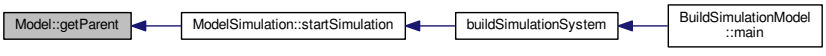
5.33.3.8 OnEventManager * Model::getOnEventManager ( ) const

Here is the caller graph for this function:



5.33.3.9 Simulator * Model::getParent ( ) const

Here is the caller graph for this function:



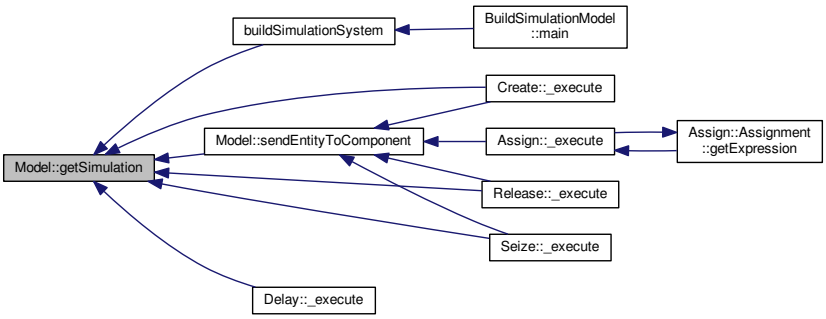
5.33.3.10 List< SimulationResponse * > * Model::getResponses ( ) const

Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.

5.33.3.11 ModelSimulation * Model::getSimulation ( ) const

Provides access to the class that manages the model simulation.

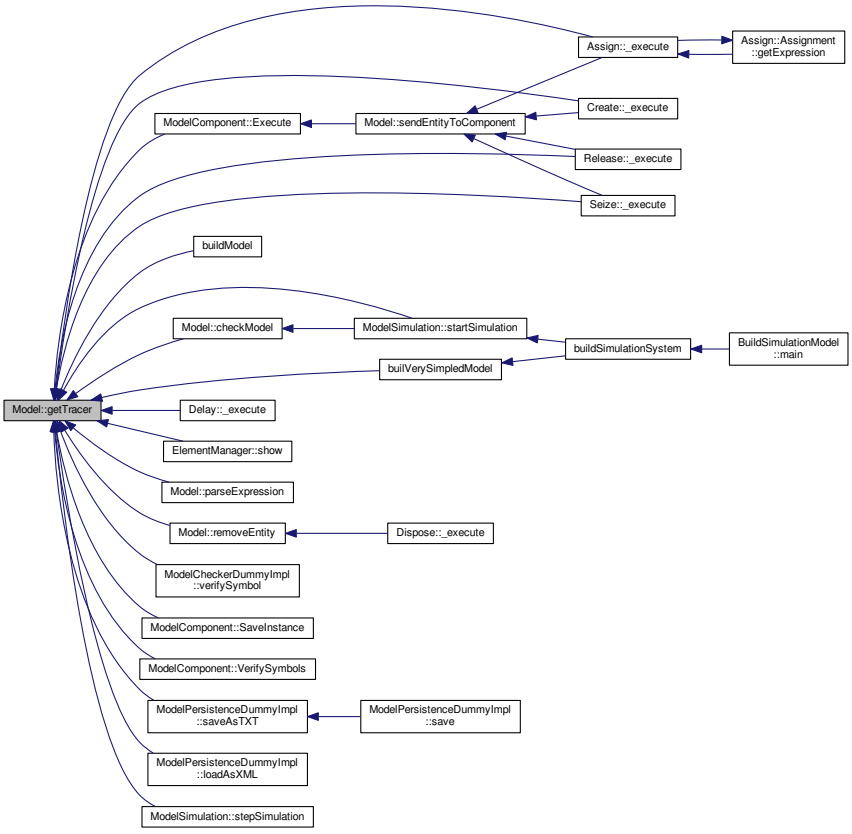
Here is the caller graph for this function:



5.33.3.12 TraceManager * Model::getTracer ( ) const

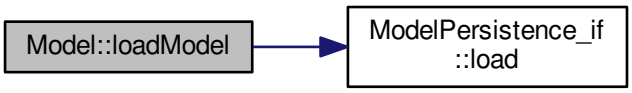
Provides access to the class that performs the trace of simulation and replications.

Here is the caller graph for this function:



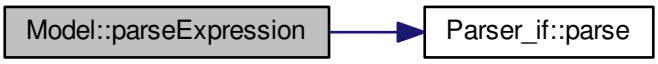
5.33.3.13 bool Model::loadModel (std::string *filename*)

Here is the call graph for this function:

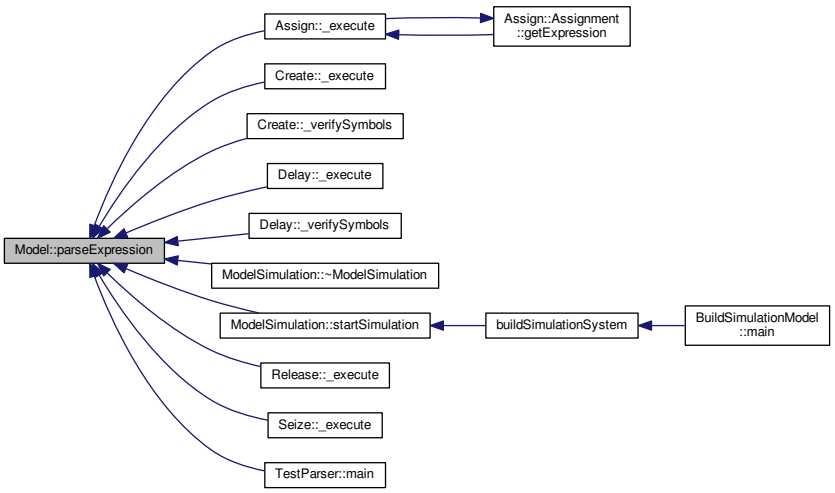


5.33.3.14 double Model::parseExpression (const std::string *expression*)

Here is the call graph for this function:

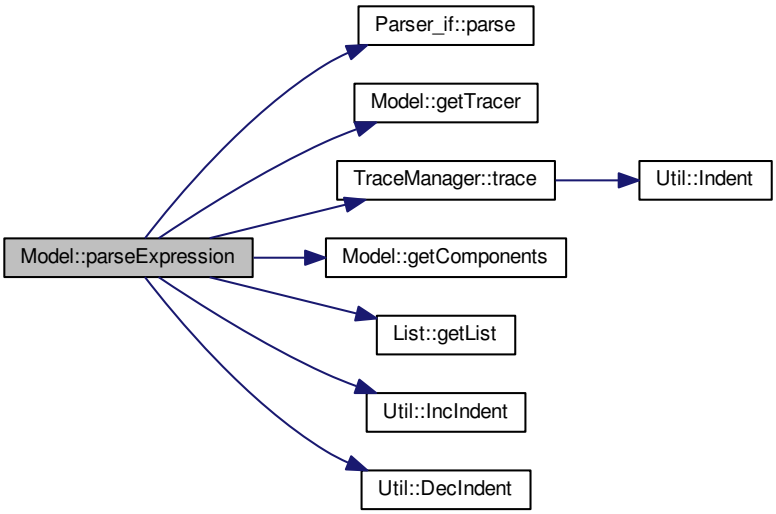


Here is the caller graph for this function:



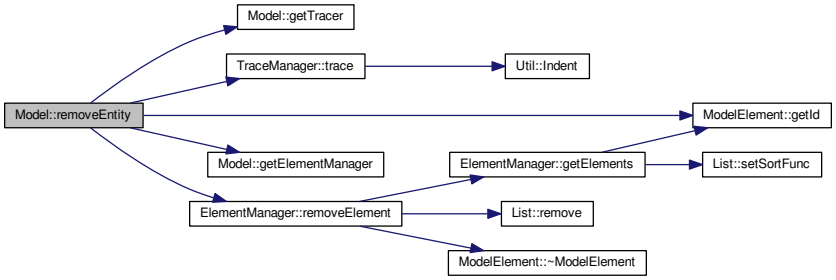
5.33.3.15 double Model::parseExpression (const std::string *expression*, bool * *success*, std::string * *errorMessage*)

Here is the call graph for this function:



5.33.3.16 void Model::removeEntity (Entity * *entity*, bool *collectStatistics*)

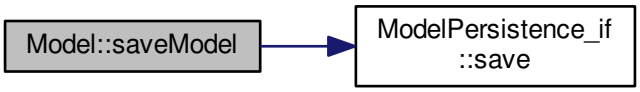
Here is the call graph for this function:



Here is the caller graph for this function:

5.33.3.17 bool Model::saveModel (std::string *filename*)

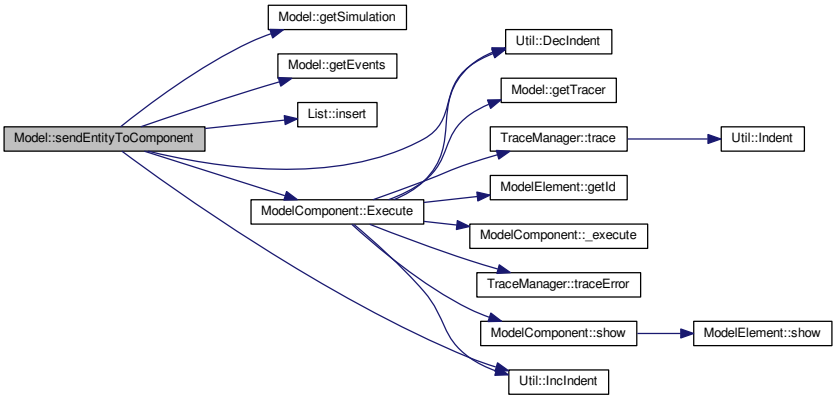
Here is the call graph for this function:



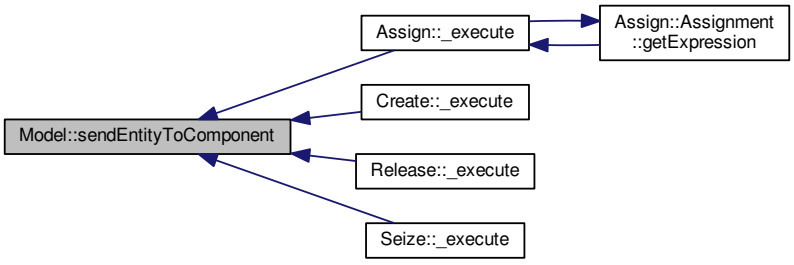
5.33.3.18 void Model::sendEntityToComponent (Entity * *entity,* ModelComponent * *component,* double *timeDelay*)

Used by components (ModelComponent) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event (Event).

Here is the call graph for this function:



Here is the caller graph for this function:



5.33.3.19 void Model::showReports ( )

5.33.3.20 bool Model::verifySymbol (std::string *componentName,* std::string *expressionName,* std::string *expression,* std::string *expressionResult,* bool *mandatory*)

Verifies if a symbol defined in a component (ModelComponent) or element is syntactically valid and addresses existing components or elements. It's used only by and directed by the component that defines the symbol.

Here is the call graph for this function:

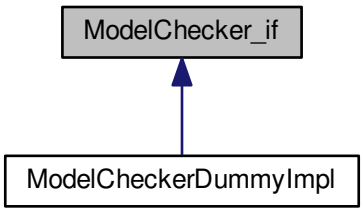
The documentation for this class was generated from the following files:

- Model.h
- Model.cpp

5.34 ModelChecker_if Class Reference

`#include <ModelChecker_if.h>`

Inheritance diagram for ModelChecker_if:



Public Member Functions

- virtual bool checkAll ()=0
- virtual bool checkAndAddInternalLiterals ()=0
- virtual bool checkConnected ()=0
- virtual bool checkSymbols ()=0
- virtual bool checkPathway ()=0
- virtual bool checkActivationCode ()=0
- virtual bool verifySymbol (std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)=0

5.34.1 Detailed Description

The ModelChecker is responsable for verifying the model consistency, fixing inconsistencies wheneaver possible

5.34.2 Member Function Documentation

5.34.2.1 virtual bool ModelChecker_if::checkActivationCode ( ) [pure virtual]

Check if components forms a valid pathway, including logical connections, such as routes, statios and pickups, for example

Implemented in ModelCheckerDummyImpl.

5.34.2.2 virtual bool ModelChecker_if::checkAll ( ) [pure virtual]

Implemented in ModelCheckerDummyImpl.

Here is the caller graph for this function:

5.34.2.3 virtual bool ModelChecker_if::checkAndAddInternalLiterals ( ) [pure virtual]

Invoques all other checks and returns true only if all of them returned true

Implemented in ModelCheckerDummyImpl.

5.34.2.4 virtual bool ModelChecker_if::checkConnected ( ) [pure virtual]

Implemented in ModelCheckerDummyImpl.

5.34.2.5 virtual bool ModelChecker_if::checkPathway ( ) [pure virtual]

Checks if user-defined strings for symbols required by components, usually expressions or functions, are valid or references existing and valid elements.

Implemented in ModelCheckerDummyImpl.

5.34.2.6 virtual bool ModelChecker_if::checkSymbols ( ) [pure virtual]

Checks if components are consistently connected to other to form a valid process-oriented model, describing how entities proceed to the flow

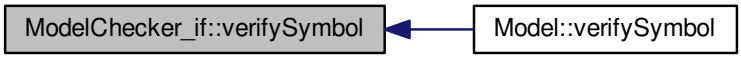
Implemented in ModelCheckerDummyImpl.

5.34.2.7 virtual bool ModelChecker_if::verifySymbol (std::string *componentName,* std::string *expressionName,* std::string *expression,* std::string *expressionResult,* bool *mandatory*) [pure virtual]

unnecessary

Implemented in ModelCheckerDummyImpl.

Here is the caller graph for this function:



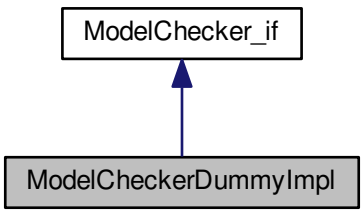
The documentation for this class was generated from the following file:

- ModelChecker_if.h

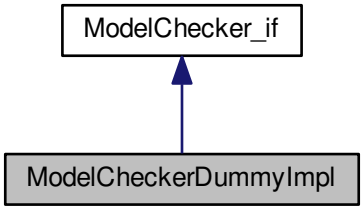
5.35 ModelCheckerDummyImpl Class Reference

`#include <ModelCheckerDummyImpl.h>`

Inheritance diagram for ModelCheckerDummyImpl:



Collaboration diagram for ModelCheckerDummyImpl:



Public Member Functions

- ModelCheckerDummyImpl (Model ∗model)
- ModelCheckerDummyImpl (const ModelCheckerDummyImpl &orig)
- ∼ModelCheckerDummyImpl ()
- bool checkAll ()
- bool checkAndAddInternalLiterals ()
- bool checkConnected ()
- bool checkSymbols ()
- bool checkPathway ()
- bool checkActivationCode ()
- bool verifySymbol (std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)

5.35.1 Detailed Description

Just an example of possible implementation of the ModelChecker interface. Developers can implement their own class

5.35.2 Constructor & Destructor Documentation

5.35.2.1 ModelCheckerDummyImpl::ModelCheckerDummyImpl (Model ∗ *model*)

5.35.2.2 ModelCheckerDummyImpl::ModelCheckerDummyImpl (const ModelCheckerDummyImpl & *orig*)

5.35.2.3 ModelCheckerDummyImpl::∼ModelCheckerDummyImpl ( )

5.35.3 Member Function Documentation

5.35.3.1 bool ModelCheckerDummyImpl::checkActivationCode ( ) [virtual]

Check if components forms a valid pathway, including logical connections, such as routes, statios and pickups, for example

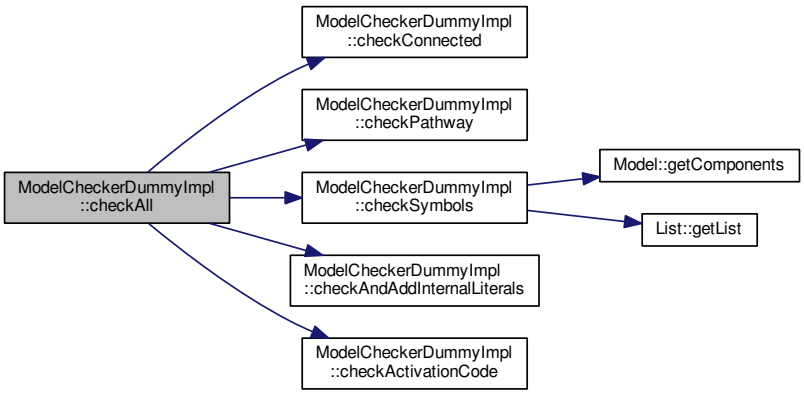
Implements ModelChecker_if.

Here is the caller graph for this function:

5.35.3.2 bool ModelCheckerDummyImpl::checkAll ( ) [virtual]

Implements ModelChecker_if.

Here is the call graph for this function:

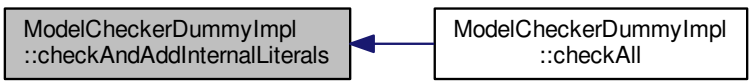


5.35.3.3 bool ModelCheckerDummyImpl::checkAndAddInternalLiterals ( ) [virtual]

Invoques all other checks and returns true only if all of them returned true

Implements ModelChecker_if.

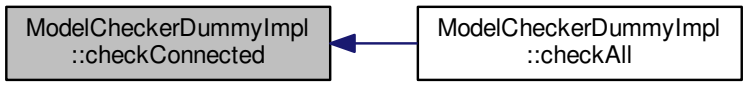
Here is the caller graph for this function:



5.35.3.4 bool ModelCheckerDummyImpl::checkConnected ( ) [virtual]

Implements ModelChecker_if.

Here is the caller graph for this function:

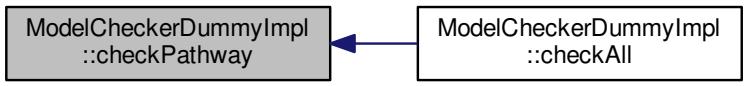


5.35.3.5 bool ModelCheckerDummyImpl::checkPathway ( ) [virtual]

Checks if user-defined strings for symbols required by components, usually expressions or functions, are valid or references existing and valid elements.

Implements ModelChecker_if.

Here is the caller graph for this function:

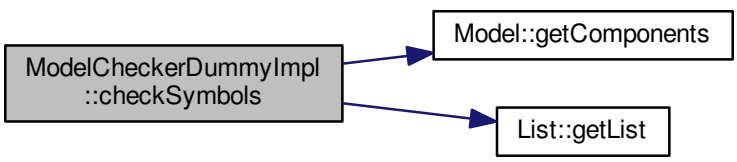


5.35.3.6 bool ModelCheckerDummyImpl::checkSymbols ( ) [virtual]

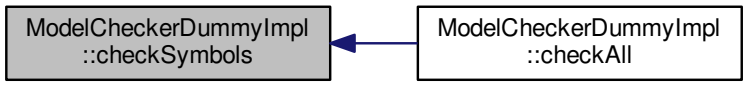
Checks if components are consistently connected to other to form a valid process-oriented model, describing how entities proceed to the flow

Implements ModelChecker_if.

Here is the call graph for this function:



Here is the caller graph for this function:

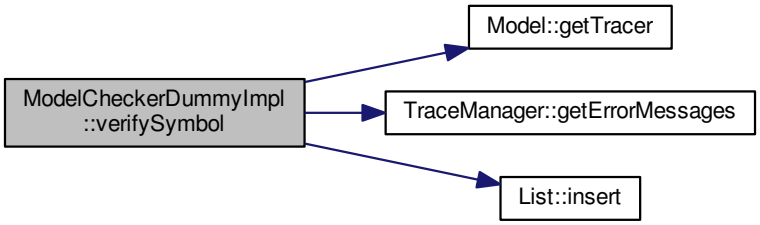


5.35.3.7 bool ModelCheckerDummyImpl::verifySymbol (std::string *componentName,* std::string *expressionName,* std::string *expression,* std::string *expressionResult,* bool *mandatory*) [virtual]

unnecessary

Implements ModelChecker_if.

Here is the call graph for this function:



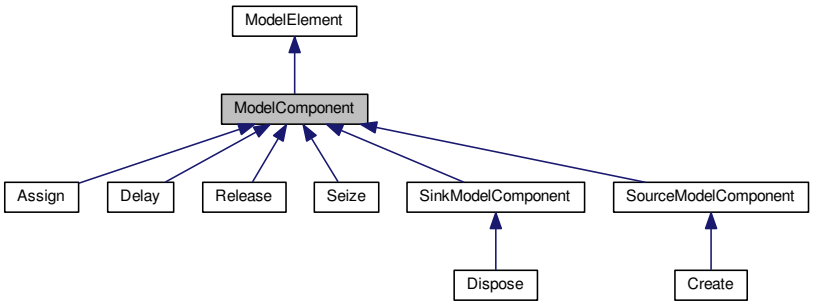
The documentation for this class was generated from the following files:

- ModelCheckerDummyImpl.h
- ModelCheckerDummyImpl.cpp

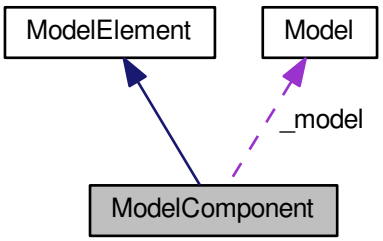
5.36 ModelComponent Class Reference

```
#include <ModelComponent.h>
```

Inheritance diagram for ModelComponent:



Collaboration diagram for ModelComponent:



Public Member Functions

- ModelComponent (Model ∗model)
- ModelComponent (const ModelComponent &orig)
- virtual ∼ModelComponent ()
- virtual std::string show ()
- List< ModelComponent ∗ > ∗ getNextComponents () const

  Returns a list of components directly connected to the output. Usually the components have a single output, but they may have none (such as Dispose) or more than one (as Decide).

Static Public Member Functions

- static void Execute (Entity ∗entity, ModelComponent ∗component)

  This method triggers the simulation of the behavior of the component. It is invoked when an event (corresponding to this component) is taken from the list of future events or when an entity arrives at this component by connection.
- static bool VerifySymbols (ModelComponent ∗component, std::string ∗errorMessage)
- static std::list< std::string > ∗ SaveInstance (ModelComponent ∗component)

Protected Member Functions

- virtual void _execute (Entity *entity)=0
- virtual std::list< std::string > * _saveInstance ()
- virtual std::list< std::string > * _saveInstance (std::string type)

Protected Attributes

- Model * _model

5.36.1 Detailed Description

Um componente do modelo é um bloco que representa um comportamento específico a ser simulado. O comportamento é disparado quando uma entidade chega ao componente, o que corresponde à ocorrência de um evento. Um modelo de simulação corresponde a um conjunto de componentes interconectados para formar o processo pelo qual a entidade é submetida.

Parameters

model	The model this component belongs to

5.36.2 Constructor & Destructor Documentation

5.36.2.1 ModelComponent::ModelComponent (Model * *model*)

5.36.2.2 ModelComponent::ModelComponent (const ModelComponent & *orig*)

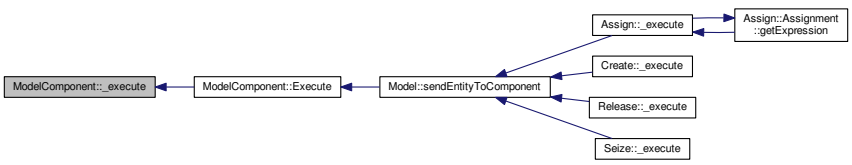
5.36.2.3 ModelComponent::~ModelComponent ( ) [virtual]

5.36.3 Member Function Documentation

5.36.3.1 virtual void ModelComponent::_execute (Entity * *entity*) [protected], [pure virtual]

Implemented in Assign, Seize, Release, Create, Delay, and Dispose.

Here is the caller graph for this function:



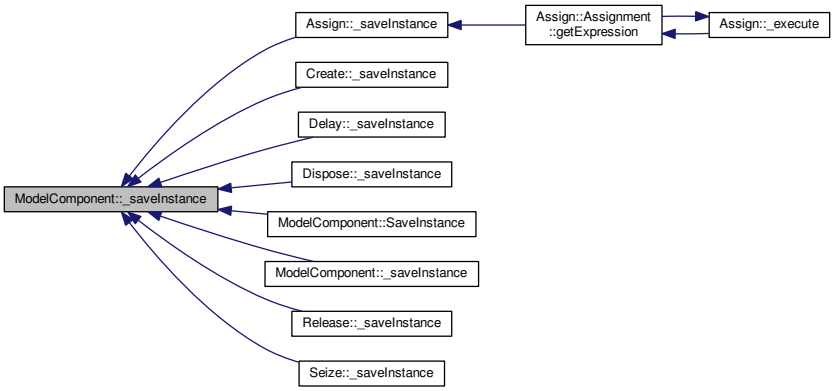
5.36.3.2 std::list< std::string > * ModelComponent::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Reimplemented in Assign, Seize, Release, Create, Delay, and Dispose.

Here is the call graph for this function:

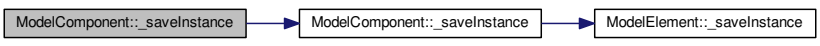
Here is the caller graph for this function:



5.36.3.3 std::list< std::string > * ModelComponent::_saveInstance (std::string *type*) [protected], [virtual]

Reimplemented from ModelElement.

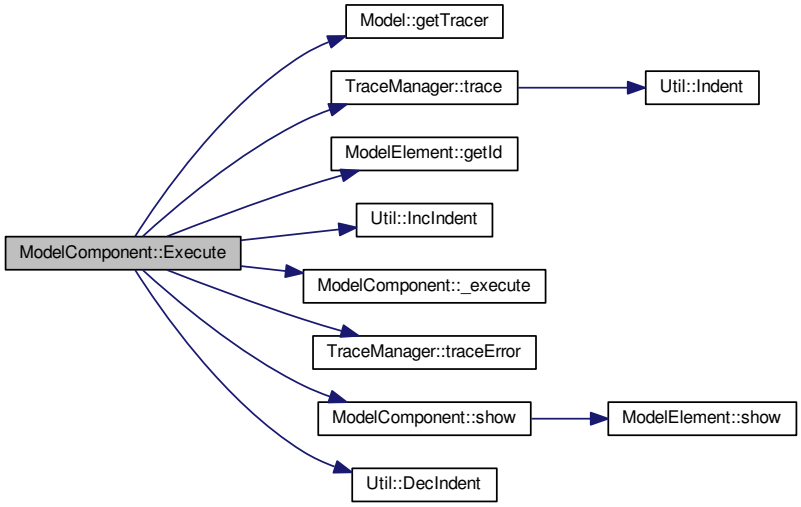
Here is the call graph for this function:



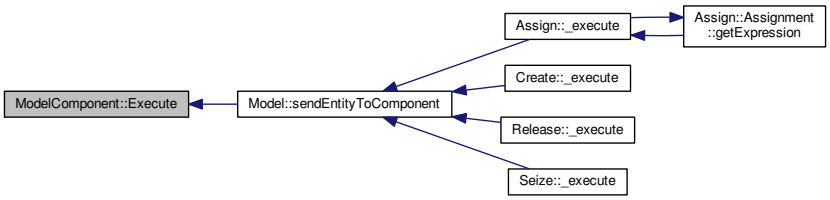
5.36.3.4 void ModelComponent::Execute (Entity * *entity*, ModelComponent * *component*) [static]

This method triggers the simulation of the behavior of the component. It is invoked when an event (corresponding to this component) is taken from the list of future events or when an entity arrives at this component by connection.

Here is the call graph for this function:



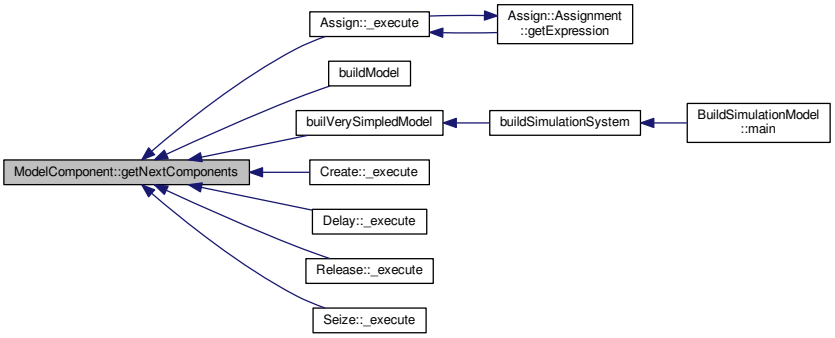
Here is the caller graph for this function:



5.36.3.5 List< ModelComponent * > * ModelComponent::getNextComponents ( ) const

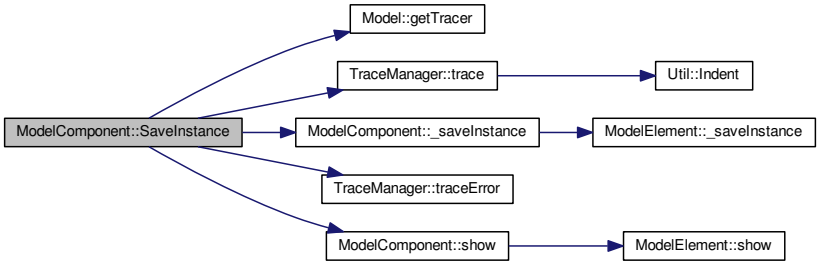
Returns a list of components directly connected to the output. Usually the components have a single output, but they may have none (such as Dispose) or more than one (as Decide).

Here is the caller graph for this function:



5.36.3.6 std::list< std::string > * ModelComponent::SaveInstance (ModelComponent * *component*) [static]

Here is the call graph for this function:

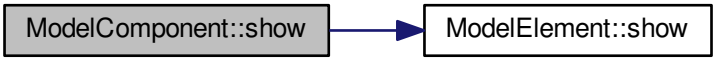


5.36.3.7 std::string ModelComponent::show ( ) [virtual]

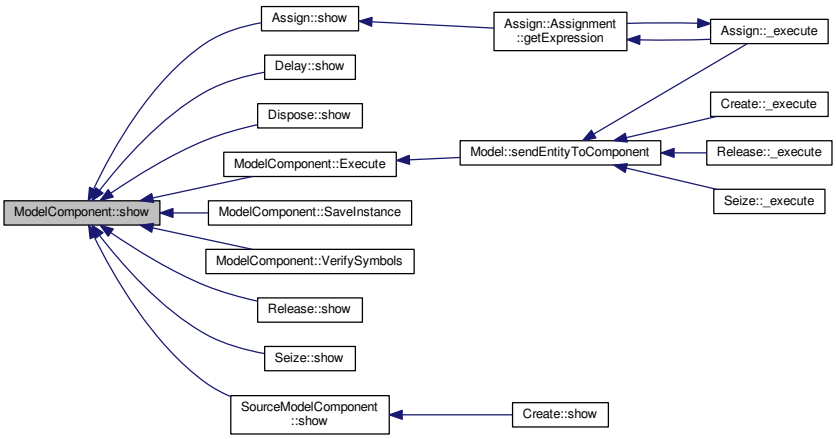
Reimplemented from ModelElement.

Reimplemented in Assign, SourceModelComponent, Seize, Create, Delay, Release, and Dispose.

Here is the call graph for this function:

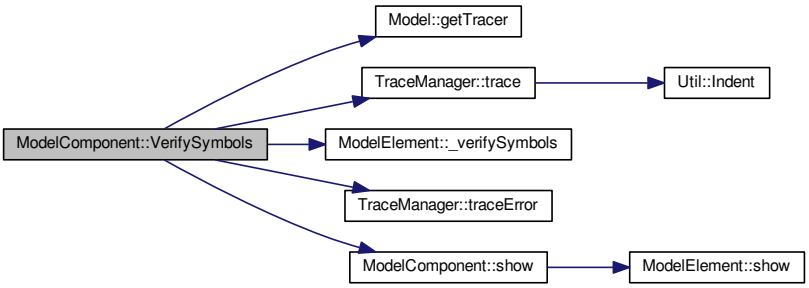


Here is the caller graph for this function:



5.36.3.8 bool ModelComponent::VerifySymbols (ModelComponent * *component*, std::string * *errorMessage*) [static]

Here is the call graph for this function:



5.36.4 Member Data Documentation

5.36.4.1 Model* ModelComponent::_model [protected]

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

- ModelComponent.h
- ModelComponent.cpp

5.37 ModelComponentManager_if Class Reference

`#include <ModelComponentManager_if.h>`

Public Member Functions

- ModelComponentManager_if ()
- ModelComponentManager_if (const ModelComponentManager_if &orig)
- virtual ~ModelComponentManager_if ()

5.37.1 Constructor & Destructor Documentation

5.37.1.1 ModelComponentManager_if::ModelComponentManager_if ( )

5.37.1.2 ModelComponentManager_if::ModelComponentManager_if (const ModelComponentManager_if & *orig*)

5.37.1.3 virtual ModelComponentManager_if::~ModelComponentManager_if ( ) [virtual]

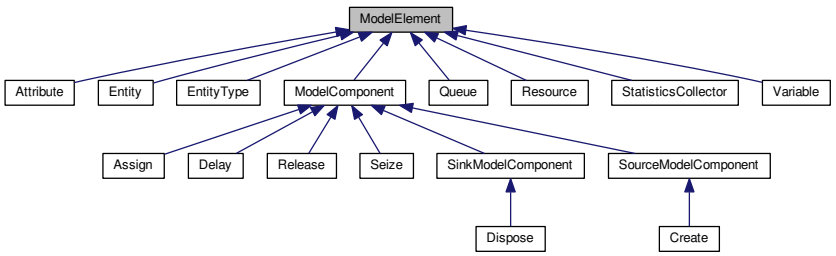
The documentation for this class was generated from the following file:

- ModelComponentManager_if.h

5.38 ModelElement Class Reference

`#include <ModelElement.h>`

Inheritance diagram for ModelElement:



Public Member Functions

- ModelElement (std::string elementTypename)
- ModelElement (const ModelElement &orig)
- virtual ~ModelElement ()
- virtual std::string show ()
- Util::identitifcation getId () const
- void setName (std::string _name)
- std::string getName () const

Static Public Member Functions

- static void LoadInstance (std::list< std::string > words)
- static std::list< std::string > * SaveInstance (ModelElement *element)
- static bool VerifySymbols (ModelElement *element, std::string *errorMessage)

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)=0
- virtual std::list< std::string > * _saveInstance ()
- virtual std::list< std::string > * _saveInstance (std::string type)
- virtual bool _verifySymbols (std::string *errorMessage)=0

Protected Attributes

- Util::identitifcation _id
- std::string _name

5.38.1 Detailed Description

This class is the basis for any element of the model (such as Queue, Resource, Variable, etc.) and also for any component of the model. It has the infrastructure to read and write on file and to verify symbols.

5.38.2 Constructor & Destructor Documentation

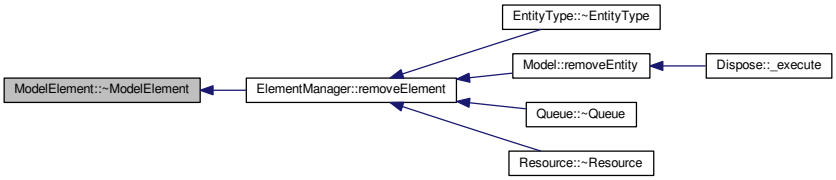
5.38.2.1 ModelElement::ModelElement (std::string *elementTypename*)

Here is the call graph for this function:

5.38.2.2 ModelElement::ModelElement (const ModelElement & *orig*)

5.38.2.3 ModelElement::~ModelElement ( ) [virtual]

Here is the caller graph for this function:



5.38.3 Member Function Documentation

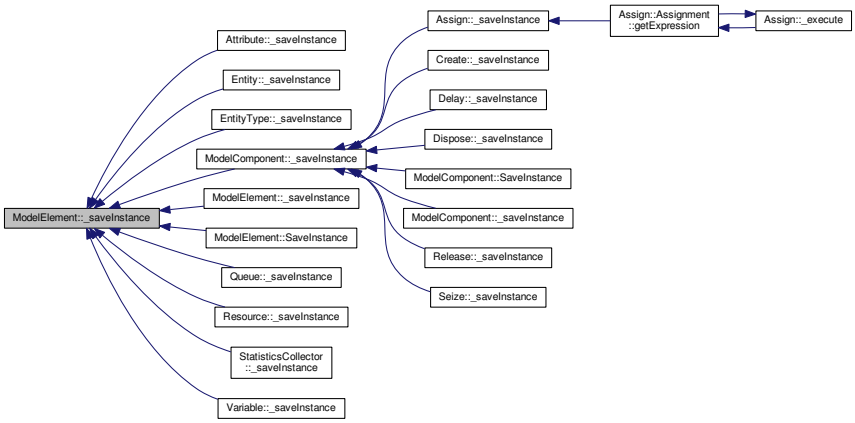
5.38.3.1 virtual void ModelElement::_loadInstance (std::list< std::string > *words*) [protected], [pure virtual]

Implemented in Assign, Resource, Seize, Queue, EntityType, Release, Entity, Variable, Create, Delay, StatisticsCollector, Attribute, and Dispose.

5.38.3.2 std::list< std::string > * ModelElement::_saveInstance ( ) [protected], [virtual]

Reimplemented in Assign, Resource, ModelComponent, Seize, Queue, EntityType, Release, Entity, Variable, Create, Delay, StatisticsCollector, Attribute, and Dispose.

Here is the caller graph for this function:



5.38.3.3 std::list< std::string > * ModelElement::_saveInstance (std::string *type*) [protected], [virtual]

Reimplemented in ModelComponent.

Here is the call graph for this function:

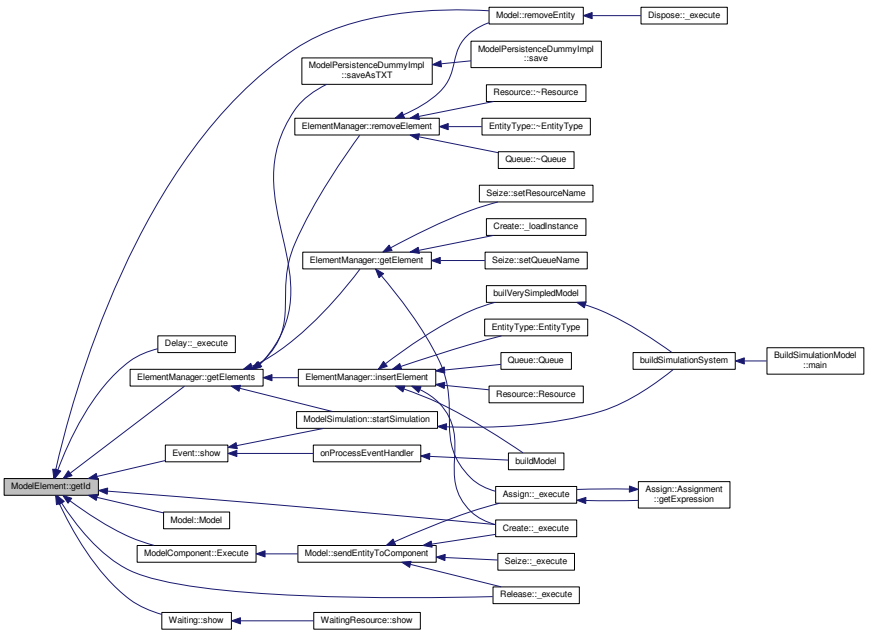
5.38.3.4 virtual bool ModelElement::_verifySymbols (std::string * *errorMessage*) [protected], [pure virtual]

Implemented in Assign, Resource, Seize, Queue, EntityType, Release, Entity, Variable, Create, Delay, StatisticsCollector, Attribute, and Dispose.

Here is the caller graph for this function:

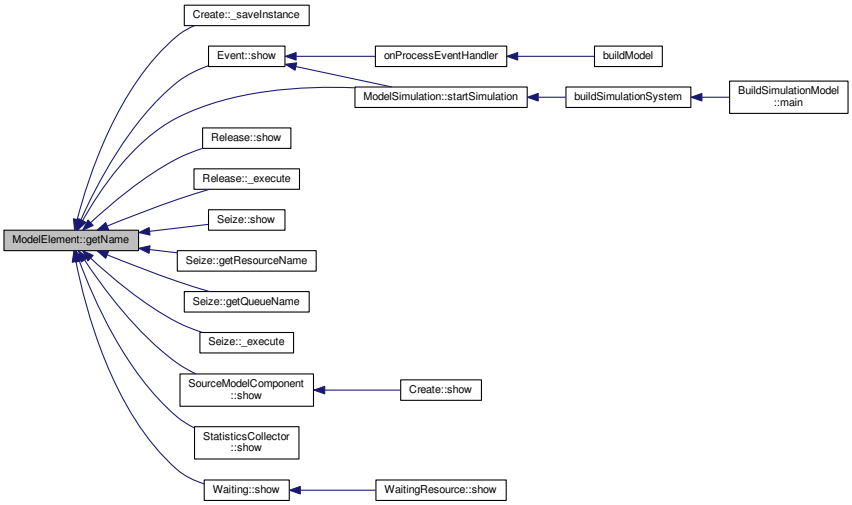
5.38.3.5 Util::identitifcation ModelElement::getId ( ) const

Here is the caller graph for this function:



5.38.3.6 std::string ModelElement::getName ( ) const

Here is the caller graph for this function:



5.38.3.7 static void ModelElement::LoadInstance (std::list< std::string > *words*) [static]

5.38.3.8 std::list< std::string > * ModelElement::SaveInstance (ModelElement * *element*) [static]

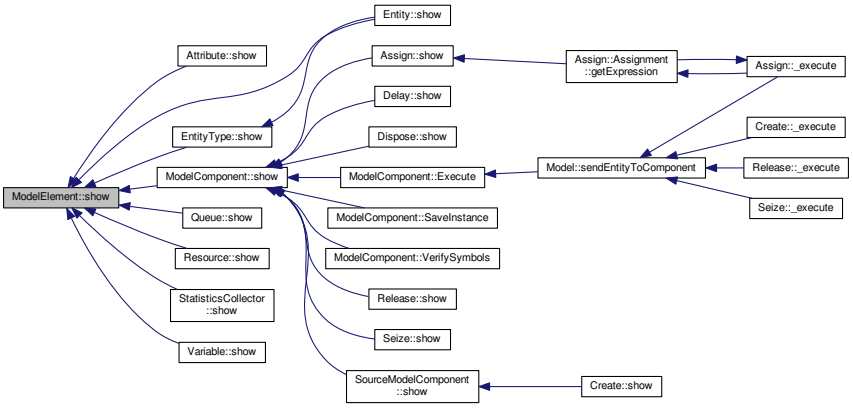
Here is the call graph for this function:

5.38.3.9 void ModelElement::setName (std::string *_name*)

5.38.3.10 std::string ModelElement::show ( ) [virtual]

Reimplemented in Assign, SourceModelComponent, ModelComponent, Resource, Queue, EntityType, Seize, Entity, Create, Delay, Attribute, Release, StatisticsCollector, Variable, and Dispose.

Here is the caller graph for this function:



5.38.3.11 static bool ModelElement::VerifySymbols (ModelElement * *element*, std::string * *errorMessage*) [static]

5.38.4 Member Data Documentation

5.38.4.1 Util::identitifcation ModelElement::_id [protected]

5.38.4.2 std::string ModelElement::_name [protected]

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

- ModelElement.h
- ModelElement.cpp

5.39 ModelInfo Class Reference

`#include <ModelInfo.h>`

Public Member Functions

- ModelInfo ()
- ModelInfo (const ModelInfo &orig)
- virtual ~ModelInfo ()
- void setName (std::string _name)
- std::string getName () const
- void setAnalystName (std::string _analystName)
- std::string getAnalystName () const
- void setDescription (std::string _description)
- std::string getDescription () const
- void setProjectTitle (std::string _projectTitle)

- std::string getProjectTitle () const
- void setVersion (std::string _version)
- std::string getVersion () const
- void setNumberOfReplications (unsigned int _numberOfReplications)
- unsigned int getNumberOfReplications () const
- void setReplicationLength (double _replicationLength)
- double getReplicationLength () const
- void setReplicationLengthTimeUnit (Util::TimeUnit _replicationLengthTimeUnit)
- Util::TimeUnit getReplicationLengthTimeUnit () const
- void setWarmUpPeriod (double _warmUpPeriod)
- double getWarmUpPeriod () const
- void setWarmUpPeriodTimeUnit (Util::TimeUnit _warmUpPeriodTimeUnit)
- Util::TimeUnit getWarmUpPeriodTimeUnit () const
- void setTerminatingCondition (std::string _terminatingCondition)
- std::string getTerminatingCondition () const

5.39.1 Detailed Description

ModelInfo stores basic model project information.

5.39.2 Constructor & Destructor Documentation

5.39.2.1 ModelInfo::ModelInfo ( )

5.39.2.2 ModelInfo::ModelInfo (const ModelInfo & *orig*)

5.39.2.3 ModelInfo::~ModelInfo ( ) [virtual]

5.39.3 Member Function Documentation

5.39.3.1 std::string ModelInfo::getAnalystName ( ) const

Here is the caller graph for this function:

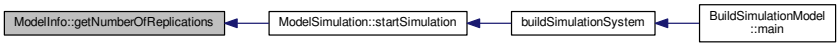
5.39.3.2 std::string ModelInfo::getDescription ( ) const

5.39.3.3 std::string ModelInfo::getName ( ) const

Here is the caller graph for this function:

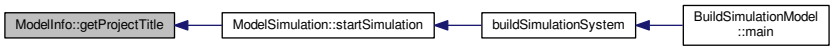
5.39.3.4 unsigned int ModelInfo::getNumberOfReplications ( ) const

Here is the caller graph for this function:



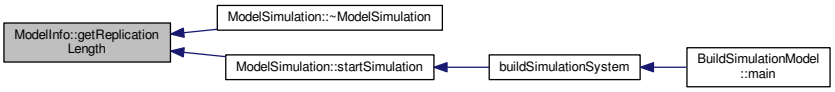
5.39.3.5 std::string ModelInfo::getProjectTitle ( ) const

Here is the caller graph for this function:



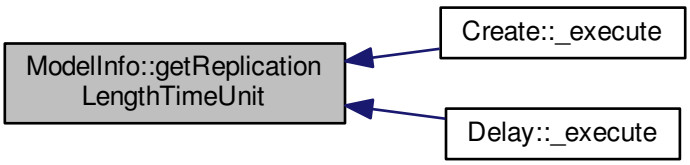
5.39.3.6 double ModelInfo::getReplicationLength ( ) const

Here is the caller graph for this function:



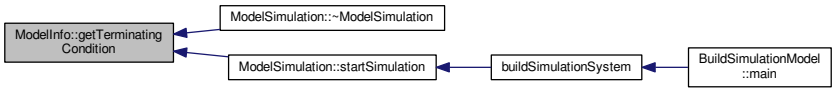
5.39.3.7 Util::TimeUnit ModelInfo::getReplicationLengthTimeUnit ( ) const

Here is the caller graph for this function:



5.39.3.8 std::string ModelInfo::getTerminatingCondition ( ) const

Here is the caller graph for this function:



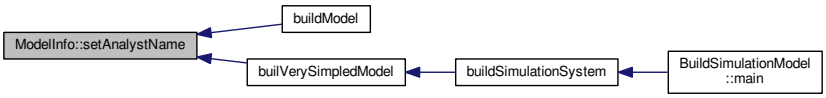
5.39.3.9 std::string ModelInfo::getVersion ( ) const

5.39.3.10 double ModelInfo::getWarmUpPeriod ( ) const

5.39.3.11 Util::TimeUnit ModelInfo::getWarmUpPeriodTimeUnit ( ) const

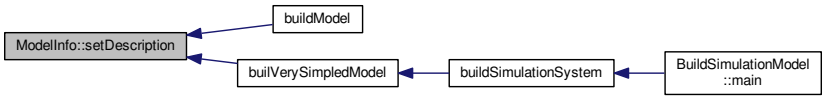
5.39.3.12 void ModelInfo::setAnalystName (std::string *_analystName*)

Here is the caller graph for this function:



5.39.3.13 void ModelInfo::setDescription (std::string *_description*)

Here is the caller graph for this function:



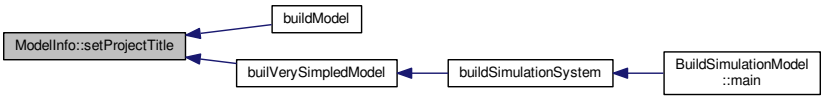
5.39.3.14 void ModelInfo::setName (std::string *_name*)

5.39.3.15 void ModelInfo::setNumberOfReplications (unsigned int *_numberOfReplications*)

Here is the caller graph for this function:

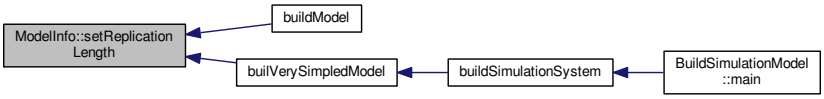
5.39.3.16 void ModelInfo::setProjectTitle (std::string *_projectTitle*)

Here is the caller graph for this function:



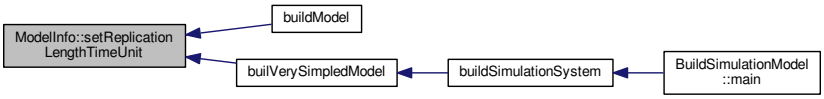
5.39.3.17 void ModelInfo::setReplicationLength (double *_replicationLength*)

Here is the caller graph for this function:



5.39.3.18 void ModelInfo::setReplicationLengthTimeUnit (Util::TimeUnit *_replicationLengthTimeUnit*)

Here is the caller graph for this function:



5.39.3.19 void ModelInfo::setTerminatingCondition (std::string *_terminatingCondition*)

5.39.3.20 void ModelInfo::setVersion (std::string *_version*)

5.39.3.21 void ModelInfo::setWarmUpPeriod (double *_warmUpPeriod*)

5.39.3.22 void ModelInfo::setWarmUpPeriodTimeUnit (Util::TimeUnit *_warmUpPeriodTimeUnit*)

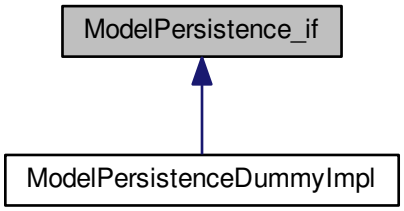
The documentation for this class was generated from the following files:

- ModelInfo.h
- ModelInfo.cpp

5.40 ModelPersistence_if Class Reference

`#include <ModelPersistence_if.h>`

Inheritance diagram for ModelPersistence_if:



Public Member Functions

- virtual bool saveAsTXT (std::string filename)=0
- virtual bool loadAsTXT (std::string filename)=0
- virtual bool saveAsXML (std::string filename)=0
- virtual bool loadAsXML (std::string filename)=0
- virtual bool save (std::string filename)=0
- virtual bool load (std::string filename)=0
- virtual bool isSaved ()=0

5.40.1 Detailed Description

First and inadequate interface for model persistence. It should use the best pattern for the DAO approach

5.40.2 Member Function Documentation

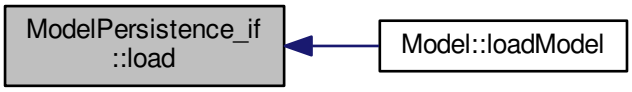
5.40.2.1 virtual bool ModelPersistence_if::isSaved ( ) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

5.40.2.2 virtual bool ModelPersistence_if::load (std::string *filename*) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

Here is the caller graph for this function:



5.40.2.3 virtual bool ModelPersistence_if::loadAsTXT (std::string *filename*) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

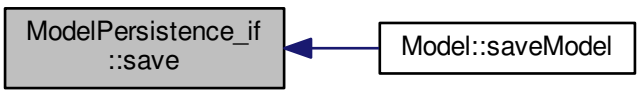
5.40.2.4 virtual bool ModelPersistence_if::loadAsXML (std::string *filename*) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

5.40.2.5 virtual bool ModelPersistence_if::save (std::string *filename*) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

Here is the caller graph for this function:



5.40.2.6 virtual bool ModelPersistence_if::saveAsTXT (std::string *filename*) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

5.40.2.7 virtual bool ModelPersistence_if::saveAsXML (std::string *filename*) [pure virtual]

Implemented in ModelPersistenceDummyImpl.

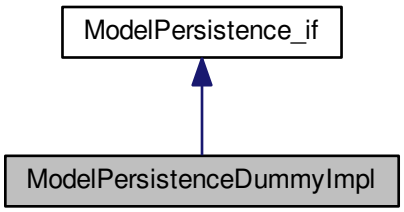
The documentation for this class was generated from the following file:

- ModelPersistence_if.h

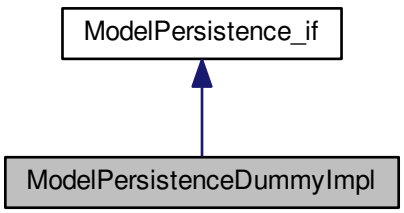
5.41 ModelPersistenceDummyImpl Class Reference

`#include <ModelPersistenceDummyImpl.h>`

Inheritance diagram for ModelPersistenceDummyImpl:



Collaboration diagram for ModelPersistenceDummyImpl:



Public Member Functions

- ModelPersistenceDummyImpl (Model ∗model)
- ModelPersistenceDummyImpl (const ModelPersistenceDummyImpl &orig)
- ∼ModelPersistenceDummyImpl ()
- virtual bool saveAsTXT (std::string filename)
- virtual bool loadAsTXT (std::string filename)
- virtual bool saveAsXML (std::string filename)
- virtual bool loadAsXML (std::string filename)
- virtual bool save (std::string filename)
- virtual bool load (std::string filename)
- virtual bool isSaved ()

5.41.1 Constructor & Destructor Documentation

5.41.1.1 ModelPersistenceDummyImpl::ModelPersistenceDummyImpl (Model ∗ *model*)

5.41.1.2 ModelPersistenceDummyImpl::ModelPersistenceDummyImpl (const ModelPersistenceDummyImpl & *orig*)

5.41.1.3 ModelPersistenceDummyImpl::∼ModelPersistenceDummyImpl ( )

5.41.2 Member Function Documentation

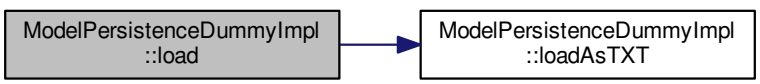
5.41.2.1 bool ModelPersistenceDummyImpl::isSaved ( ) [virtual]

Implements ModelPersistence_if.

5.41.2.2 bool ModelPersistenceDummyImpl::load (std::string *filename*) [virtual]

Implements ModelPersistence_if.

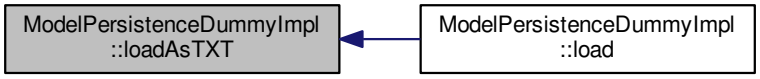
Here is the call graph for this function:



5.41.2.3 bool ModelPersistenceDummyImpl::loadAsTXT (std::string *filename*) [virtual]

Implements ModelPersistence_if.

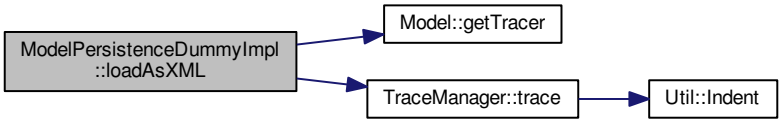
Here is the caller graph for this function:



5.41.2.4 bool ModelPersistenceDummyImpl::loadAsXML (std::string *filename*) [virtual]

Implements ModelPersistence_if.

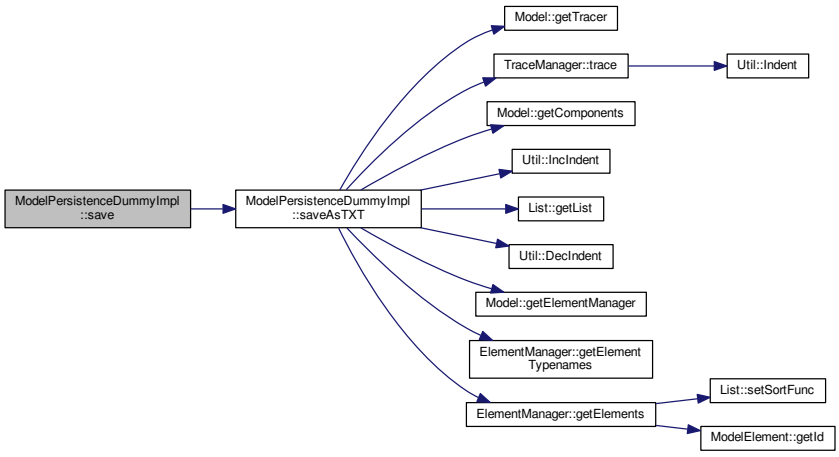
Here is the call graph for this function:



5.41.2.5 bool ModelPersistenceDummyImpl::save (std::string *filename*) [virtual]

Implements ModelPersistence_if.

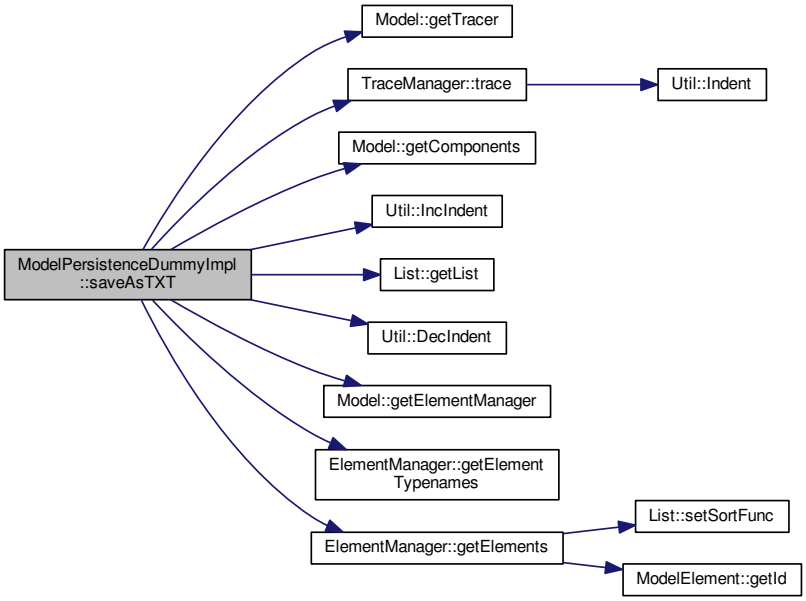
Here is the call graph for this function:



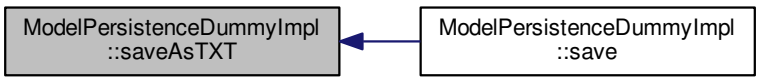
5.41.2.6 bool ModelPersistenceDummyImpl::saveAsTXT (std::string *filename*) [virtual]

Implements ModelPersistence_if.

Here is the call graph for this function:



Here is the caller graph for this function:



5.41.2.7 bool ModelPersistenceDummyImpl::saveAsXML (std::string *filename*) [virtual]

Implements ModelPersistence_if.

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

- ModelPersistenceDummyImpl.h
- ModelPersistenceDummyImpl.cpp

5.42 ModelSimulation Class Reference

`#include <ModelSimulation.h>`

Public Member Functions

- ModelSimulation (Model *model)
- ModelSimulation (const ModelSimulation &orig)
- virtual ~ModelSimulation ()
- void startSimulation ()

  Starts a sequential execution of a simulation, ie, a set of replications of this model.
- void pauseSimulation ()
- void stepSimulation ()

  Executes the processing of a single event, the next one in the future events list.
- void stopSimulation ()
- void restartSimulation ()
- void setPauseOnEvent (bool _pauseOnEvent)
- bool isPauseOnEvent () const
- void setStepByStep (bool _stepByStep)
- bool isStepByStep () const
- void setInitializeStatistics (bool _initializeStatistics)
- bool isInitializeStatistics () const
- void setInitializeSystem (bool _initializeSystem)
- bool isInitializeSystem () const
- void setPauseOnReplication (bool _pauseBetweenReplications)
- bool isPauseOnReplication () const
- double getSimulatedTime () const
- bool isRunning () const
- unsigned int getCurrentReplicationNumber () const
- ModelComponent * getCurrentComponent () const
- Entity * getCurrentEntity () const

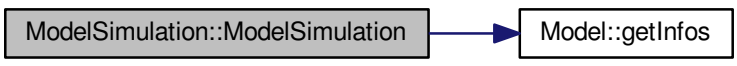
5.42.1 Detailed Description

The ModelSimulation controls the simulation of a model, alowing to start, pause, resume e stop a simulation, composed by a set of replications.

5.42.2 Constructor & Destructor Documentation

5.42.2.1 ModelSimulation::ModelSimulation (Model * *model*)

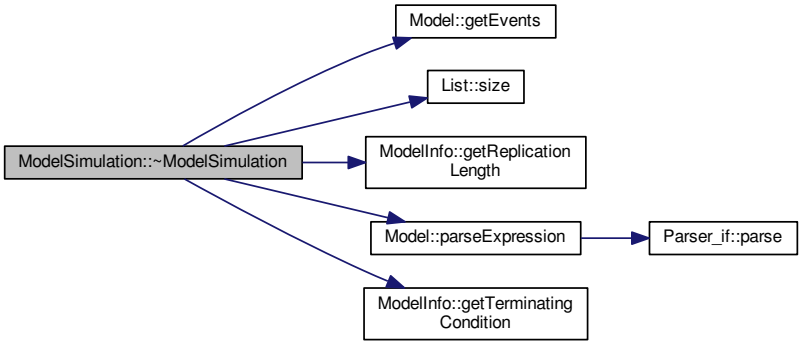
Here is the call graph for this function:



5.42.2.2 ModelSimulation::ModelSimulation (const ModelSimulation & *orig*)

5.42.2.3 ModelSimulation::~ModelSimulation ( ) [virtual]

Here is the call graph for this function:



5.42.3 Member Function Documentation

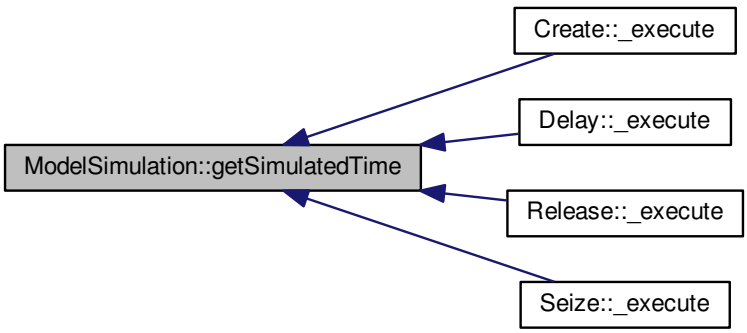
5.42.3.1 ModelComponent * ModelSimulation::getCurrentComponent ( ) const

5.42.3.2 Entity * ModelSimulation::getCurrentEntity ( ) const

5.42.3.3 unsigned int ModelSimulation::getCurrentReplicationNumber ( ) const

5.42.3.4 double ModelSimulation::getSimulatedTime ( ) const

Here is the caller graph for this function:



5.42.3.5 bool ModelSimulation::isInitializeStatistics ( ) const

5.42.3.6 bool ModelSimulation::isInitializeSystem ( ) const

5.42.3.7 bool ModelSimulation::isPauseOnEvent ( ) const

5.42.3.8 bool ModelSimulation::isPauseOnReplication ( ) const

5.42.3.9 bool ModelSimulation::isRunning ( ) const

The current time in the model being simulated, i.e., the instant when the current event was triggered

5.42.3.10 bool ModelSimulation::isStepByStep ( ) const

5.42.3.11 void ModelSimulation::pauseSimulation ( )

5.42.3.12 void ModelSimulation::restartSimulation ( )

5.42.3.13 void ModelSimulation::setInitializeStatistics (bool *_initializeStatistics*)

5.42.3.14 void ModelSimulation::setInitializeSystem (bool *_initializeSystem*)

5.42.3.15 void ModelSimulation::setPauseOnEvent (bool *_pauseOnEvent*)

5.42.3.16 void ModelSimulation::setPauseOnReplication (bool *_pauseBetweenReplications*)

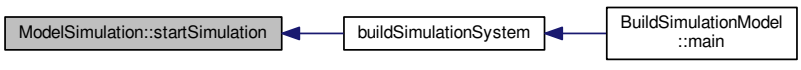
5.42.3.17 void ModelSimulation::setStepByStep (bool *_stepByStep*)

5.42.3.18 void ModelSimulation::startSimulation ( )

Starts a sequential execution of a simulation, ie, a set of replications of this model.

Checks the model and if ok then initialize the simulation, execute repeatedly each replication and then show simulation statistics

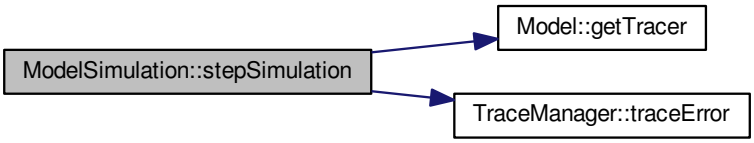
Here is the caller graph for this function:



5.42.3.19 void ModelSimulation::stepSimulation ( )

Executes the processing of a single event, the next one in the future events list.

Here is the call graph for this function:



5.42.3.20 void ModelSimulation::stopSimulation ( )

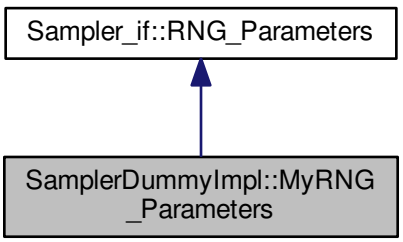
The documentation for this class was generated from the following files:

- ModelSimulation.h
- ModelSimulation.cpp

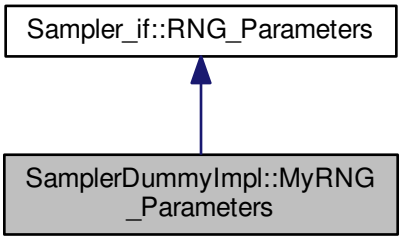
5.43 SamplerDummyImpl::MyRNG_Parameters Class Reference

`#include <SamplerDummyImpl.h>`

Inheritance diagram for SamplerDummyImpl::MyRNG_Parameters:



Collaboration diagram for SamplerDummyImpl::MyRNG_Parameters:



Public Attributes

- unsigned int seed
- unsigned int module
- unsigned int multiplier

5.43.1 Member Data Documentation

5.43.1.1 unsigned int SamplerDummyImpl::MyRNG_Parameters::module

5.43.1.2 unsigned int SamplerDummyImpl::MyRNG_Parameters::multiplier

5.43.1.3 unsigned int SamplerDummyImpl::MyRNG_Parameters::seed

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

- SamplerDummyImpl.h

5.44 OnEventManager Class Reference

`#include <OnEventManager.h>`

Public Member Functions

- OnEventManager ()
- OnEventManager (const OnEventManager &orig)
- virtual ~OnEventManager ()
- void addOnReplicationStartHandler (simulationEventHandler EventHandler)
- void addOnReplicationStepHandler (simulationEventHandler EventHandler)
- void addOnReplicationEndHandler (simulationEventHandler EventHandler)
- void addOnProcessEventHandler (simulationEventHandler EventHandler)
- void addOnSimulationStartHandler (simulationEventHandler EventHandler)
- void addOnSimulationEndHandler (simulationEventHandler EventHandler)
- void NotifyReplicationStartHandlers (SimulationEvent *se)
- void NotifyReplicationStepHandlers (SimulationEvent *se)
- void NotifyReplicationEndHandlers (SimulationEvent *se)
- void NotifyProcessEventHandlers (SimulationEvent *se)
- void NotifySimulationStartHandlers (SimulationEvent *se)
- void NotifySimulationEndHandlers (SimulationEvent *se)

5.44.1 Detailed Description

OnEventManager allows external methods to hook interval simulation events as listeners (or observers) of pecific events. All methods added as listeners of an event will be invovked when that event is triggered.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 OnEventManager::OnEventManager ( )

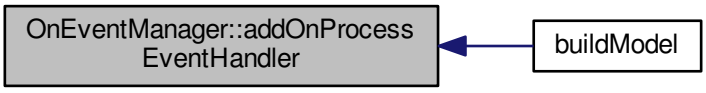
5.44.2.2 OnEventManager::OnEventManager (const OnEventManager & *orig*)

5.44.2.3 OnEventManager::~OnEventManager ( ) [virtual]

5.44.3 Member Function Documentation

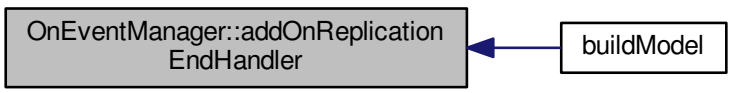
5.44.3.1 void OnEventManager::addOnProcessEventHandler (simulationEventHandler *EventHandler*)

Here is the caller graph for this function:



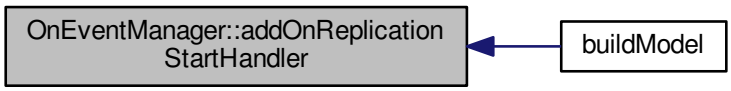
5.44.3.2 void OnEventManager::addOnReplicationEndHandler (simulationEventHandler *EventHandler*)

Here is the caller graph for this function:



5.44.3.3 void OnEventManager::addOnReplicationStartHandler (simulationEventHandler *EventHandler*)

Here is the caller graph for this function:

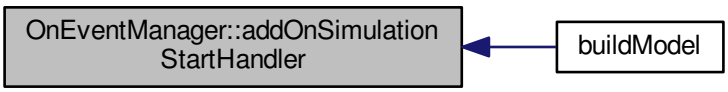


5.44.3.4 void OnEventManager::addOnReplicationStepHandler (simulationEventHandler *EventHandler*)

5.44.3.5 void OnEventManager::addOnSimulationEndHandler (simulationEventHandler *EventHandler*)

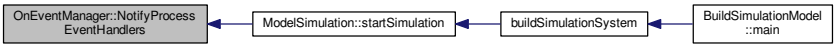
5.44.3.6 void OnEventManager::addOnSimulationStartHandler (simulationEventHandler *EventHandler*)

Here is the caller graph for this function:



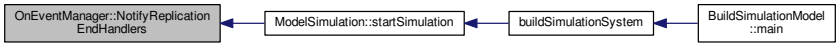
5.44.3.7 void OnEventManager::NotifyProcessEventHandlers (SimulationEvent * *se*)

Here is the caller graph for this function:



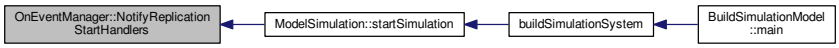
5.44.3.8 void OnEventManager::NotifyReplicationEndHandlers (SimulationEvent * se)

Here is the caller graph for this function:



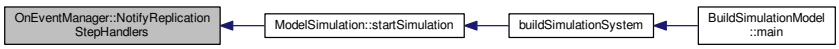
5.44.3.9 void OnEventManager::NotifyReplicationStartHandlers (SimulationEvent * se)

Here is the caller graph for this function:



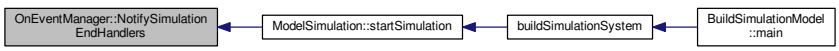
5.44.3.10 void OnEventManager::NotifyReplicationStepHandlers (SimulationEvent * se)

Here is the caller graph for this function:



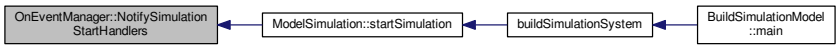
5.44.3.11 void OnEventManager::NotifySimulationEndHandlers (SimulationEvent * se)

Here is the caller graph for this function:



5.44.3.12 void OnEventManager::NotifySimulationStartHandlers (SimulationEvent * *se*)

Here is the caller graph for this function:



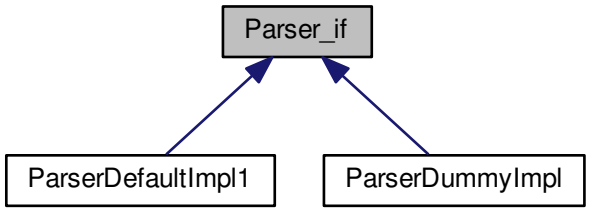
The documentation for this class was generated from the following files:

- OnEventManager.h
- OnEventManager.cpp

5.45 Parser_if Class Reference

`#include <Parser_if.h>`

Inheritance diagram for Parser_if:



Public Member Functions

- virtual double parse (const std::string expression)=0
- virtual double parse (const std::string expression, bool *success, std::string *errorMessage)=0
- virtual std::string * getErrorMessage ()=0

5.45.1 Member Function Documentation

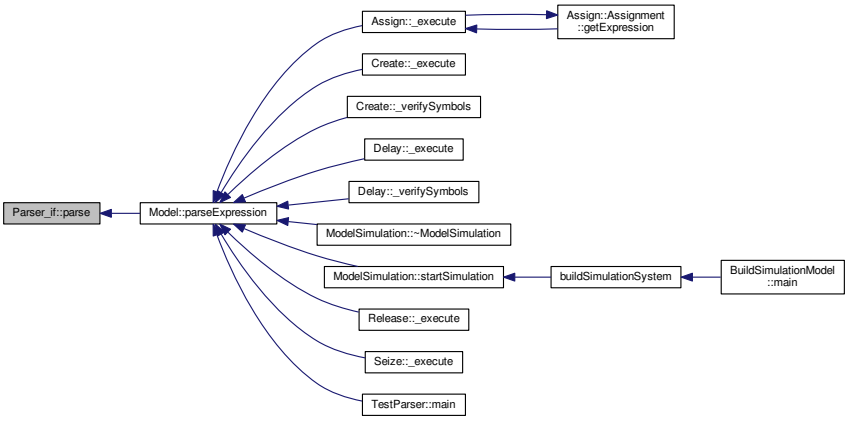
5.45.1.1 virtual std::string* Parser_if::getErrorMessage ( ) [pure virtual]

Implemented in ParserDummyImpl, and ParserDefaultImpl1.

5.45.1.2 virtual double Parser_if::parse (const std::string *expression*) [pure virtual]

Implemented in ParserDummyImpl, and ParserDefaultImpl1.

Here is the caller graph for this function:



5.45.1.3 virtual double Parser_if::parse (const std::string *expression*, bool * *success*, std::string * *errorMessage*) [pure virtual]

Implemented in ParserDummyImpl, and ParserDefaultImpl1.

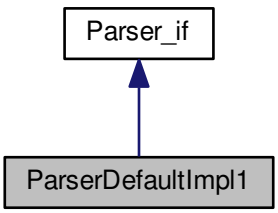
The documentation for this class was generated from the following file:

- Parser_if.h

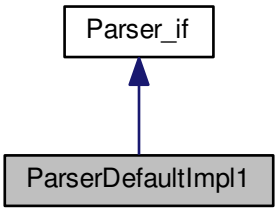
5.46 ParserDefaultImpl1 Class Reference

`#include <ParserDefaultImpl1.h>`

Inheritance diagram for ParserDefaultImpl1:



Collaboration diagram for ParserDefaultImpl1:



Public Member Functions

- ParserDefaultImpl1 (Model ∗model)
- ParserDefaultImpl1 (const ParserDefaultImpl1 &orig)
- virtual ∼ParserDefaultImpl1 ()
- double parse (const std::string expression)
- double parse (const std::string expression, bool ∗success, std::string ∗errorMessage)
- std::string ∗ getErrorMessage ()

5.46.1 Constructor & Destructor Documentation

5.46.1.1 ParserDefaultImpl1::ParserDefaultImpl1 (Model ∗ *model*)

5.46.1.2 ParserDefaultImpl1::ParserDefaultImpl1 (const ParserDefaultImpl1 & *orig*)

5.46.1.3 ParserDefaultImpl1::∼ParserDefaultImpl1 ( ) [virtual]

5.46.2 Member Function Documentation

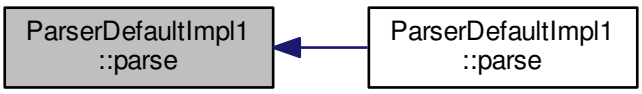
5.46.2.1 std::string ∗ ParserDefaultImpl1::getErrorMessage ( ) [virtual]

Implements Parser_if.

5.46.2.2 double ParserDefaultImpl1::parse (const std::string *expression*) [virtual]

Implements Parser_if.

Here is the caller graph for this function:



5.46.2.3 double ParserDefaultImpl1::parse (const std::string *expression*, bool * *success*, std::string * *errorMessage*) [virtual]

Implements Parser_if.

Here is the call graph for this function:

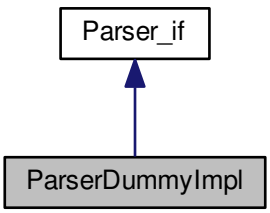
The documentation for this class was generated from the following files:

- ParserDefaultImpl1.h
- ParserDefaultImpl1.cpp

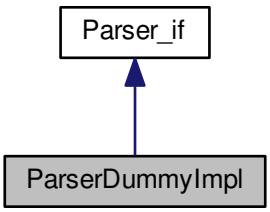
5.47 ParserDummyImpl Class Reference

`#include <ParserDummyImpl.h>`

Inheritance diagram for ParserDummyImpl:



Collaboration diagram for ParserDummyImpl:



Public Member Functions

- ParserDummyImpl (Model *model)
- ParserDummyImpl (const ParserDummyImpl &orig)
- virtual ~ParserDummyImpl ()
- double parse (const std::string expression)
- double parse (const std::string expression, bool *success, std::string *errorMessage)
- std::string * getErrorMessage ()

5.47.1 Constructor & Destructor Documentation

5.47.1.1 ParserDummyImpl::ParserDummyImpl (Model * *model*)

5.47.1.2 ParserDummyImpl::ParserDummyImpl (const ParserDummyImpl & *orig*)

5.47.1.3 ParserDummyImpl::~ParserDummyImpl ( ) [virtual]

5.47.2 Member Function Documentation

5.47.2.1 std::string * ParserDummyImpl::getErrorMessage ( ) [virtual]

Implements Parser_if.

5.47.2.2 double ParserDummyImpl::parse (const std::string *expression*) [virtual]

Implements Parser_if.

Here is the caller graph for this function:

5.47.2.3 double ParserDummyImpl::parse (const std::string *expression*, bool * *success*, std::string * *errorMessage*) [virtual]

Implements Parser_if.

Here is the call graph for this function:

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

- ParserDummyImpl.h
- ParserDummyImpl.cpp

5.48 Plugin Class Reference

`#include <Plugin.h>`

Public Member Functions

- Plugin (std::string name, bool source, bool drain)
- Plugin (const Plugin &orig)
- virtual ~Plugin ()
- bool isDrain () const
- bool isSource () const

5.48.1 Detailed Description

A Plugin represents a dynamically linked component class (ModelComponent) or element class (ModelElement); It gives access to a ModelComponent so it can be used by the model. Classes like Create, Delay, and Dispose are examples of PlugIns. It corresponds directly to the "Expansible" part (the capitalized 'E') of the GenESyS acronymous PlugIns are NOT implemented yet

5.48.2 Constructor & Destructor Documentation

5.48.2.1 Plugin::Plugin (std::string *name,* bool *source,* bool *drain*)

5.48.2.2 Plugin::Plugin (const Plugin & *orig*)

5.48.2.3 Plugin::~Plugin ( ) [virtual]

5.48.3 Member Function Documentation

5.48.3.1 bool Plugin::isDrain ( ) const

5.48.3.2 bool Plugin::isSource ( ) const

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

- Plugin.h
- Plugin.cpp

5.49 ProbDistrib Class Reference

`#include <ProbDistrib.h>`

Static Public Member Functions

- static double uniform (double x, double min, double max)
- static double exponential (double x, double mean)
- static double erlang (double x, double mean, double M)
- static double normal (double x, double mean, double stddev)
- static double gamma (double x, double mean, double alpha)
- static double beta (double x, double alpha, double beta)
- static double weibull (double x, double alpha, double scale)
- static double logNormal (double x, double mean, double stddev)
- static double triangular (double x, double min, double mode, double max)

5.49.1 Member Function Documentation

5.49.1.1 double ProbDistrib::beta (double *x*, double *alpha*, double *beta*) [static]

5.49.1.2 double ProbDistrib::erlang (double *x*, double *mean*, double *M*) [static]

5.49.1.3 double ProbDistrib::exponential (double *x*, double *mean*) [static]

5.49.1.4 double ProbDistrib::gamma (double *x*, double *mean*, double *alpha*) [static]

5.49.1.5 double ProbDistrib::logNormal (double *x*, double *mean*, double *stddev*) [static]

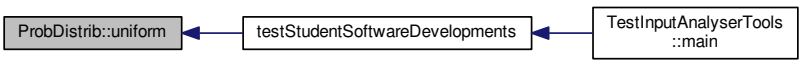
5.49.1.6 double ProbDistrib::normal (double *x*, double *mean*, double *stddev*) [static]

Here is the caller graph for this function:

5.49.1.7 double ProbDistrib::triangular (double *x*, double *min*, double *mode*, double *max*) [static]

5.49.1.8 double ProbDistrib::uniform (double *x*, double *min*, double *max*) [static]

Here is the caller graph for this function:



5.49.1.9 double ProbDistrib::weibull (double *x*, double *alpha*, double *scale*) [static]

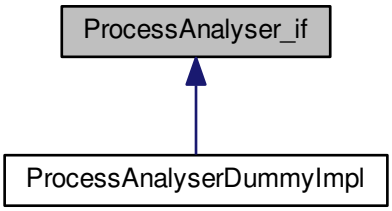
The documentation for this class was generated from the following files:

- ProbDistrib.h
- ProbDistrib.cpp

5.50 ProcessAnalyser_if Class Reference

`#include <ProcessAnalyser_if.h>`

Inheritance diagram for ProcessAnalyser_if:



Public Member Functions

- virtual List< SimulationScenario * > * getScenarios () const =0
- virtual List< SimulationControl * > * getControls () const =0
- virtual List< SimulationResponse * > * getResponses () const =0
- virtual List< SimulationControl * > * extractControlsFromModel (std::string modelFilename) const =0
- virtual List< SimulationResponse * > * extractResponsesFromModel (std::string modelFilename) const =0
- virtual void startSimulationOfScenario (SimulationScenario *scenario)=0
- virtual void startSimulation ()=0
- virtual void stopSimulation ()=0
- virtual void addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)=0

5.50.1 Detailed Description

The process analyser allows to extract controls and responses from a model, incluse some of then as controls and responses for a set of scenarios to be simulated

5.50.2 Member Function Documentation

5.50.2.1 virtual void ProcessAnalyser_if::addTraceSimulationHandler (traceSimulationProcessListener *traceSimulationProcessListener*) [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.2 virtual List<SimulationControl*>* ProcessAnalyser_if::extractControlsFromModel (std::string *modelFilename*) const [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.3 virtual List<SimulationResponse*>* ProcessAnalyser_if::extractResponsesFromModel (std::string *modelFilename*) const [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.4 virtual List<SimulationControl*>* ProcessAnalyser_if::getControls ( ) const [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.5 virtual List<SimulationResponse*>* ProcessAnalyser_if::getResponses ( ) const [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.6 virtual List<SimulationScenario*>* ProcessAnalyser_if::getScenarios ( ) const [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.7 virtual void ProcessAnalyser_if::startSimulation ( ) [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.8 virtual void ProcessAnalyser_if::startSimulationOfScenario (SimulationScenario * *scenario*) [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

5.50.2.9 virtual void ProcessAnalyser_if::stopSimulation ( ) [pure virtual]

Implemented in ProcessAnalyserDummyImpl.

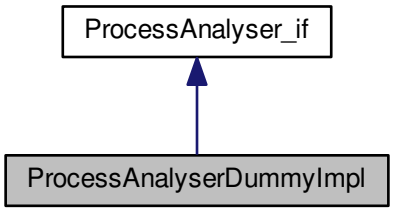
The documentation for this class was generated from the following file:

- ProcessAnalyser_if.h

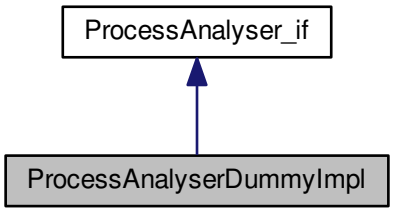
5.51 ProcessAnalyserDummyImpl Class Reference

`#include <ProcessAnalyserDummyImpl.h>`

Inheritance diagram for ProcessAnalyserDummyImpl:



Collaboration diagram for ProcessAnalyserDummyImpl:



Public Member Functions

- ProcessAnalyserDummyImpl ()
- ProcessAnalyserDummyImpl (const ProcessAnalyserDummyImpl &orig)
- ~ProcessAnalyserDummyImpl ()
- List< SimulationScenario * > * getScenarios () const
- List< SimulationControl * > * getControls () const
- List< SimulationResponse * > * getResponses () const
- List< SimulationControl * > * extractControlsFromModel (std::string modelFilename) const
- List< SimulationResponse * > * extractResponsesFromModel (std::string modelFilename) const
- void startSimulationOfScenario (SimulationScenario *scenario)
- void startSimulation ()
- void stopSimulation ()
- void addTraceSimulationHandler (traceSimulationProcessListener traceSimulationProcessListener)

5.51.1 Constructor & Destructor Documentation

5.51.1.1 ProcessAnalyserDummyImpl::ProcessAnalyserDummyImpl ( )

5.51.1.2 ProcessAnalyserDummyImpl::ProcessAnalyserDummyImpl (const ProcessAnalyserDummyImpl & *orig*)

5.51.1.3 ProcessAnalyserDummyImpl::∼ProcessAnalyserDummyImpl ( )

5.51.2 Member Function Documentation

5.51.2.1 void ProcessAnalyserDummyImpl::addTraceSimulationHandler (traceSimulationProcessListener *traceSimulationProcessListener*) [virtual]

Implements ProcessAnalyser_if.

5.51.2.2 List< SimulationControl * > * ProcessAnalyserDummyImpl::extractControlsFromModel (std::string *modelFilename*) const [virtual]

Implements ProcessAnalyser_if.

5.51.2.3 List< SimulationResponse * > * ProcessAnalyserDummyImpl::extractResponsesFromModel (std::string *modelFilename*) const [virtual]

Implements ProcessAnalyser_if.

5.51.2.4 List< SimulationControl * > * ProcessAnalyserDummyImpl::getControls ( ) const [virtual]

Implements ProcessAnalyser_if.

5.51.2.5 List< SimulationResponse * > * ProcessAnalyserDummyImpl::getResponses ( ) const [virtual]

Implements ProcessAnalyser_if.

5.51.2.6 List< SimulationScenario * > * ProcessAnalyserDummyImpl::getScenarios ( ) const [virtual]

Implements ProcessAnalyser_if.

5.51.2.7 void ProcessAnalyserDummyImpl::startSimulation ( ) [virtual]

Implements ProcessAnalyser_if.

5.51.2.8 void ProcessAnalyserDummyImpl::startSimulationOfScenario (SimulationScenario * *scenario*) [virtual]

Implements ProcessAnalyser_if.

5.51.2.9 void ProcessAnalyserDummyImpl::stopSimulation ( ) [virtual]

Implements ProcessAnalyser_if.

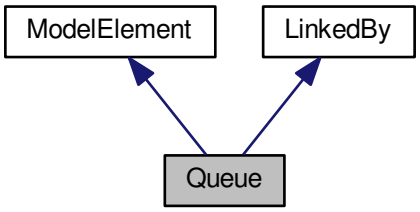
The documentation for this class was generated from the following files:

- ProcessAnalyserDummyImpl.h
- ProcessAnalyserDummyImpl.cpp

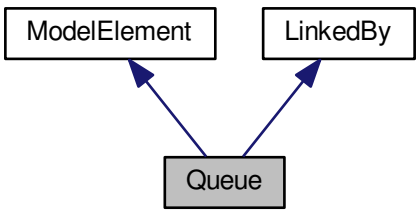
5.52 Queue Class Reference

`#include <Queue.h>`

Inheritance diagram for Queue:



Collaboration diagram for Queue:



Public Types

- enum OrderRule : int { OrderRule::FIFO = 1, OrderRule::LIFO = 2, OrderRule::HIGHESTVALUE = 3, OrderRule::SMALLESTVALUE = 4 }

Public Member Functions

- Queue (ElementManager *elems)
- Queue (ElementManager *elems, std::string name)
- Queue (const Queue &orig)
- virtual ~Queue ()
- virtual std::string show ()
- void insertElement (Waiting *element)
- void removeElement (Waiting *element, double tnow)
- unsigned int size ()
- Waiting * first ()
- void setAttributeName (std::string _attributeName)
- std::string getAttributeName () const
- void setOrderRule (OrderRule _orderRule)
- Queue::OrderRule getOrderRule () const

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.52.1 Member Enumeration Documentation

5.52.1.1 enum Queue::OrderRule : int [strong]

Enumerator

FIFO

LIFO

HIGHESTVALUE

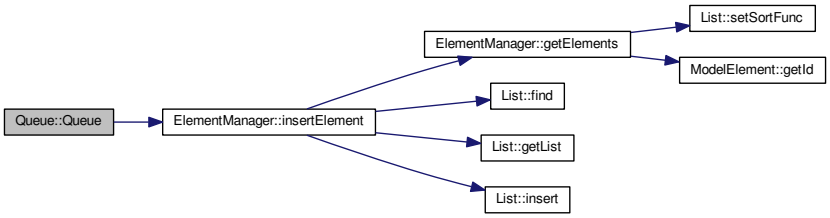
SMALLESTVALUE

5.52.2 Constructor & Destructor Documentation

5.52.2.1 Queue::Queue (ElementManager * *elems*)

5.52.2.2 Queue::Queue (ElementManager * *elems*, std::string *name*)

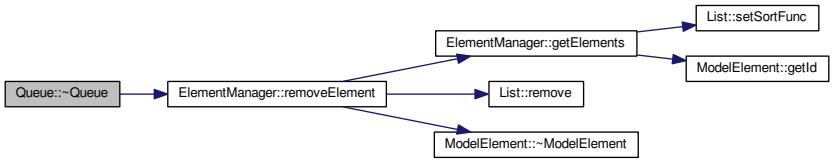
Here is the call graph for this function:



5.52.2.3 Queue::Queue (const Queue & *orig*)

5.52.2.4 Queue::~Queue ( ) [virtual]

Here is the call graph for this function:



5.52.3 Member Function Documentation

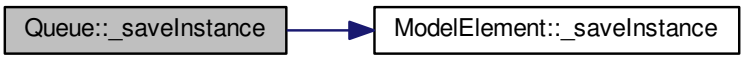
5.52.3.1 void Queue::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.52.3.2 std::list< std::string > * Queue::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



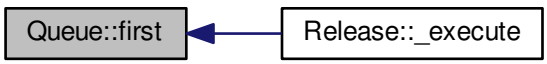
5.52.3.3 bool Queue::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.52.3.4 Waiting * Queue::first ( )

Here is the call graph for this function:

Here is the caller graph for this function:

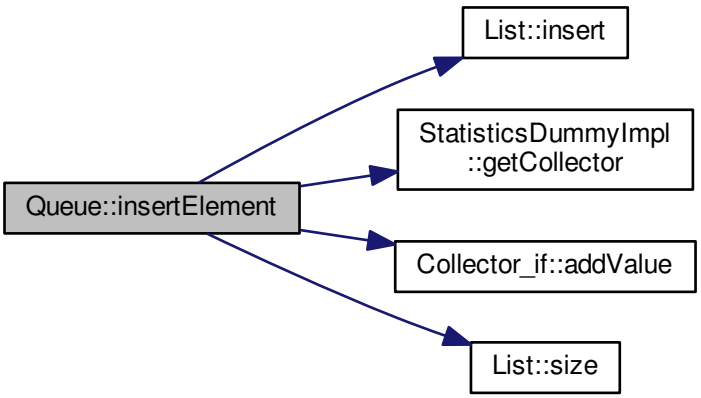


5.52.3.5 std::string Queue::getAttributeName ( ) const

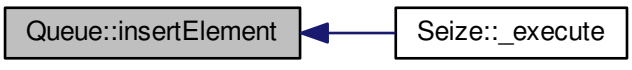
5.52.3.6 Queue::OrderRule Queue::getOrderRule ( ) const

5.52.3.7 void Queue::insertElement (Waiting * *element*)

Here is the call graph for this function:

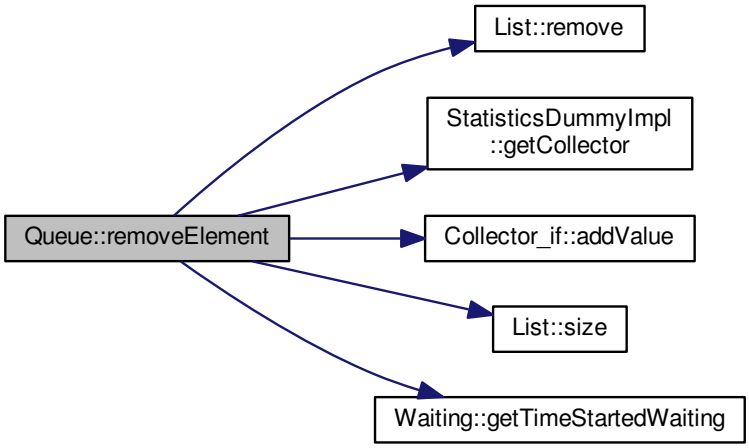


Here is the caller graph for this function:

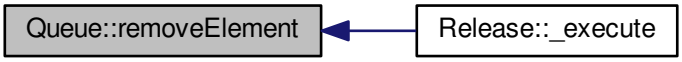


5.52.3.8 void Queue::removeElement (Waiting * *element,* double *tnow*)

Here is the call graph for this function:



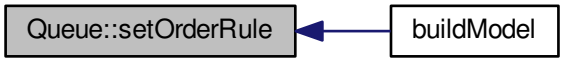
Here is the caller graph for this function:



5.52.3.9 void Queue::setAttributeName (std::string *_attributeName*)

5.52.3.10 void Queue::setOrderRule (OrderRule *_orderRule*)

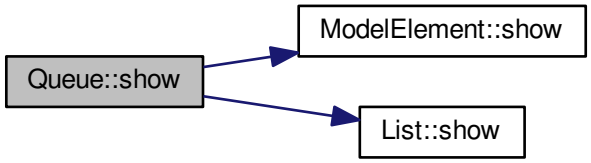
Here is the caller graph for this function:



5.52.3.11 std::string Queue::show ( ) [virtual]

Reimplemented from ModelElement.

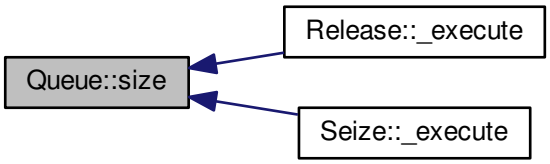
Here is the call graph for this function:



5.52.3.12 unsigned int Queue::size ( )

Here is the call graph for this function:

Here is the caller graph for this function:



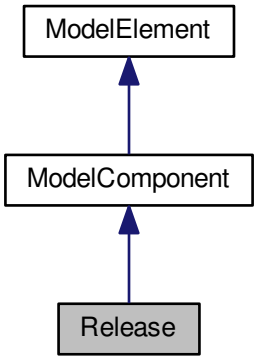
The documentation for this class was generated from the following files:

- Queue.h
- Queue.cpp

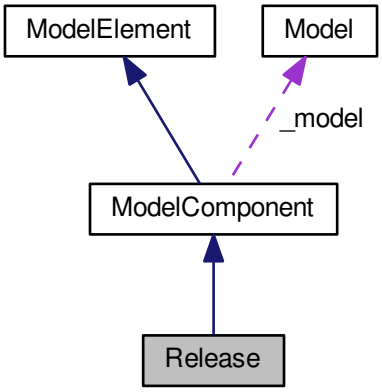
5.53 Release Class Reference

`#include <Release.h>`

Inheritance diagram for Release:



Collaboration diagram for Release:



Public Member Functions

- Release (Model *model)
- Release (const Release &orig)
- virtual ~Release ()
- virtual std::string show ()

- void setPriority (unsigned short _priority)
- unsigned short getPriority () const
- void setResourceType (Resource::ResourceType _resourceType)
- Resource::ResourceType getResourceType () const
- void setQuantity (std::string _quantity)
- std::string getQuantity () const
- void setRule (Resource::ResourceRule _rule)
- Resource::ResourceRule getRule () const
- void setSaveAttribute (std::string _saveAttribute)
- std::string getSaveAttribute () const
- void setResource (Resource *_resource)
- Resource * getResource () const

Protected Member Functions

- virtual void _execute (Entity *entity)
- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.53.1 Constructor & Destructor Documentation

5.53.1.1 Release::Release (Model * *model*)

5.53.1.2 Release::Release (const Release & *orig*)

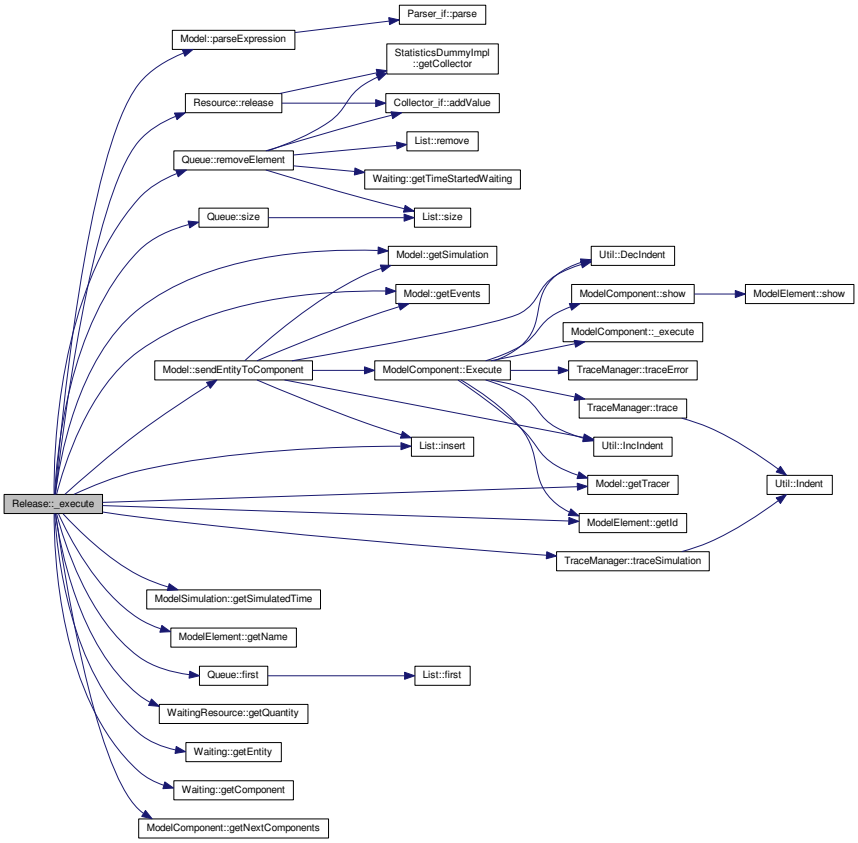
5.53.1.3 Release::~Release ( ) [virtual]

5.53.2 Member Function Documentation

5.53.2.1 void Release::_execute (Entity * *entity*) [protected], [virtual]

Implements ModelComponent.

Here is the call graph for this function:



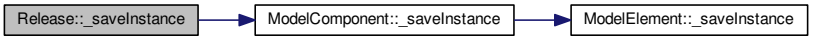
5.53.2.2 void Release::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.53.2.3 std::list< std::string > * Release::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:



5.53.2.4 bool Release::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.53.2.5 unsigned short Release::getPriority ( ) const

5.53.2.6 std::string Release::getQuantity ( ) const

5.53.2.7 Resource * Release::getResource ( ) const

5.53.2.8 Resource::ResourceType Release::getResourceType ( ) const

5.53.2.9 Resource::ResourceRule Release::getRule ( ) const

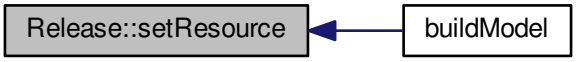
5.53.2.10 std::string Release::getSaveAttribute ( ) const

5.53.2.11 void Release::setPriority (unsigned short *_priority*)

5.53.2.12 void Release::setQuantity (std::string *_quantity*)

5.53.2.13 void Release::setResource (Resource * *_resource*)

Here is the caller graph for this function:



5.53.2.14 void Release::setResourceType (Resource::ResourceType *_resourceType*)

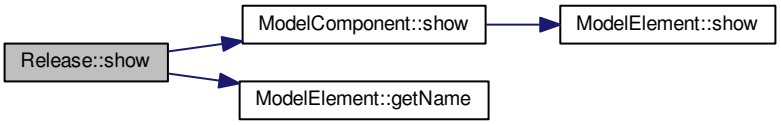
5.53.2.15 void Release::setRule (Resource::ResourceRule *_rule*)

5.53.2.16 void Release::setSaveAttribute (std::string *_saveAttribute*)

5.53.2.17 std::string Release::show ( ) [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:



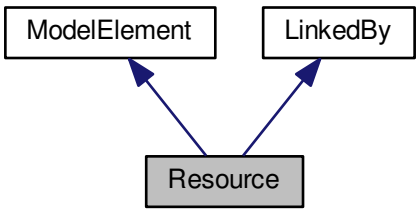
The documentation for this class was generated from the following files:

- Release.h
- Release.cpp

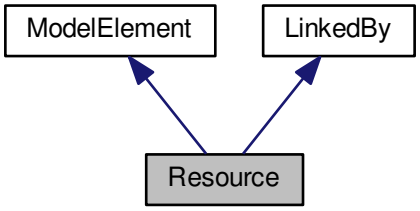
5.54 Resource Class Reference

`#include <Resource.h>`

Inheritance diagram for Resource:



Collaboration diagram for Resource:



Public Types

- enum ResourceType : int { ResourceType::SET = 1, ResourceType::RESOURCE = 2 }
- enum ResourceRule : int {
  ResourceRule::RANDOM = 1, ResourceRule::CICLICAL = 2, ResourceRule::ESPECIFIC = 3, Resource↩
  Rule::SMALLESTBUSY = 4,
  ResourceRule::LARGESTREMAININGCAPACITY = 5 }
- enum ResourceState : int {
  ResourceState::IDLE = 1, ResourceState::BUSY = 2, ResourceState::FAILED = 3, ResourceState::INACT↩
  IVE = 4,
  ResourceState::OTHER = 5 }

Public Member Functions

- Resource (ElementManager *elems)
- Resource (ElementManager *elems, std::string name)
- Resource (const Resource &orig)
- virtual ~Resource ()
- virtual std::string show ()
- void seize (unsigned int quantity, double tnow)
- void release (unsigned int quantity, double tnow)
- void setResourceState (ResourceState _resourceState)
- Resource::ResourceState getResourceState () const
- void setCapacity (unsigned int _capacity)
- unsigned int getCapacity () const
- void setCostBusyHour (double _costBusyHour)
- double getCostBusyHour () const
- void setCostIdleHour (double _costIdleHour)
- double getCostIdleHour () const
- void setCostPerUse (double _costPerUse)
- double getCostPerUse () const
- unsigned int getNumberBusy () const
- unsigned int getNumberOut () const

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.54.1 Member Enumeration Documentation

5.54.1.1 enum Resource::ResourceRule : int [strong]

Enumerator

RANDOM

CICLICAL

ESPECIFIC

SMALLESTBUSY

LARGESTREMAININGCAPACITY

5.54.1.2 enum Resource::ResourceState : int [strong]

Enumerator

IDLE

BUSY

FAILED

INACTIVE

OTHER

5.54.1.3 enum Resource::ResourceType : int [strong]

Enumerator

SET

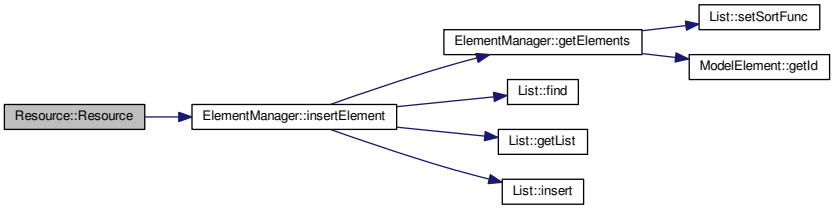
RESOURCE

5.54.2 Constructor & Destructor Documentation

5.54.2.1 Resource::Resource (ElementManager * *elems*)

5.54.2.2 Resource::Resource (ElementManager * *elems*, std::string *name*)

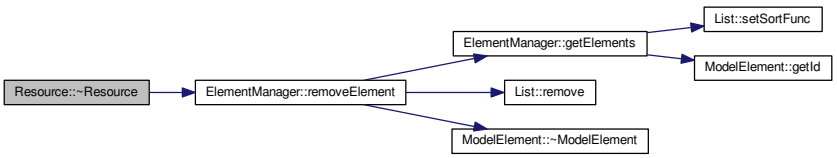
Here is the call graph for this function:



5.54.2.3 Resource::Resource (const Resource & *orig*)

5.54.2.4 Resource::~Resource ( ) [virtual]

Here is the call graph for this function:



5.54.3 Member Function Documentation

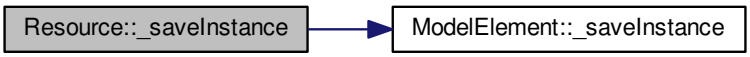
5.54.3.1 void Resource::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.54.3.2 std::list< std::string > * Resource::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:

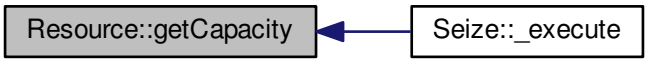


5.54.3.3 bool Resource::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.54.3.4 unsigned int Resource::getCapacity ( ) const

Here is the caller graph for this function:



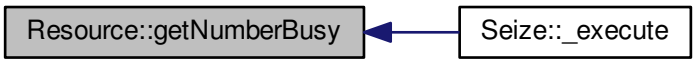
5.54.3.5 double Resource::getCostBusyHour ( ) const

5.54.3.6 double Resource::getCostIdleHour ( ) const

5.54.3.7 double Resource::getCostPerUse ( ) const

5.54.3.8 unsigned int Resource::getNumberBusy ( ) const

Here is the caller graph for this function:

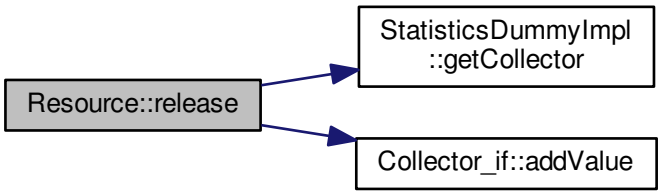


5.54.3.9 unsigned int Resource::getNumberOut ( ) const

5.54.3.10 Resource::ResourceState Resource::getResourceState ( ) const

5.54.3.11 void Resource::release (unsigned int *quantity,* double *tnow*)

Here is the call graph for this function:



Here is the caller graph for this function:

5.54.3.12 void Resource::seize (unsigned int *quantity,* double *tnow*)

Here is the caller graph for this function:

5.54.3.13 void Resource::setCapacity (unsigned int *_capacity*)

Here is the caller graph for this function:

5.54.3.14 void Resource::setCostBusyHour (double *_costBusyHour*)

5.54.3.15 void Resource::setCostIdleHour (double *_costIdleHour*)

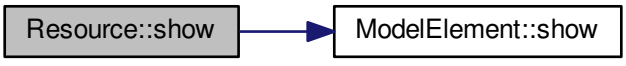
5.54.3.16 void Resource::setCostPerUse (double *_costPerUse*)

5.54.3.17 void Resource::setResourceState (ResourceState *_resourceState*)

5.54.3.18 std::string Resource::show ( ) [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



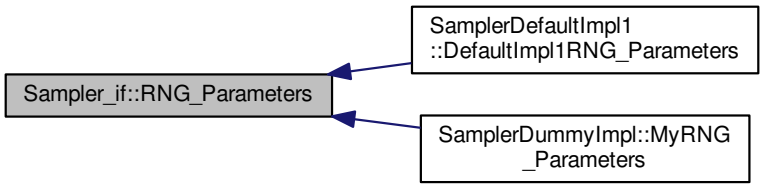
The documentation for this class was generated from the following files:

- Resource.h
- Resource.cpp

5.55 Sampler_if::RNG_Parameters Class Reference

`#include <Sampler_if.h>`

Inheritance diagram for Sampler_if::RNG_Parameters:



5.55.1 Detailed Description

class that encapsulates attributes required to generate random numbers, which depends on the generation method used.

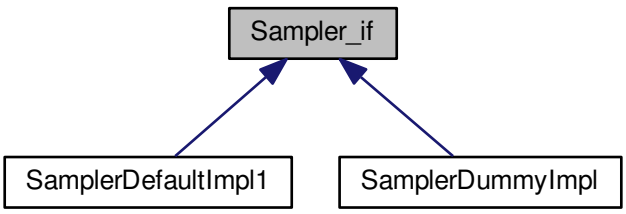
The documentation for this class was generated from the following file:

- Sampler_if.h

5.56 Sampler_if Class Reference

`#include <Sampler_if.h>`

Inheritance diagram for Sampler_if:



Classes

- class RNG_Parameters

Public Member Functions

- virtual double random ()=0
- virtual double sampleUniform (double min, double max)=0
- virtual double sampleExponential (double mean)=0
- virtual double sampleErlang (double mean, int M)=0
- virtual double sampleNormal (double mean, double stddev)=0
- virtual double sampleGamma (double mean, double alpha)=0
- virtual double sampleBeta (double alpha, double beta, double infLimit, double supLimit)=0
- virtual double sampleWeibull (double alpha, double scale)=0
- virtual double sampleLogNormal (double mean, double stddev)=0
- virtual double sampleTriangular (double min, double mode, double max)=0
- virtual double sampleDiscrete (double value, double acumProb,...)=0
- virtual void setRNGparameters (RNG_Parameters *param)=0
- virtual RNG_Parameters * getRNGparameters () const =0

5.56.1 Detailed Description

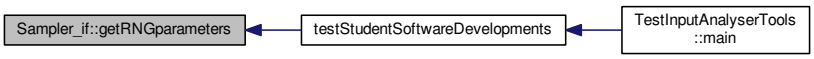
Interface that describes the methods to be implemented by classes that generate random values that follow a specific probability distribution.

5.56.2 Member Function Documentation

5.56.2.1 virtual RNG_Parameters* Sampler_if::getRNGparameters ( ) const [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

Here is the caller graph for this function:



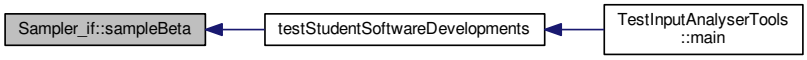
5.56.2.2 virtual double Sampler_if::random ( ) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

5.56.2.3 virtual double Sampler_if::sampleBeta (double *alpha,* double *beta,* double *infLimit,* double *supLimit*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

Here is the caller graph for this function:



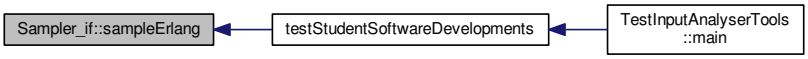
5.56.2.4 virtual double Sampler_if::sampleDiscrete (double *value,* double *acumProb,* ...) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

5.56.2.5 virtual double Sampler_if::sampleErlang (double *mean,* int *M*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

Here is the caller graph for this function:



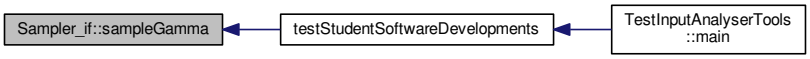
5.56.2.6 virtual double Sampler_if::sampleExponential (double *mean*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

5.56.2.7 virtual double Sampler_if::sampleGamma (double *mean,* double *alpha*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

Here is the caller graph for this function:



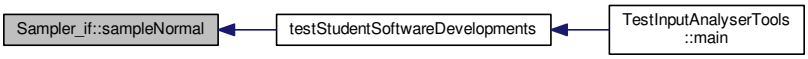
5.56.2.8 virtual double Sampler_if::sampleLogNormal (double *mean,* double *stddev*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

5.56.2.9 virtual double Sampler_if::sampleNormal (double *mean,* double *stddev*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

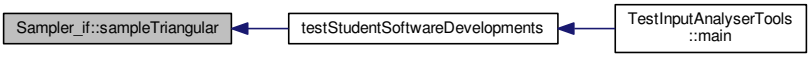
Here is the caller graph for this function:



5.56.2.10 virtual double Sampler_if::sampleTriangular (double *min,* double *mode,* double *max*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

Here is the caller graph for this function:



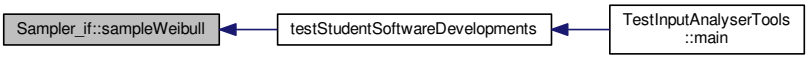
5.56.2.11 virtual double Sampler_if::sampleUniform (double *min,* double *max*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

5.56.2.12 virtual double Sampler_if::sampleWeibull (double *alpha,* double *scale*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

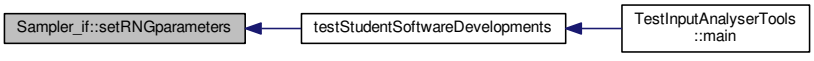
Here is the caller graph for this function:



5.56.2.13 virtual void Sampler_if::setRNGparameters (RNG_Parameters * *param*) [pure virtual]

Implemented in SamplerDefaultImpl1, and SamplerDummyImpl.

Here is the caller graph for this function:



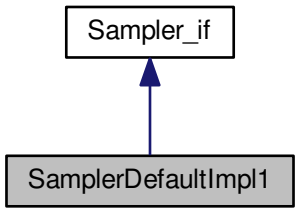
The documentation for this class was generated from the following file:

- Sampler_if.h

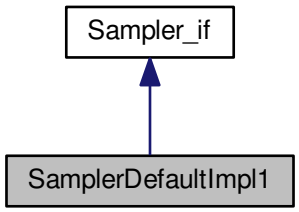
5.57 SamplerDefaultImpl1 Class Reference

`#include <SamplerDefaultImpl1.h>`

Inheritance diagram for SamplerDefaultImpl1:



Collaboration diagram for SamplerDefaultImpl1:



Classes

- class DefaultImpl1RNG_Parameters

Public Member Functions

- SamplerDefaultImpl1 ()
- SamplerDefaultImpl1 (const SamplerDefaultImpl1 &orig)
- virtual ~SamplerDefaultImpl1 ()
- double random ()
- double sampleUniform (double min, double max)
- double sampleExponential (double mean)
- double sampleErlang (double mean, int M)
- double sampleNormal (double mean, double stddev)
- double sampleGamma (double mean, double alpha)

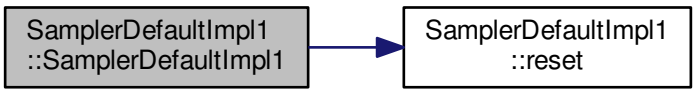
- double sampleBeta (double alpha, double beta, double infLimit, double supLimit)
- double sampleWeibull (double alpha, double scale)
- double sampleLogNormal (double mean, double stddev)
- double sampleTriangular (double min, double mode, double max)
- double sampleDiscrete (double value, double acumProb,...)
- void reset ()

  reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.
- void setRNGparameters (RNG_Parameters *param)
- RNG_Parameters * getRNGparameters () const

5.57.1 Constructor & Destructor Documentation

5.57.1.1 SamplerDefaultImpl1::SamplerDefaultImpl1 ( )

Here is the call graph for this function:



5.57.1.2 SamplerDefaultImpl1::SamplerDefaultImpl1 (const SamplerDefaultImpl1 & *orig*)

5.57.1.3 SamplerDefaultImpl1::~SamplerDefaultImpl1 ( ) [virtual]

5.57.2 Member Function Documentation

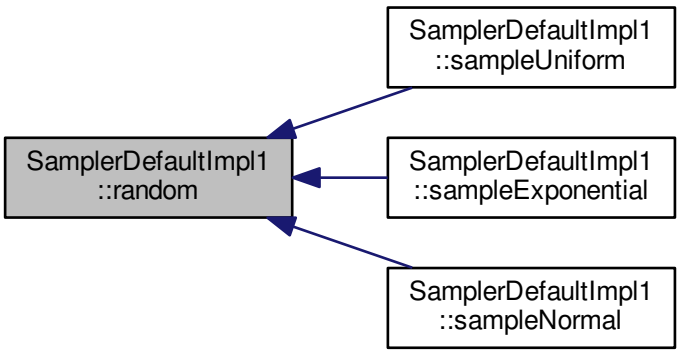
5.57.2.1 Sampler_if::RNG_Parameters * SamplerDefaultImpl1::getRNGparameters ( ) const [virtual]

Implements Sampler_if.

5.57.2.2 double SamplerDefaultImpl1::random ( ) [virtual]

Implements Sampler_if.

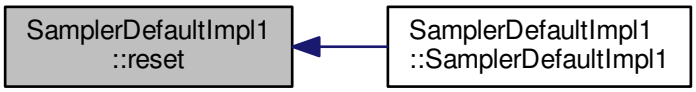
Here is the caller graph for this function:



5.57.2.3 void SamplerDefaultImpl1::reset ( )

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

Here is the caller graph for this function:



5.57.2.4 double SamplerDefaultImpl1::sampleBeta (double *alpha,* double *beta,* double *infLimit,* double *supLimit*) [virtual]

Implements Sampler_if.

5.57.2.5 double SamplerDefaultImpl1::sampleDiscrete (double *value,* double *acumProb,* ...) [virtual]

Implements Sampler_if.

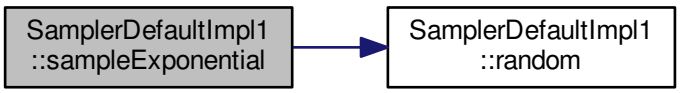
5.57.2.6 double SamplerDefaultImpl1::sampleErlang (double *mean,* int *M*) [virtual]

Implements Sampler_if.

5.57.2.7 double SamplerDefaultImpl1::sampleExponential (double *mean*) [virtual]

Implements Sampler_if.

Here is the call graph for this function:



5.57.2.8 double SamplerDefaultImpl1::sampleGamma (double *mean,* double *alpha*) [virtual]

Implements Sampler_if.

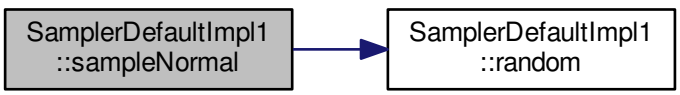
5.57.2.9 double SamplerDefaultImpl1::sampleLogNormal (double *mean,* double *stddev*) [virtual]

Implements Sampler_if.

5.57.2.10 double SamplerDefaultImpl1::sampleNormal (double *mean,* double *stddev*) [virtual]

Implements Sampler_if.

Here is the call graph for this function:



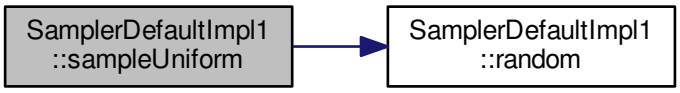
5.57.2.11 double SamplerDefaultImpl1::sampleTriangular (double *min,* double *mode,* double *max*) [virtual]

Implements Sampler_if.

5.57.2.12 double SamplerDefaultImpl1::sampleUniform (double *min*, double *max*) [virtual]

Implements Sampler_if.

Here is the call graph for this function:



5.57.2.13 double SamplerDefaultImpl1::sampleWeibull (double *alpha*, double *scale*) [virtual]

Implements Sampler_if.

5.57.2.14 void SamplerDefaultImpl1::setRNGparameters (Sampler_if::RNG_Parameters * *param*) [virtual]

Implements Sampler_if.

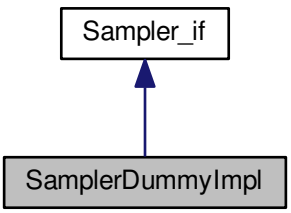
The documentation for this class was generated from the following files:

- SamplerDefaultImpl1.h
- SamplerDefaultImpl1.cpp

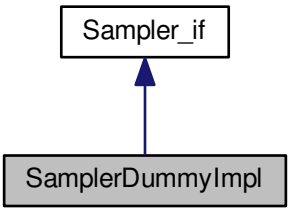
5.58 SamplerDummyImpl Class Reference

`#include <SamplerDummyImpl.h>`

Inheritance diagram for SamplerDummyImpl:



Collaboration diagram for SamplerDummyImpl:



Classes

- class MyRNG_Parameters

Public Member Functions

- SamplerDummyImpl ()
- SamplerDummyImpl (const SamplerDummyImpl &orig)
- ~SamplerDummyImpl ()
- double random ()
- double sampleUniform (double min, double max)
- double sampleExponential (double mean)
- double sampleErlang (double mean, int M)
- double sampleNormal (double mean, double stddev)
- double sampleGamma (double mean, double alpha)
- double sampleBeta (double alpha, double beta, double infLimit, double supLimit)
- double sampleWeibull (double alpha, double scale)
- double sampleLogNormal (double mean, double stddev)
- double sampleTriangular (double min, double mode, double max)
- double sampleDiscrete (double value, double acumProb,...)
- void setRNGparameters (RNG_Parameters *param)
- RNG_Parameters * getRNGparameters () const

5.58.1 Constructor & Destructor Documentation

5.58.1.1 SamplerDummyImpl::SamplerDummyImpl ( )

5.58.1.2 SamplerDummyImpl::SamplerDummyImpl (const SamplerDummyImpl & *orig*)

5.58.1.3 SamplerDummyImpl::~SamplerDummyImpl ( )

5.58.2 Member Function Documentation

5.58.2.1 Sampler_if::RNG_Parameters * SamplerDummyImpl::getRNGparameters ( ) const [virtual]

Implements Sampler_if.

5.58.2.2 double SamplerDummyImpl::random ( ) [virtual]

Implements Sampler_if.

5.58.2.3 double SamplerDummyImpl::sampleBeta (double *alpha,* double *beta,* double *infLimit,* double *supLimit*) [virtual]

Implements Sampler_if.

5.58.2.4 double SamplerDummyImpl::sampleDiscrete (double *value,* double *acumProb,* ...) [virtual]

Implements Sampler_if.

5.58.2.5 double SamplerDummyImpl::sampleErlang (double *mean,* int *M*) [virtual]

Implements Sampler_if.

5.58.2.6 double SamplerDummyImpl::sampleExponential (double *mean*) [virtual]

Implements Sampler_if.

5.58.2.7 double SamplerDummyImpl::sampleGamma (double *mean,* double *alpha*) [virtual]

Implements Sampler_if.

5.58.2.8 double SamplerDummyImpl::sampleLogNormal (double *mean,* double *stddev*) [virtual]

Implements Sampler_if.

5.58.2.9 double SamplerDummyImpl::sampleNormal (double *mean,* double *stddev*) [virtual]

Implements Sampler_if.

5.58.2.10 double SamplerDummyImpl::sampleTriangular (double *min,* double *mode,* double *max*) [virtual]

Implements Sampler_if.

5.58.2.11 double SamplerDummyImpl::sampleUniform (double *min,* double *max*) [virtual]

Implements Sampler_if.

5.58.2.12 double SamplerDummyImpl::sampleWeibull (double *alpha,* double *scale*) [virtual]

Implements Sampler_if.

5.58.2.13 void SamplerDummyImpl::setRNGparameters (Sampler_if::RNG_Parameters * *param*) [virtual]

Implements Sampler_if.

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

- SamplerDummyImpl.h
- SamplerDummyImpl.cpp

5.59 ScenarioExperiment_if Class Reference

`#include <ScenarioExperiment_if.h>`

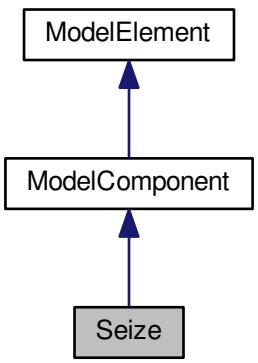
The documentation for this class was generated from the following file:

- ScenarioExperiment_if.h

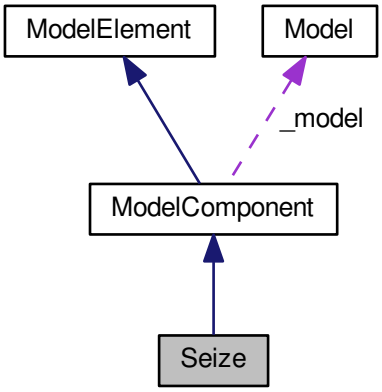
5.60 Seize Class Reference

`#include <Seize.h>`

Inheritance diagram for Seize:



Collaboration diagram for Seize:



Public Member Functions

- Seize (Model *model)
- Seize (const Seize &orig)
- virtual ~Seize ()
- virtual std::string show ()
- void setLastMemberSeized (unsigned int _lastMemberSeized)
- unsigned int getLastMemberSeized () const
- void setSaveAttribute (std::string _saveAttribute)
- std::string getSaveAttribute () const
- void setRule (Resource::ResourceRule _rule)
- Resource::ResourceRule getRule () const
- void setQuantity (std::string _quantity)
- std::string getQuantity () const
- void setResourceType (Resource::ResourceType _resourceType)
- Resource::ResourceType getResourceType () const
- void setPriority (unsigned short _priority)
- unsigned short getPriority () const
- void setAllocationType (unsigned int _allocationType)
- unsigned int getAllocationType () const
- void setResourceName (std::string _resourceName) throw ()
- std::string getResourceName () const
- void setQueueName (std::string queueName) throw ()
- std::string getQueueName () const
- void setResource (Resource *resource)
- Resource * getResource () const
- void setQueue (Queue *queue)
- Queue * getQueue () const

Protected Member Functions

- virtual void _execute (Entity *entity)
- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.60.1 Detailed Description

Seize tries to allocate a certain amount of a resource

5.60.2 Constructor & Destructor Documentation

5.60.2.1 Seize::Seize (Model * *model*)

5.60.2.2 Seize::Seize (const Seize & *orig*)

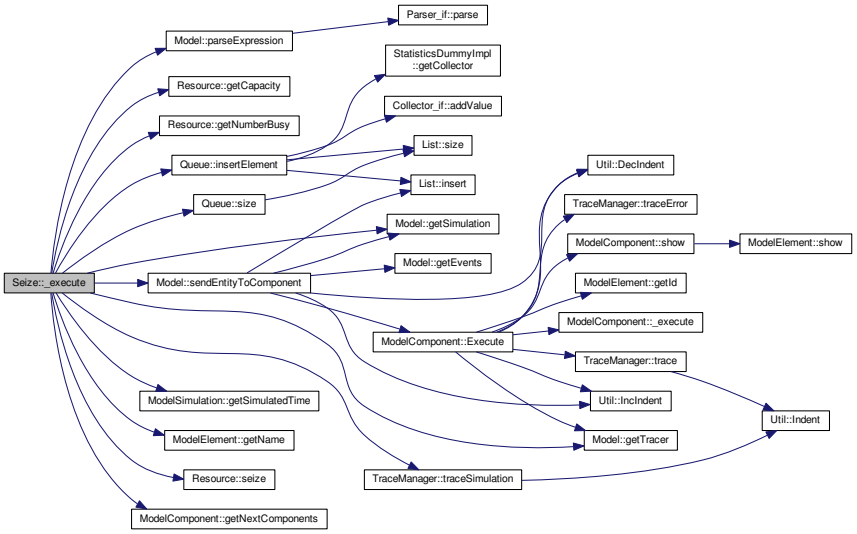
5.60.2.3 Seize::~Seize ( ) [virtual]

5.60.3 Member Function Documentation

5.60.3.1 void Seize::_execute (Entity * *entity*) [protected], [virtual]

Implements ModelComponent.

Here is the call graph for this function:



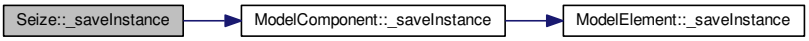
5.60.3.2 void Seize::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.60.3.3 std::list< std::string > * Seize::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:



5.60.3.4 bool Seize::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

5.60.3.5 unsigned int Seize::getAllocationType ( ) const

5.60.3.6 unsigned int Seize::getLastMemberSeized ( ) const

5.60.3.7 unsigned short Seize::getPriority ( ) const

5.60.3.8 std::string Seize::getQuantity ( ) const

5.60.3.9 Queue * Seize::getQueue ( ) const

5.60.3.10 std::string Seize::getQueueName ( ) const

Here is the call graph for this function:

5.60.3.11 Resource * Seize::getResource ( ) const

5.60.3.12 std::string Seize::getResourceName ( ) const

Here is the call graph for this function:

5.60.3.13 Resource::ResourceType Seize::getResourceType ( ) const

5.60.3.14 Resource::ResourceRule Seize::getRule ( ) const

5.60.3.15 std::string Seize::getSaveAttribute ( ) const

5.60.3.16 void Seize::setAllocationType (unsigned int *_allocationType*)

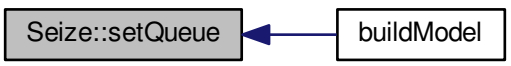
5.60.3.17 void Seize::setLastMemberSeized (unsigned int *_lastMemberSeized*)

5.60.3.18 void Seize::setPriority (unsigned short *_priority*)

5.60.3.19 void Seize::setQuantity (std::string *_quantity*)

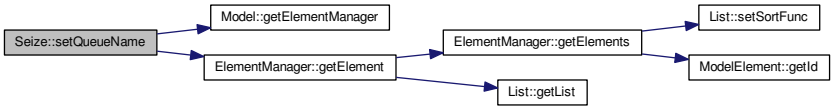
5.60.3.20 void Seize::setQueue (Queue * *queue*)

Here is the caller graph for this function:



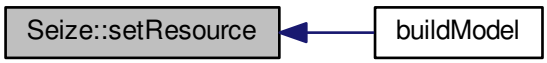
5.60.3.21 void Seize::setQueueName (std::string *queueName*) throw)

Here is the call graph for this function:



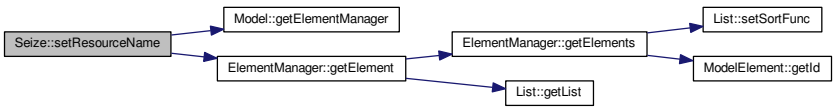
5.60.3.22 void Seize::setResource (Resource * *resource*)

Here is the caller graph for this function:



5.60.3.23 void Seize::setResourceName (std::string *_resourceName*) throw)

Here is the call graph for this function:



5.60.3.24 void Seize::setResourceType (Resource::ResourceType *_resourceType*)

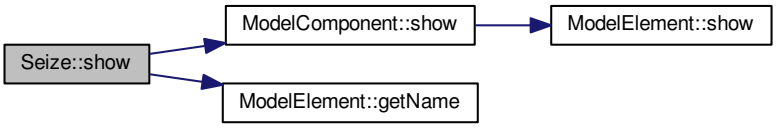
5.60.3.25 void Seize::setRule (Resource::ResourceRule *_rule*)

5.60.3.26 void Seize::setSaveAttribute (std::string *_saveAttribute*)

5.60.3.27 std::string Seize::show ( ) [virtual]

Reimplemented from ModelComponent.

Here is the call graph for this function:



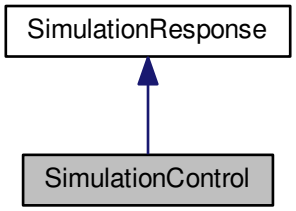
The documentation for this class was generated from the following files:

- Seize.h
- Seize.cpp

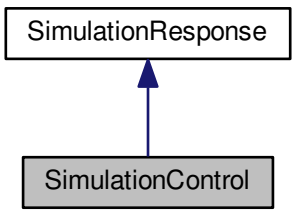
5.61 SimulationControl Class Reference

`#include <SimulationControl.h>`

Inheritance diagram for SimulationControl:



Collaboration diagram for SimulationControl:



Public Member Functions

- SimulationControl (std::string type, std::string name, GetterMember getterMember, SetterMember setter↩ Member)
- SimulationControl (const SimulationControl &orig)
- virtual ~SimulationControl ()
- void setValue (double value)

Additional Inherited Members

5.61.1 Detailed Description

Represents any possible parameter or control for a simulation. Any element or event the model can declare one of its own attribute as a simulation control. It just have to create a SimulationControl object, passing the access to the methods that gets and sets the control value and including this SimulationControl in the corresponding list of the model

5.61.2 Constructor & Destructor Documentation

5.61.2.1 SimulationControl::SimulationControl (std::string *type,* std::string *name,* GetterMember *getterMember,* SetterMember *setterMember*)

5.61.2.2 SimulationControl::SimulationControl (const SimulationControl & *orig*)

5.61.2.3 SimulationControl::~SimulationControl ( ) [virtual]

5.61.3 Member Function Documentation

5.61.3.1 void SimulationControl::setValue (double *value*)

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

- SimulationControl.h
- SimulationControl.cpp

5.62 SimulationEvent Class Reference

```
#include <OnEventManager.h>
```

Public Member Functions

- SimulationEvent (unsigned int replicationNumber, Event *event)
- unsigned int getReplicationNumber () const
- Event * getEventProcessed () const

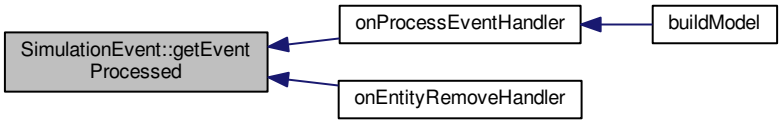
5.62.1 Constructor & Destructor Documentation

5.62.1.1 SimulationEvent::SimulationEvent (unsigned int *replicationNumber,* Event * *event*) [inline]

5.62.2 Member Function Documentation

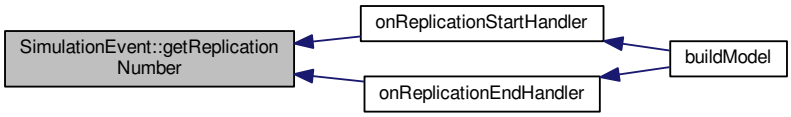
5.62.2.1 Event* SimulationEvent::getEventProcessed ( ) const [inline]

Here is the caller graph for this function:



5.62.2.2 unsigned int SimulationEvent::getReplicationNumber ( ) const [inline]

Here is the caller graph for this function:



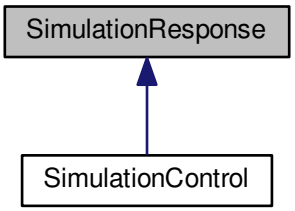
The documentation for this class was generated from the following file:

- OnEventManager.h

5.63 SimulationResponse Class Reference

`#include <SimulationResponse.h>`

Inheritance diagram for SimulationResponse:



Public Member Functions

- SimulationResponse (std::string type, std::string name, GetterMember getterMember)
- SimulationResponse (const SimulationResponse &orig)
- virtual ~SimulationResponse ()
- double getValue ()
- std::string getName () const
- std::string getType () const

Protected Attributes

- std::string _type
- std::string _name
- GetterMember _getterMemberFunction

5.63.1 Detailed Description

Represents any possible response of a simulation. Any element or event the model can declare one of its own attribute as a simulation response. It just have to create a SimulationResponse object, passing the access to the method that gets the response value and including this SimulationResponse in the corresponding list of the model

5.63.2 Constructor & Destructor Documentation

5.63.2.1 SimulationResponse::SimulationResponse (std::string *type,* std::string *name,* GetterMember *getterMember*)

5.63.2.2 SimulationResponse::SimulationResponse (const SimulationResponse & *orig*)

5.63.2.3 SimulationResponse::~SimulationResponse ( ) [virtual]

5.63.3 Member Function Documentation

5.63.3.1 std::string SimulationResponse::getName ( ) const

5.63.3.2 std::string SimulationResponse::getType ( ) const

5.63.3.3 double SimulationResponse::getValue ( )

5.63.4 Member Data Documentation

5.63.4.1 GetterMember SimulationResponse::_getterMemberFunction [protected]

5.63.4.2 std::string SimulationResponse::_name [protected]

5.63.4.3 std::string SimulationResponse::_type [protected]

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

- SimulationResponse.h
- SimulationResponse.cpp

5.64 SimulationScenario Class Reference

`#include <SimulationScenario.h>`

Public Member Functions

- SimulationScenario ()
- SimulationScenario (const SimulationScenario &orig)
- virtual ~SimulationScenario ()
- void setName (std::string _name)
- std::string getName () const
- std::list< double > * getResponseValues () const
- std::list< double > * getControlValues () const
- void setModelFilename (std::string _modelFilename)
- std::string getModelFilename () const
- double getResponseValue (SimulationResponse *value)
- double getControlValue (SimulationControl *control)
- void setControlValue (SimulationControl *control, double value)

5.64.1 Detailed Description

Represents a scenario where a specific model (defined my ModelFilename) will be simulated. To each scenario will be associated a set of SimulationControl and SimulationResponse, and their values are set to the scenario by the ProcessAnalyser.

5.64.2 Constructor & Destructor Documentation

5.64.2.1 SimulationScenario::SimulationScenario ( )

5.64.2.2 SimulationScenario::SimulationScenario (const SimulationScenario & *orig*)

5.64.2.3 SimulationScenario::~SimulationScenario ( ) [virtual]

5.64.3 Member Function Documentation

5.64.3.1 double SimulationScenario::getControlValue (SimulationControl * *control*)

5.64.3.2 std::list< double > * SimulationScenario::getControlValues ( ) const

5.64.3.3 std::string SimulationScenario::getModelFilename ( ) const

5.64.3.4 std::string SimulationScenario::getName ( ) const

5.64.3.5 double SimulationScenario::getResponseValue (SimulationResponse * *value*)

5.64.3.6 std::list< double > * SimulationScenario::getResponseValues ( ) const

5.64.3.7 void SimulationScenario::setControlValue (SimulationControl * *control*, double *value*)

5.64.3.8 void SimulationScenario::setModelFilename (std::string *_modelFilename*)

5.64.3.9 void SimulationScenario::setName (std::string *_name*)

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

- SimulationScenario.h
- SimulationScenario.cpp

5.65 Simulator Class Reference

```
#include <Simulator.h>
```

Public Member Functions

- Simulator ()
- Simulator (const Simulator &orig)
- virtual ~Simulator ()
- List< Model * > * getModels () const

  Returns the list of open models in the simulator.
- List< Plugin * > * getPlugins () const
- std::string getVersion () const
- std::string getLicense () const
- std::string getName () const
- Sampler_if * getSampler () const

  Returns the Sampler, used to generate samples accordingly to a probability distribution.
- Fitter_if * getFitter () const

  Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

5.65.1 Detailed Description

The main class of the ReGenesys KERNEL simulation. It gives access to simulation models and tools.

5.65.2 Constructor & Destructor Documentation

5.65.2.1 Simulator::Simulator ( )

5.65.2.2 Simulator::Simulator (const Simulator & *orig*)

5.65.2.3 Simulator::~Simulator ( ) [virtual]

5.65.3 Member Function Documentation

5.65.3.1 Fitter_if * Simulator::getFitter ( ) const

Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

Here is the caller graph for this function:

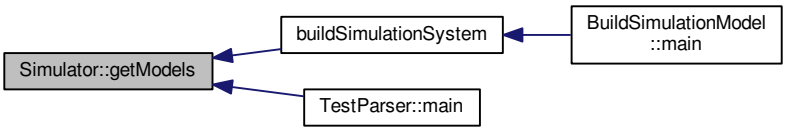
5.65.3.2 std::string Simulator::getLicense ( ) const

Here is the caller graph for this function:

5.65.3.3 List< Model * > * Simulator::getModels ( ) const

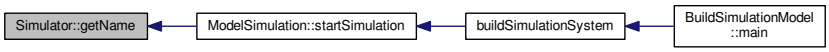
Returns the list of open models in the simulator.

Here is the caller graph for this function:



5.65.3.4 std::string Simulator::getName ( ) const

Here is the caller graph for this function:

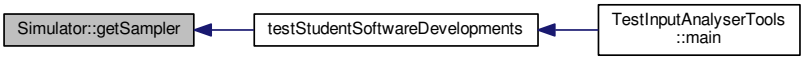


5.65.3.5 List< Plugin * > * Simulator::getPlugins ( ) const

5.65.3.6 Sampler_if * Simulator::getSampler ( ) const

Returns the Sampler, used to generate samples accordingly to a probability distribution.

Here is the caller graph for this function:



5.65.3.7 std::string Simulator::getVersion ( ) const

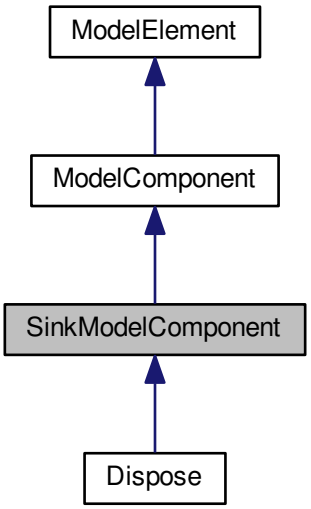
The documentation for this class was generated from the following files:

- Simulator.h
- Simulator.cpp

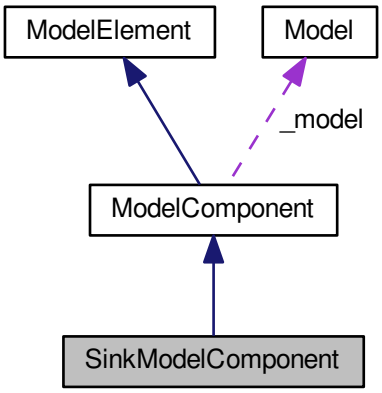
5.66 SinkModelComponent Class Reference

```
#include <SinkModelComponent.h>
```

Inheritance diagram for SinkModelComponent:



Collaboration diagram for SinkModelComponent:



Public Member Functions

- SinkModelComponent (Model *model)
- SinkModelComponent (const SinkModelComponent &orig)
- virtual ~SinkModelComponent ()
- void setCollectStatistics (bool _collectStatistics)
- bool isCollectStatistics () const

Additional Inherited Members

5.66.1 Detailed Description

This class is the basis for any component representing the end of a process flow, such as a Dispose. It can remove entities from the system and collect statistics.

5.66.2 Constructor & Destructor Documentation

5.66.2.1 SinkModelComponent::SinkModelComponent (Model * *model*)

5.66.2.2 SinkModelComponent::SinkModelComponent (const SinkModelComponent & *orig*)

5.66.2.3 SinkModelComponent::~SinkModelComponent ( ) [virtual]

5.66.3 Member Function Documentation

5.66.3.1 bool SinkModelComponent::isCollectStatistics ( ) const

5.66.3.2 void SinkModelComponent::setCollectStatistics (bool *_collectStatistics*)

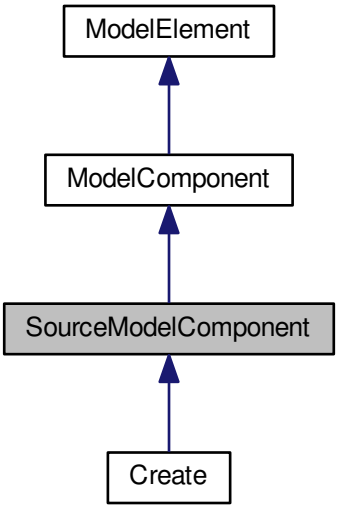
The documentation for this class was generated from the following files:

- SinkModelComponent.h
- SinkModelComponent.cpp

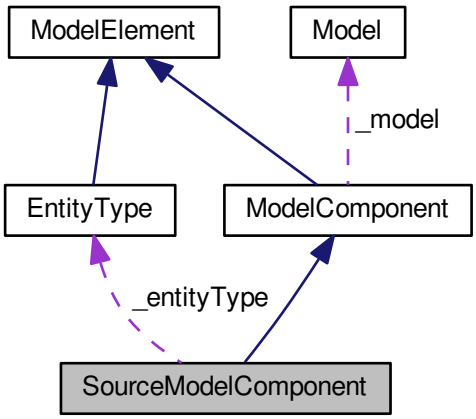
5.67 SourceModelComponent Class Reference

`#include <SourceModelComponent.h>`

Inheritance diagram for SourceModelComponent:



Collaboration diagram for SourceModelComponent:



Public Member Functions

- SourceModelComponent (Model *model)
- SourceModelComponent (const SourceModelComponent &orig)
- virtual ~SourceModelComponent ()
- void setFirstCreation (double _firstCreation)
- double getFirstCreation () const
- void setCollectStatistics (bool _collectStatistics)
- bool isCollectStatistics () const
- void setEntityType (EntityType *_entityType)
- EntityType * getEntityType () const
- void setTimeUnit (Util::TimeUnit _timeUnit)
- Util::TimeUnit getTimeUnit () const
- void setTimeBetweenCreationsExpression (std::string _timeBetweenCreations)
- std::string getTimeBetweenCreationsExpression () const
- void setMaxCreations (unsigned int _maxCreations)
- unsigned int getMaxCreations () const
- unsigned int getEntitiesCreated () const
- void setEntitiesCreated (unsigned int _entitiesCreated)
- void setEntitiesPerCreation (unsigned int _entitiesPerCreation)
- unsigned int getEntitiesPerCreation () const
- virtual std::string show ()

Protected Attributes

- EntityType * _entityType
- double _firstCreation = 0.0
- unsigned int _entitiesPerCreation = 1
- unsigned int _maxCreations = std::numeric_limits<unsigned int>::max()
- std::string _timeBetweenCreationsExpression = "10"
- Util::TimeUnit _timeBetweenCreationsTimeUnit = Util::TimeUnit::second
- bool _collectStatistics = true
- unsigned int _entitiesCreatedSoFar = 0

Additional Inherited Members

5.67.1 Detailed Description

A source component implements the base for inserting entities into the model when its simulation is initialized. During the initialization, the new and empty future events list is populated by events of creating entities and sending them to the source components existing in the model

5.67.2 Constructor & Destructor Documentation

5.67.2.1 SourceModelComponent::SourceModelComponent (Model * *model*)

5.67.2.2 SourceModelComponent::SourceModelComponent (const SourceModelComponent & *orig*)

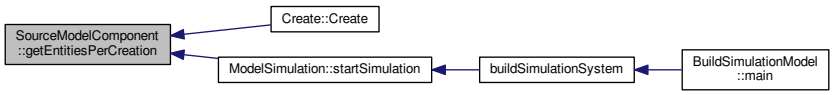
5.67.2.3 SourceModelComponent::~SourceModelComponent ( ) [virtual]

5.67.3 Member Function Documentation

5.67.3.1 unsigned int SourceModelComponent::getEntitiesCreated ( ) const

5.67.3.2 unsigned int SourceModelComponent::getEntitiesPerCreation ( ) const

Here is the caller graph for this function:



5.67.3.3 EntityType * SourceModelComponent::getEntityType ( ) const

5.67.3.4 double SourceModelComponent::getFirstCreation ( ) const

Here is the caller graph for this function:

5.67.3.5 unsigned int SourceModelComponent::getMaxCreations ( ) const

5.67.3.6 std::string SourceModelComponent::getTimeBetweenCreationsExpression ( ) const

5.67.3.7 Util::TimeUnit SourceModelComponent::getTimeUnit ( ) const

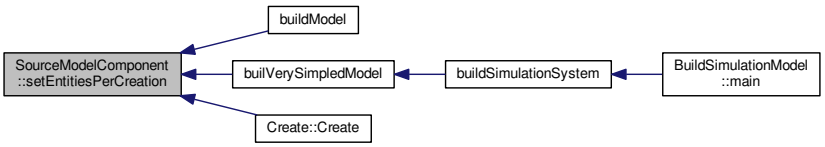
5.67.3.8 bool SourceModelComponent::isCollectStatistics ( ) const

5.67.3.9 void SourceModelComponent::setCollectStatistics (bool *_collectStatistics*)

5.67.3.10 void SourceModelComponent::setEntitiesCreated (unsigned int *_entitiesCreated*)

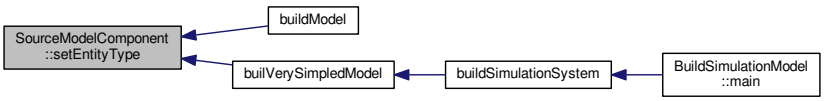
5.67.3.11 void SourceModelComponent::setEntitiesPerCreation (unsigned int *_entitiesPerCreation*)

Here is the caller graph for this function:



5.67.3.12 void SourceModelComponent::setEntityType (EntityType * *_entityType*)

Here is the caller graph for this function:

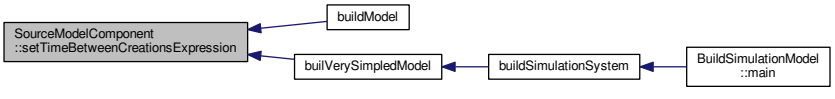


5.67.3.13 void SourceModelComponent::setFirstCreation (double *_firstCreation*)

5.67.3.14 void SourceModelComponent::setMaxCreations (unsigned int *_maxCreations*)

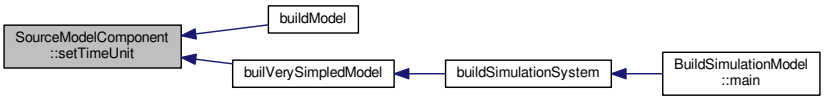
5.67.3.15 void SourceModelComponent::setTimeBetweenCreationsExpression (std::string *_timeBetweenCreations*)

Here is the caller graph for this function:



5.67.3.16 void SourceModelComponent::setTimeUnit (Util::TimeUnit *_timeUnit*)

Here is the caller graph for this function:

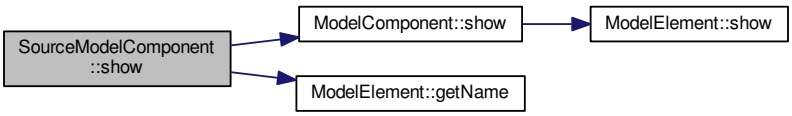


5.67.3.17 std::string SourceModelComponent::show ( ) [virtual]

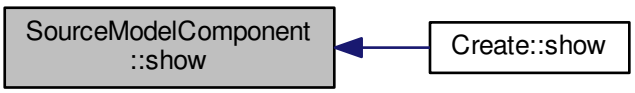
Reimplemented from ModelComponent.

Reimplemented in Create.

Here is the call graph for this function:



Here is the caller graph for this function:



5.67.4 Member Data Documentation

5.67.4.1 bool SourceModelComponent::_collectStatistics = true [protected]

5.67.4.2 unsigned int SourceModelComponent::_entitiesCreatedSoFar = 0 [protected]

5.67.4.3 unsigned int SourceModelComponent::_entitiesPerCreation = 1 [protected]

5.67.4.4 EntityType* SourceModelComponent::_entityType [protected]

5.67.4.5 double SourceModelComponent::_firstCreation = 0.0 [protected]

5.67.4.6 unsigned int SourceModelComponent::_maxCreations = std::numeric_limits<unsigned int>::max() [protected]

5.67.4.7 std::string SourceModelComponent::_timeBetweenCreationsExpression = "10" [protected]

5.67.4.8 Util::TimeUnit SourceModelComponent::_timeBetweenCreationsTimeUnit = Util::TimeUnit::second [protected]

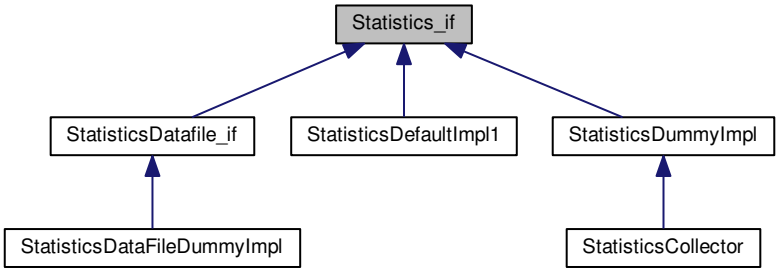
The documentation for this class was generated from the following files:

- SourceModelComponent.h
- SourceModelComponent.cpp

5.68 Statistics_if Class Reference

`#include <Statistics_if.h>`

Inheritance diagram for Statistics_if:



Public Member Functions

- virtual Collector_if * getCollector ()=0
- virtual void setCollector (Collector_if *collector)=0
- virtual unsigned int numElements ()=0
- virtual double min ()=0
- virtual double max ()=0
- virtual double average ()=0
- virtual double variance ()=0
- virtual double stddeviation ()=0
- virtual double variationCoef ()=0
- virtual double halfWidthConfidenceInterval (double confidencelevel)=0
- virtual unsigned int newSampleSize (double confidencelevel, double halfWidth)=0

5.68.1 Detailed Description

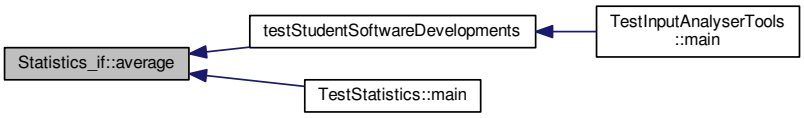
Interface for statisct synthesis of a stochastic variable collected by a Collector_if. The statistics generated may be updated based only on the previous statistics and the single newest added value or they may be updated based on a datafile, depending on the Collector implementation.

5.68.2 Member Function Documentation

5.68.2.1 virtual double Statistics_if::average ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

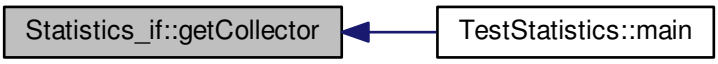
Here is the caller graph for this function:



5.68.2.2 virtual Collector_if* Statistics_if::getCollector ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

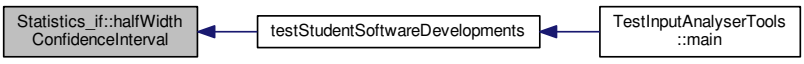
Here is the caller graph for this function:



5.68.2.3 virtual double Statistics_if::halfWidthConfidenceInterval (double *confidencelevel*) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

Here is the caller graph for this function:



5.68.2.4 virtual double Statistics_if::max ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

Here is the caller graph for this function:

5.68.2.5 virtual double Statistics_if::min ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

Here is the caller graph for this function:

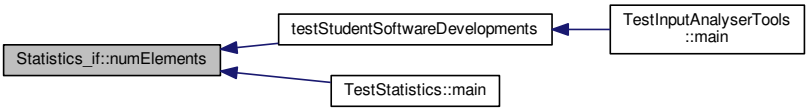
5.68.2.6 virtual unsigned int Statistics_if::newSampleSize (double *confidencelevel,* double *halfWidth*) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

5.68.2.7 virtual unsigned int Statistics_if::numElements ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

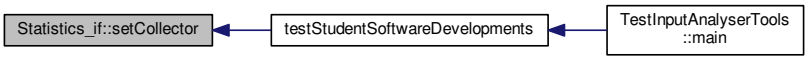
Here is the caller graph for this function:



5.68.2.8 virtual void Statistics_if::setCollector (Collector_if * *collector*) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

Here is the caller graph for this function:



5.68.2.9 virtual double Statistics_if::stddeviation ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

Here is the caller graph for this function:

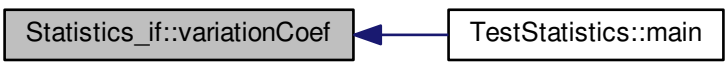
5.68.2.10 virtual double Statistics_if::variance ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

5.68.2.11 virtual double Statistics_if::variationCoef ( ) [pure virtual]

Implemented in StatisticsDefaultImpl1, StatisticsDummyImpl, and StatisticsDataFileDummyImpl.

Here is the caller graph for this function:



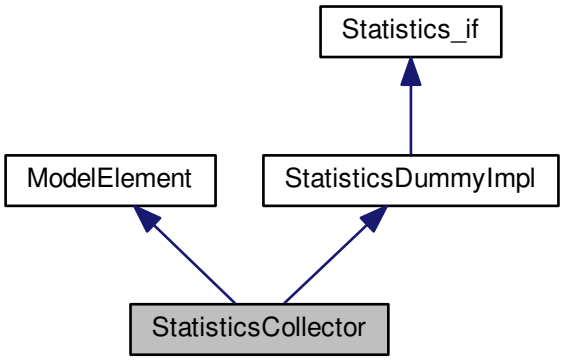
The documentation for this class was generated from the following file:

- Statistics_if.h

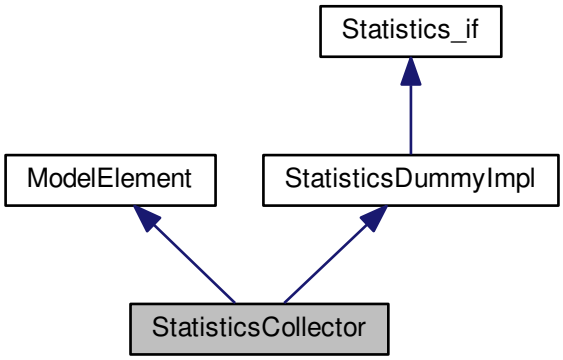
5.69 StatisticsCollector Class Reference

`#include <StatisticsCollector.h>`

Inheritance diagram for StatisticsCollector:



Collaboration diagram for StatisticsCollector:



Public Member Functions

- StatisticsCollector ()
- StatisticsCollector (std::string name)
- StatisticsCollector (std::string name, ModelElement *parent)
- StatisticsCollector (const StatisticsCollector &orig)
- virtual ~StatisticsCollector ()
- virtual std::string show ()
- ModelElement * getParent () const

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.69.1 Constructor & Destructor Documentation

5.69.1.1 StatisticsCollector::StatisticsCollector ( )

5.69.1.2 StatisticsCollector::StatisticsCollector (std::string *name*)

5.69.1.3 StatisticsCollector::StatisticsCollector (std::string *name*, ModelElement * *parent*)

5.69.1.4 StatisticsCollector::StatisticsCollector (const StatisticsCollector & *orig*)

5.69.1.5 StatisticsCollector::~StatisticsCollector ( ) [virtual]

5.69.2 Member Function Documentation

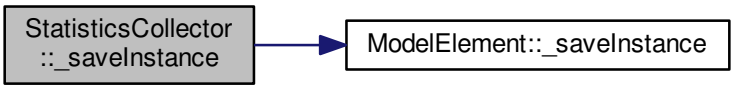
5.69.2.1 void StatisticsCollector::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.69.2.2 std::list< std::string > * StatisticsCollector::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



5.69.2.3 bool StatisticsCollector::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

Implements ModelElement.

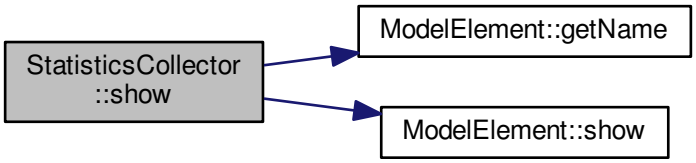
5.69.2.4 ModelElement * StatisticsCollector::getParent ( ) const

Here is the caller graph for this function:

5.69.2.5 std::string StatisticsCollector::show ( ) [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



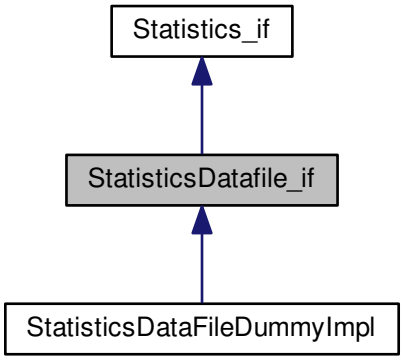
The documentation for this class was generated from the following files:

- StatisticsCollector.h
- StatisticsCollector.cpp

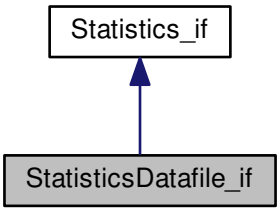
5.70 StatisticsDatafile_if Class Reference

```
#include <StatisticsDataFile_if.h>
```

Inheritance diagram for StatisticsDatafile_if:



Collaboration diagram for StatisticsDatafile_if:



Public Member Functions

- virtual double mode ()=0
- virtual double mediane ()=0
- virtual double quartil (unsigned short num)=0
- virtual double decil (unsigned short num)=0
- virtual double centil (unsigned short num)=0
- virtual void setHistogramNumClasses (unsigned short num)=0
- virtual unsigned short histogramNumClasses ()=0
- virtual double histogramClassLowerLimit (unsigned short classNum)=0
- virtual unsigned int histogramClassFrequency (unsigned short classNum)=0

5.70.1 Member Function Documentation

5.70.1.1 virtual double StatisticsDatafile_if::centil (unsigned short *num*) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.2 virtual double StatisticsDatafile_if::decil (unsigned short *num*) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.3 virtual unsigned int StatisticsDatafile_if::histogramClassFrequency (unsigned short *classNum*) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.4 virtual double StatisticsDatafile_if::histogramClassLowerLimit (unsigned short *classNum*) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.5 virtual unsigned short StatisticsDatafile_if::histogramNumClasses ( ) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.6 virtual double StatisticsDatafile_if::mediane ( ) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.7 virtual double StatisticsDatafile_if::mode ( ) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.8 virtual double StatisticsDatafile_if::quartil (unsigned short *num*) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

5.70.1.9 virtual void StatisticsDatafile_if::setHistogramNumClasses (unsigned short *num*) [pure virtual]

Implemented in StatisticsDataFileDummyImpl.

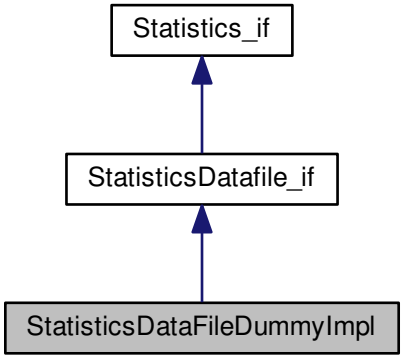
The documentation for this class was generated from the following file:

- StatisticsDataFile_if.h

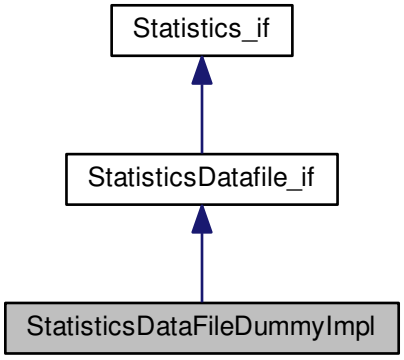
5.71 StatisticsDataFileDummyImpl Class Reference

`#include <StatisticsDataFileDummyImpl.h>`

Inheritance diagram for StatisticsDataFileDummyImpl:



Collaboration diagram for StatisticsDataFileDummyImpl:



Public Member Functions

- StatisticsDataFileDummyImpl ()
- StatisticsDataFileDummyImpl (const StatisticsDataFileDummyImpl &orig)
- virtual ~StatisticsDataFileDummyImpl ()
- virtual Collector_if * getCollector ()
- void setCollector (Collector_if *collector)

- virtual unsigned int numElements ()
- virtual double min ()
- virtual double max ()
- virtual double average ()
- virtual double variance ()
- virtual double stddeviation ()
- virtual double variationCoef ()
- virtual double halfWidthConfidenceInterval (double confidencelevel)
- virtual unsigned int newSampleSize (double confidencelevel, double halfWidth)
- virtual double mode ()
- virtual double mediane ()
- virtual double quartil (unsigned short num)
- virtual double decil (unsigned short num)
- virtual double centil (unsigned short num)
- virtual void setHistogramNumClasses (unsigned short num)
- virtual unsigned short histogramNumClasses ()
- virtual double histogramClassLowerLimit (unsigned short classNum)
- virtual unsigned int histogramClassFrequency (unsigned short classNum)

5.71.1 Constructor & Destructor Documentation

5.71.1.1 StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl ( )

5.71.1.2 StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl (const StatisticsDataFileDummyImpl & *orig*)

5.71.1.3 StatisticsDataFileDummyImpl::~StatisticsDataFileDummyImpl ( ) [virtual]

5.71.2 Member Function Documentation

5.71.2.1 double StatisticsDataFileDummyImpl::average ( ) [virtual]

Implements Statistics_if.

5.71.2.2 double StatisticsDataFileDummyImpl::centil (unsigned short *num*) [virtual]

Implements StatisticsDatafile_if.

5.71.2.3 double StatisticsDataFileDummyImpl::decil (unsigned short *num*) [virtual]

Implements StatisticsDatafile_if.

5.71.2.4 Collector_if * StatisticsDataFileDummyImpl::getCollector ( ) [virtual]

Implements Statistics_if.

5.71.2.5 double StatisticsDataFileDummyImpl::halfWidthConfidenceInterval (double *confidencelevel*) [virtual]

Implements Statistics_if.

5.71.2.6 unsigned int StatisticsDataFileDummyImpl::histogramClassFrequency (unsigned short *classNum*) [virtual]

Implements StatisticsDatafile_if.

5.71.2.7 double StatisticsDataFileDummyImpl::histogramClassLowerLimit (unsigned short *classNum*) [virtual]

Implements StatisticsDatafile_if.

5.71.2.8 unsigned short StatisticsDataFileDummyImpl::histogramNumClasses ( ) [virtual]

Implements StatisticsDatafile_if.

5.71.2.9 double StatisticsDataFileDummyImpl::max ( ) [virtual]

Implements Statistics_if.

5.71.2.10 double StatisticsDataFileDummyImpl::mediane ( ) [virtual]

Implements StatisticsDatafile_if.

5.71.2.11 double StatisticsDataFileDummyImpl::min ( ) [virtual]

Implements Statistics_if.

5.71.2.12 double StatisticsDataFileDummyImpl::mode ( ) [virtual]

Implements StatisticsDatafile_if.

5.71.2.13 unsigned int StatisticsDataFileDummyImpl::newSampleSize (double *confidencelevel,* double *halfWidth*) [virtual]

Implements Statistics_if.

5.71.2.14 unsigned int StatisticsDataFileDummyImpl::numElements ( ) [virtual]

Implements Statistics_if.

5.71.2.15 double StatisticsDataFileDummyImpl::quartil (unsigned short *num*) [virtual]

Implements StatisticsDatafile_if.

5.71.2.16 void StatisticsDataFileDummyImpl::setCollector (Collector_if * *collector*) [virtual]

Implements Statistics_if.

5.71.2.17 void StatisticsDataFileDummyImpl::setHistogramNumClasses (unsigned short *num*) [virtual]

Implements StatisticsDatafile_if.

5.71.2.18 double StatisticsDataFileDummyImpl::stddeviation ( ) [virtual]

Implements Statistics_if.

5.71.2.19 double StatisticsDataFileDummyImpl::variance ( ) [virtual]

Implements Statistics_if.

5.71.2.20 double StatisticsDataFileDummyImpl::variationCoef ( ) [virtual]

Implements Statistics_if.

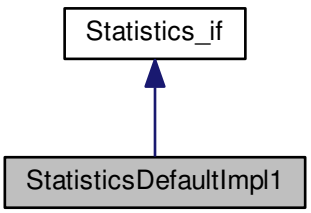
The documentation for this class was generated from the following files:

- StatisticsDataFileDummyImpl.h
- StatisticsDataFileDummyImpl.cpp

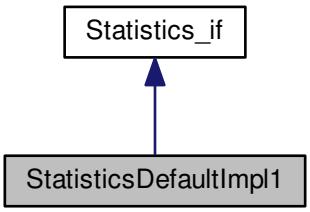
5.72 StatisticsDefaultImpl1 Class Reference

`#include <StatisticsDefaultImpl1.h>`

Inheritance diagram for StatisticsDefaultImpl1:



Collaboration diagram for StatisticsDefaultImpl1:



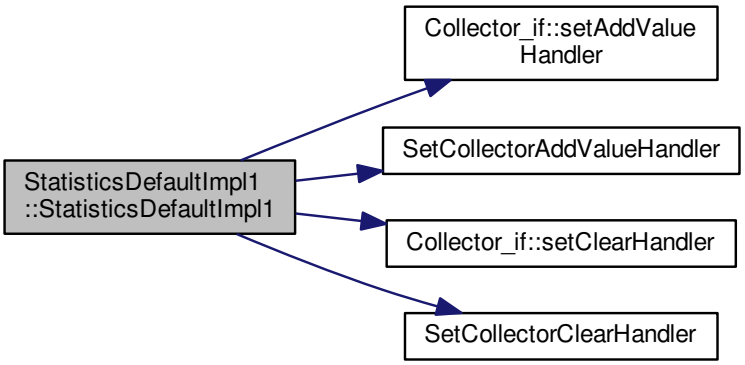
Public Member Functions

- StatisticsDefaultImpl1 ()
- StatisticsDefaultImpl1 (const StatisticsDefaultImpl1 &orig)
- virtual ~StatisticsDefaultImpl1 ()
- virtual Collector_if * getCollector ()
- void setCollector (Collector_if *collector)
- virtual unsigned int numElements ()
- virtual double min ()
- virtual double max ()
- virtual double average ()
- virtual double variance ()
- virtual double stddeviation ()
- virtual double variationCoef ()
- virtual double halfWidthConfidenceInterval (double confidencelevel)
- virtual unsigned int newSampleSize (double confidencelevel, double halfWidth)

5.72.1 Constructor & Destructor Documentation

5.72.1.1 StatisticsDefaultImpl1::StatisticsDefaultImpl1 ( )

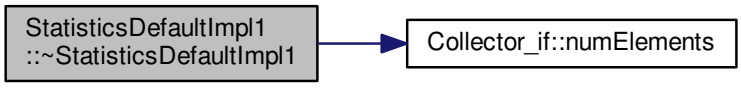
Here is the call graph for this function:



5.72.1.2 StatisticsDefaultImpl1::StatisticsDefaultImpl1 (const StatisticsDefaultImpl1 & *orig*)

5.72.1.3 StatisticsDefaultImpl1::~StatisticsDefaultImpl1 ( ) [virtual]

Here is the call graph for this function:



5.72.2 Member Function Documentation

5.72.2.1 double StatisticsDefaultImpl1::average ( ) [virtual]

Implements Statistics_if.

5.72.2.2 Collector_if * StatisticsDefaultImpl1::getCollector ( ) [virtual]

Implements Statistics_if.

Here is the caller graph for this function:

5.72.2.3 double StatisticsDefaultImpl1::halfWidthConfidenceInterval (double *confidencelevel*) [virtual]

Implements Statistics_if.

5.72.2.4 double StatisticsDefaultImpl1::max ( ) [virtual]

Implements Statistics_if.

5.72.2.5 double StatisticsDefaultImpl1::min ( ) [virtual]

Implements Statistics_if.

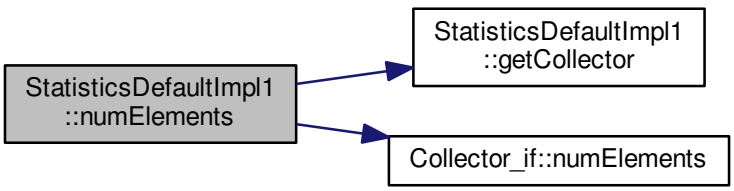
5.72.2.6 unsigned int StatisticsDefaultImpl1::newSampleSize (double *confidencelevel,* double *halfWidth*) [virtual]

Implements Statistics_if.

5.72.2.7 unsigned int StatisticsDefaultImpl1::numElements ( ) [virtual]

Implements Statistics_if.

Here is the call graph for this function:



5.72.2.8 void StatisticsDefaultImpl1::setCollector (Collector_if * *collector*) [virtual]

Implements Statistics_if.

5.72.2.9 double StatisticsDefaultImpl1::stddeviation ( ) [virtual]

Implements Statistics_if.

5.72.2.10 double StatisticsDefaultImpl1::variance ( ) [virtual]

Implements Statistics_if.

5.72.2.11 double StatisticsDefaultImpl1::variationCoef ( ) [virtual]

Implements Statistics_if.

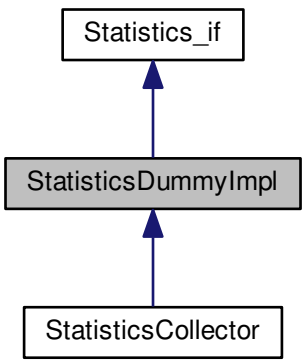
The documentation for this class was generated from the following files:

- StatisticsDefaultImpl1.h
- StatisticsDefaultImpl1.cpp

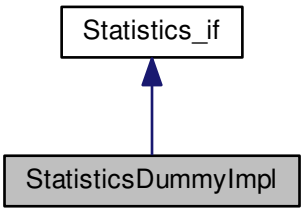
5.73 StatisticsDummyImpl Class Reference

`#include <StatisticsDummyImpl.h>`

Inheritance diagram for StatisticsDummyImpl:



Collaboration diagram for StatisticsDummyImpl:



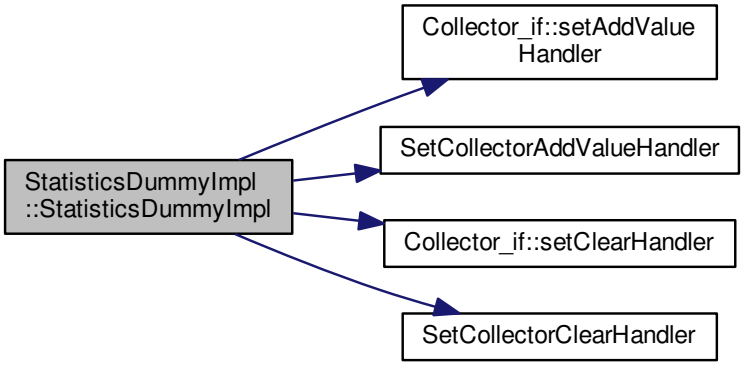
Public Member Functions

- StatisticsDummyImpl ()
- StatisticsDummyImpl (const StatisticsDummyImpl &orig)
- virtual ~StatisticsDummyImpl ()
- virtual Collector_if * getCollector ()
- void setCollector (Collector_if *collector)
- virtual unsigned int numElements ()
- virtual double min ()
- virtual double max ()
- virtual double average ()
- virtual double variance ()
- virtual double stddeviation ()
- virtual double variationCoef ()
- virtual double halfWidthConfidenceInterval (double confidencelevel)
- virtual unsigned int newSampleSize (double confidencelevel, double halfWidth)

5.73.1 Constructor & Destructor Documentation

5.73.1.1 StatisticsDummyImpl::StatisticsDummyImpl ( )

Here is the call graph for this function:



5.73.1.2 StatisticsDummyImpl::StatisticsDummyImpl (const StatisticsDummyImpl & *orig*)

5.73.1.3 StatisticsDummyImpl::~StatisticsDummyImpl ( ) [virtual]

5.73.2 Member Function Documentation

5.73.2.1 double StatisticsDummyImpl::average ( ) [virtual]

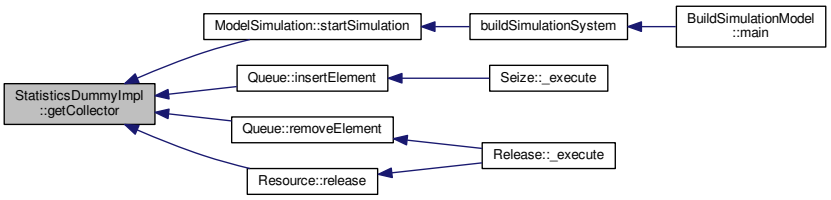
Implements Statistics_if.

Here is the caller graph for this function:

5.73.2.2 Collector_if * StatisticsDummyImpl::getCollector ( ) [virtual]

Implements Statistics_if.

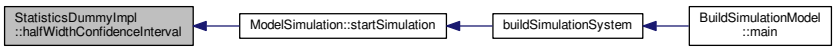
Here is the caller graph for this function:



5.73.2.3 double StatisticsDummyImpl::halfWidthConfidenceInterval (double *confidencelevel*) [virtual]

Implements Statistics_if.

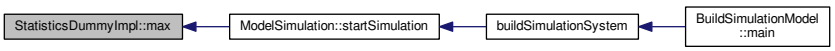
Here is the caller graph for this function:



5.73.2.4 double StatisticsDummyImpl::max ( ) [virtual]

Implements Statistics_if.

Here is the caller graph for this function:



5.73.2.5 double StatisticsDummyImpl::min ( ) [virtual]

Implements Statistics_if.

Here is the caller graph for this function:

5.73.2.6 unsigned int StatisticsDummyImpl::newSampleSize (double *confidencelevel,* double *halfWidth*) [virtual]

Implements Statistics_if.

5.73.2.7 unsigned int StatisticsDummyImpl::numElements ( ) [virtual]

Implements Statistics_if.

Here is the caller graph for this function:

5.73.2.8 void StatisticsDummyImpl::setCollector (Collector_if * *collector*) [virtual]

Implements Statistics_if.

5.73.2.9 double StatisticsDummyImpl::stddeviation ( ) [virtual]

Implements Statistics_if.

Here is the caller graph for this function:

5.73.2.10 double StatisticsDummyImpl::variance ( ) [virtual]

Implements Statistics_if.

5.73.2.11 double StatisticsDummyImpl::variationCoef ( ) [virtual]

Implements Statistics_if.

Here is the caller graph for this function:

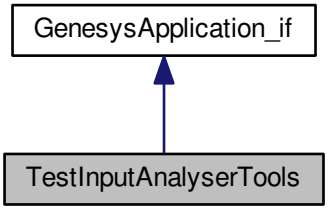
The documentation for this class was generated from the following files:

- StatisticsDummyImpl.h
- StatisticsDummyImpl.cpp

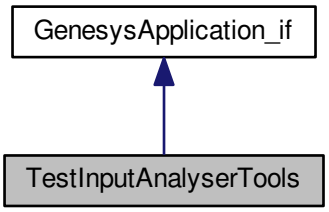
5.74 TestInputAnalyserTools Class Reference

`#include <TestInputAnalyserTools.h>`

Inheritance diagram for TestInputAnalyserTools:



Collaboration diagram for TestInputAnalyserTools:



Public Member Functions

- TestInputAnalyserTools ()
- int main (int argc, char **argv)

5.74.1 Constructor & Destructor Documentation

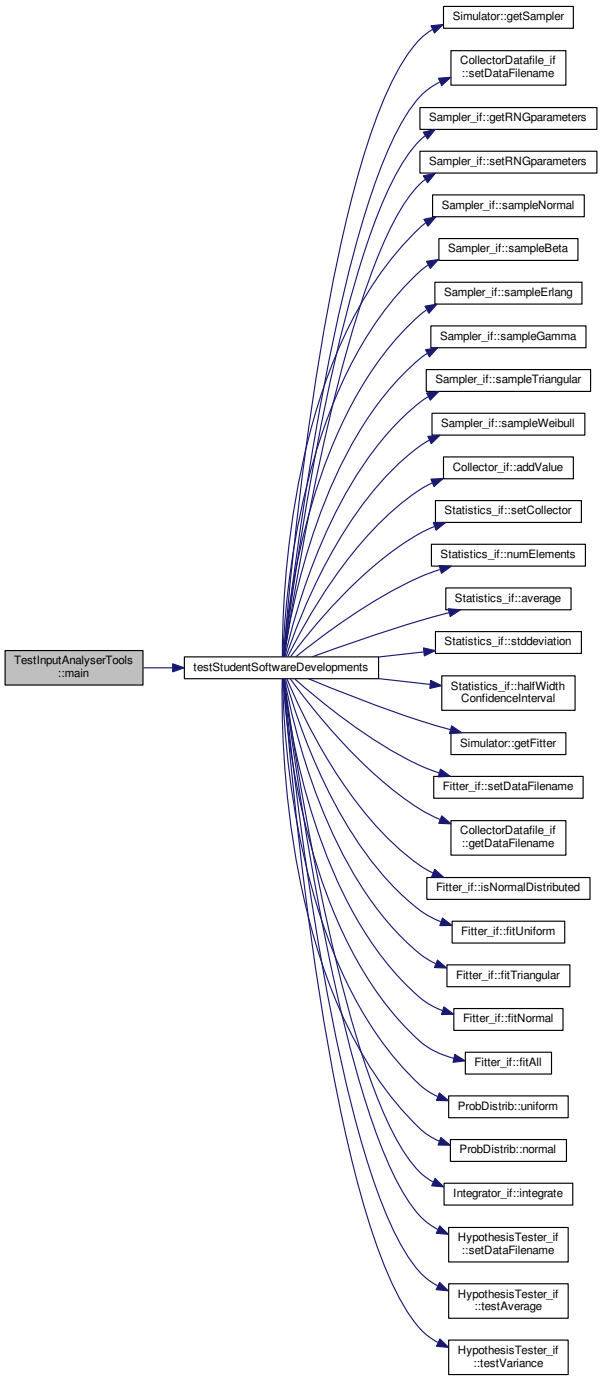
5.74.1.1 TestInputAnalyserTools::TestInputAnalyserTools ( )

5.74.2 Member Function Documentation

5.74.2.1 int TestInputAnalyserTools::main (int *argc,* char ** *argv*) [virtual]

Implements GenesysApplication_if.

Here is the call graph for this function:



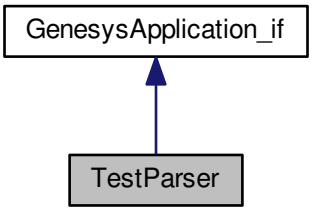
The documentation for this class was generated from the following files:

- TestInputAnalyserTools.h
- TestInputAnalyserTools.cpp

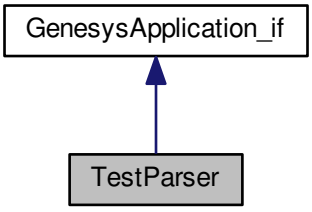
5.75 TestParser Class Reference

`#include <TestParser.h>`

Inheritance diagram for TestParser:



Collaboration diagram for TestParser:



Public Member Functions

- TestParser ()
- TestParser (const TestParser &orig)
- virtual ~TestParser ()
- virtual int main (int argc, char **argv)

5.75.1 Constructor & Destructor Documentation

5.75.1.1 TestParser::TestParser ( )

5.75.1.2 TestParser::TestParser (const TestParser & *orig*)

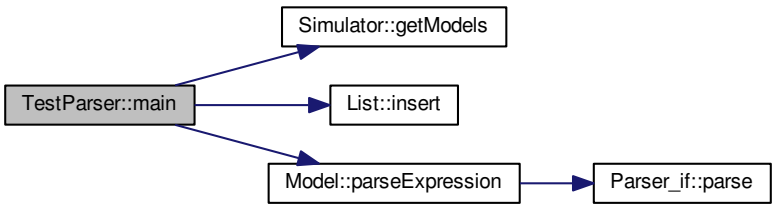
5.75.1.3 TestParser::~TestParser ( ) [virtual]

5.75.2 Member Function Documentation

5.75.2.1 int TestParser::main (int *argc*, char ** *argv*) [virtual]

Implements GenesysApplication_if.

Here is the call graph for this function:



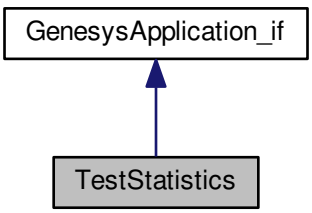
The documentation for this class was generated from the following files:

- TestParser.h
- TestParser.cpp

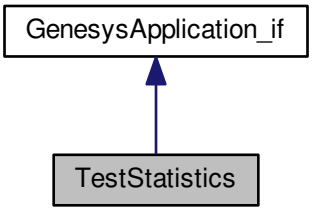
5.76 TestStatistics Class Reference

`#include <TestStatistics.h>`

Inheritance diagram for TestStatistics:



Collaboration diagram for TestStatistics:



Public Member Functions

- TestStatistics ()
- int main (int argc, char **argv)

5.76.1 Constructor & Destructor Documentation

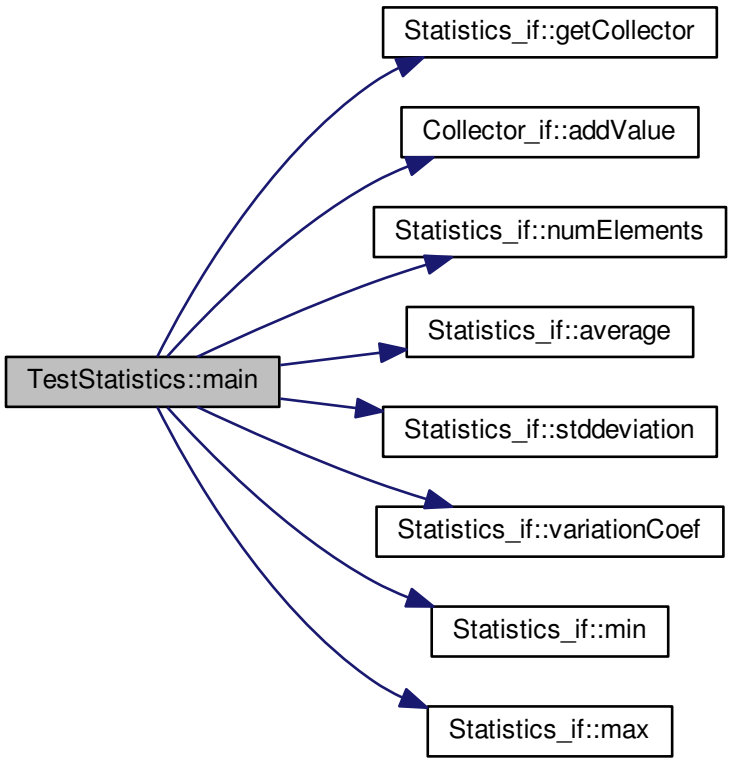
5.76.1.1 TestStatistics::TestStatistics ( )

5.76.2 Member Function Documentation

5.76.2.1 int TestStatistics::main (int *argc,* char ** *argv*) [virtual]

Implements GenesysApplication_if.

Here is the call graph for this function:



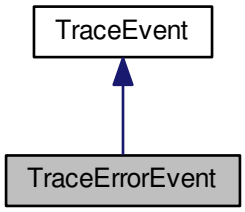
The documentation for this class was generated from the following files:

- TestStatistics.h
- TestStatistics.cpp

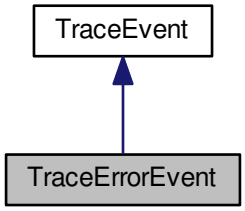
5.77 TraceErrorEvent Class Reference

`#include <TraceManager.h>`

Inheritance diagram for TraceErrorEvent:



Collaboration diagram for TraceErrorEvent:



Public Member Functions

- TraceErrorEvent (std::string text, std::exception e)
- std::exception getException () const

5.77.1 Constructor & Destructor Documentation

5.77.1.1 TraceErrorEvent::TraceErrorEvent (std::string *text,* std::exception *e*) [inline]

5.77.2 Member Function Documentation

5.77.2.1 std::exception TraceErrorEvent::getException ( ) const [inline]

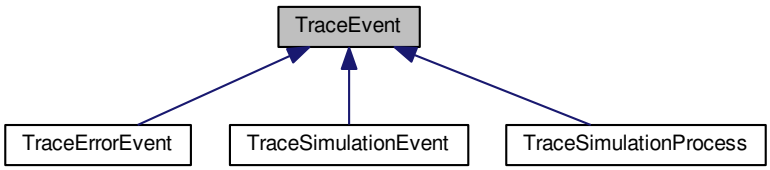
The documentation for this class was generated from the following file:

- TraceManager.h

5.78 TraceEvent Class Reference

`#include <TraceManager.h>`

Inheritance diagram for TraceEvent:



Public Member Functions

- TraceEvent (Util::TraceLevel tracelevel, std::string text)
- Util::TraceLevel getTracelevel () const
- std::string getText () const

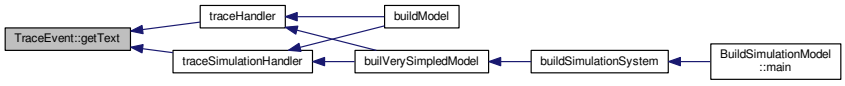
5.78.1 Constructor & Destructor Documentation

5.78.1.1 TraceEvent::TraceEvent (Util::TraceLevel *tracelevel,* std::string *text*) [inline]

5.78.2 Member Function Documentation

5.78.2.1 std::string TraceEvent::getText ( ) const [inline]

Here is the caller graph for this function:



5.78.2.2 Util::TraceLevel TraceEvent::getTracelevel ( ) const [inline]

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

- TraceManager.h

5.79 TraceManager Class Reference

`#include <TraceManager.h>`

Public Member Functions

- TraceManager (Model *model)
- TraceManager (const TraceManager &orig)
- virtual ~TraceManager ()
- void addTraceHandler (traceListener traceListener)
- void addTraceErrorHandler (traceErrorListener traceErrorListener)
- void addTraceReportHandler (traceListener traceReportListener)
- void addTraceSimulationHandler (traceSimulationListener traceSimulationListener)
- void trace (Util::TraceLevel tracelevel, std::string text)
- void traceError (std::exception e, std::string text)
- void traceSimulation (Util::TraceLevel tracelevel, double time, Entity *entity, ModelComponent *component, std::string text)
- void traceReport (Util::TraceLevel tracelevel, std::string text)
- List< std::string > * getErrorMessages () const
- void setTraceLevel (Util::TraceLevel _traceLevel)
- Util::TraceLevel getTraceLevel () const

5.79.1 Detailed Description

The TraceManager is used to trace back model simulation information and track/debug the simulation. It works as the model simulation output (cout) and allows external methods to hook up such output as listeners.

5.79.2 Constructor & Destructor Documentation

5.79.2.1 TraceManager::TraceManager (Model * *model*)

5.79.2.2 TraceManager::TraceManager (const TraceManager & *orig*)

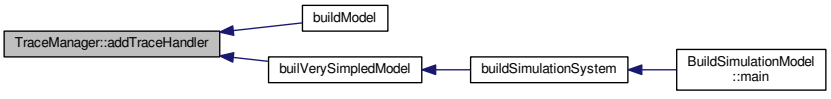
5.79.2.3 TraceManager::~TraceManager ( ) [virtual]

5.79.3 Member Function Documentation

5.79.3.1 void TraceManager::addTraceErrorHandler (traceErrorListener *traceErrorListener*)

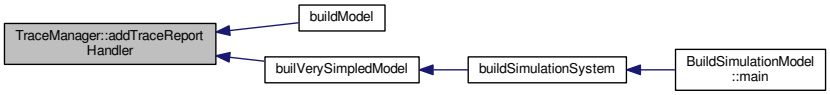
5.79.3.2 void TraceManager::addTraceHandler (traceListener *traceListener*)

Here is the caller graph for this function:



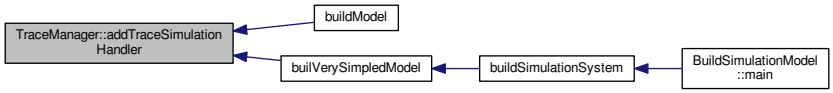
5.79.3.3 void TraceManager::addTraceReportHandler (traceListener *traceReportListener*)

Here is the caller graph for this function:



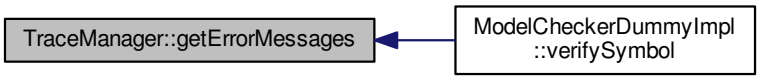
5.79.3.4 void TraceManager::addTraceSimulationHandler (traceSimulationListener *traceSimulationListener*)

Here is the caller graph for this function:



5.79.3.5 List< std::string > * TraceManager::getErrorMessages ( ) const

Here is the caller graph for this function:

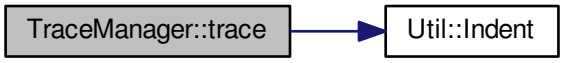


5.79.3.6 Util::TraceLevel TraceManager::getTraceLevel ( ) const

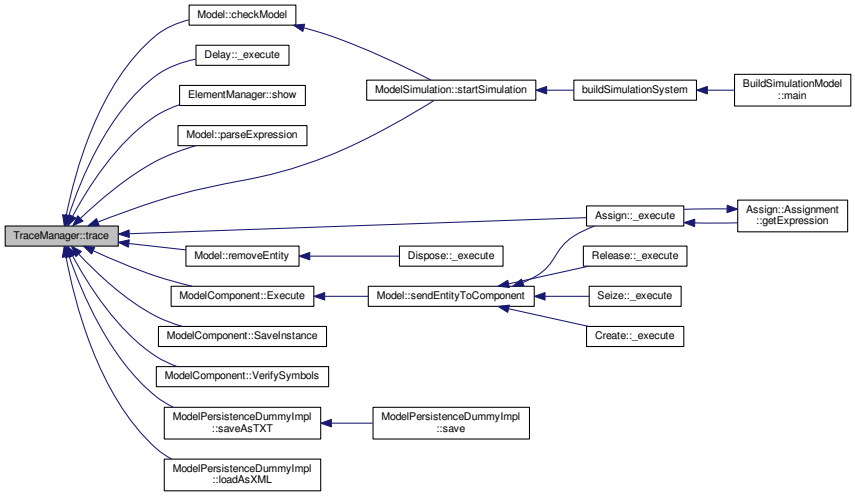
5.79.3.7 void TraceManager::setTraceLevel (Util::TraceLevel *_traceLevel*)

5.79.3.8 void TraceManager::trace (Util::TraceLevel *tracelevel,* std::string *text*)

Here is the call graph for this function:

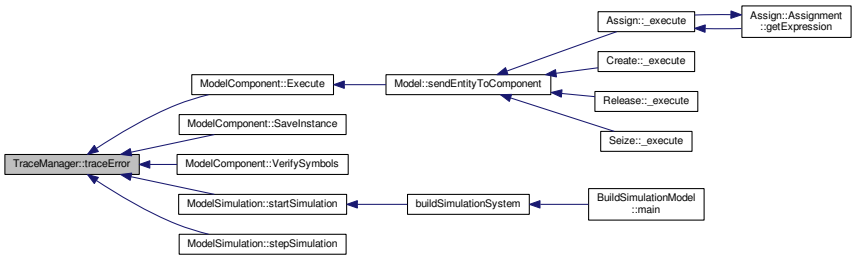


Here is the caller graph for this function:



5.79.3.9 void TraceManager::traceError (std::exception *e,* std::string *text*)

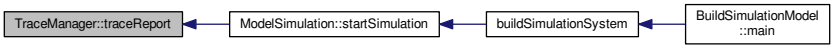
Here is the caller graph for this function:



5.79.3.10 void TraceManager::traceReport (Util::TraceLevel *tracelevel,* std::string *text*)

Here is the call graph for this function:

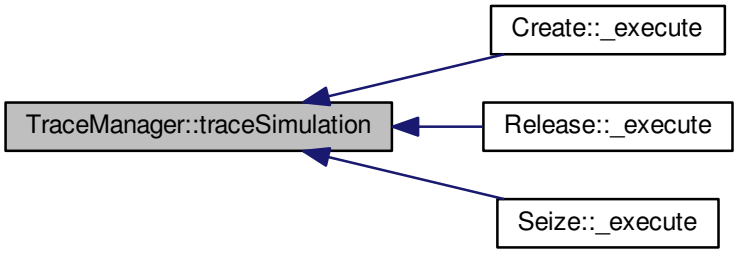
Here is the caller graph for this function:



5.79.3.11 void TraceManager::traceSimulation (Util::TraceLevel *tracelevel,* double *time,* Entity * *entity,* ModelComponent * *component,* std::string *text*)

Here is the call graph for this function:

Here is the caller graph for this function:



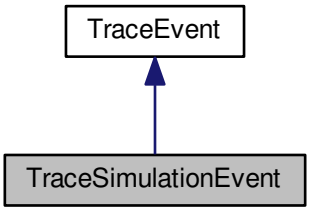
The documentation for this class was generated from the following files:

- TraceManager.h
- TraceManager.cpp

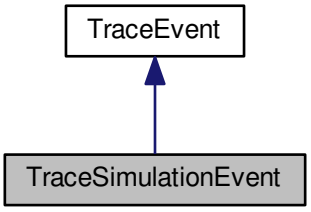
5.80 TraceSimulationEvent Class Reference

`#include <TraceManager.h>`

Inheritance diagram for TraceSimulationEvent:



Collaboration diagram for TraceSimulationEvent:



Public Member Functions

- ModelComponent * getComponent () const
- Entity * getEntity () const
- double getTime () const
- TraceSimulationEvent (Util::TraceLevel tracelevel, double time, Entity *entity, ModelComponent *component, std::string text)

5.80.1 Constructor & Destructor Documentation

5.80.1.1 TraceSimulationEvent::TraceSimulationEvent (Util::TraceLevel *tracelevel,* double *time,* Entity * *entity,* ModelComponent * *component,* std::string *text*) [inline]

5.80.2 Member Function Documentation

5.80.2.1 ModelComponent* TraceSimulationEvent::getComponent ( ) const [inline]

5.80.2.2 Entity* TraceSimulationEvent::getEntity ( ) const [inline]

5.80.2.3 double TraceSimulationEvent::getTime ( ) const [inline]

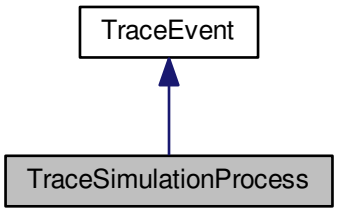
The documentation for this class was generated from the following file:

- TraceManager.h

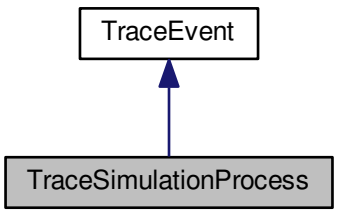
5.81 TraceSimulationProcess Class Reference

`#include <TraceManager.h>`

Inheritance diagram for TraceSimulationProcess:



Collaboration diagram for TraceSimulationProcess:



Public Member Functions

- TraceSimulationProcess (Util::TraceLevel tracelevel, std::string text)

5.81.1 Detailed Description

Events related to simulation "process" (usually process analyser), associated to entire replication or simulation events (begin/end/pause of replication/simulation) TODO: CLASS NOT COMPLETE

5.81.2 Constructor & Destructor Documentation

5.81.2.1 TraceSimulationProcess::TraceSimulationProcess (Util::TraceLevel *tracelevel,* std::string *text*) [inline]

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

- TraceManager.h

5.82 Traits< T > Struct Template Reference

`#include <Traits.h>`

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

- Traits.h

5.83 Traits< Collector_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef CollectorDatafileDefaultImpl1 Implementation

5.83.1 Member Typedef Documentation

5.83.1.1 typedef CollectorDatafileDefaultImpl1 Traits< Collector_if >::Implementation

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

- Traits.h

5.84 Traits< ExperimentDesign_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef ExperimentDesignDummyImpl Implementation

5.84.1 Member Typedef Documentation

5.84.1.1 typedef ExperimentDesignDummyImpl Traits< ExperimentDesign_if >::Implementation

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

- Traits.h

5.85 Traits< Fitter_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef FitterDummyImpl Implementation

5.85.1 Member Typedef Documentation

5.85.1.1 typedef FitterDummyImpl Traits< Fitter_if >::Implementation

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

- Traits.h

5.86 Traits< GenesysApplication_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef BuildSimulationModel Application

5.86.1 Member Typedef Documentation

5.86.1.1 typedef BuildSimulationModel Traits< GenesysApplication_if >::Application

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

- Traits.h

5.87 Traits< HypothesisTester_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef HypothesisTesterDummyImpl Implementation

5.87.1 Member Typedef Documentation

5.87.1.1 typedef HypothesisTesterDummyImpl Traits< HypothesisTester_if >::Implementation

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

- Traits.h

5.88 Traits< Integrator_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef IntegratorDefaultImpl1 Implementation

Static Public Attributes

- static constexpr unsigned int MaxIterations = 1000
- static constexpr double Precision = 1e-9

5.88.1 Member Typedef Documentation

5.88.1.1 typedef IntegratorDefaultImpl1 Traits< Integrator_if >::Implementation

5.88.2 Member Data Documentation

5.88.2.1 constexpr unsigned int Traits< Integrator_if >::MaxIterations = 1000 [static]

5.88.2.2 constexpr double Traits< Integrator_if >::Precision = 1e-9 [static]

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

- Traits.h

5.89 Traits< Model > Struct Template Reference

`#include <Traits.h>`

Static Public Attributes

- static const bool debugged = true
- static const Util::TraceLevel traceLevel = Util::TraceLevel::mostDetailed

5.89.1 Member Data Documentation

5.89.1.1 const bool Traits< Model >::debugged = true [static]

5.89.1.2 const Util::TraceLevel Traits< Model >::traceLevel = Util::TraceLevel::mostDetailed [static]

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

- Traits.h

5.90 Traits< ModelChecker_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef ModelCheckerDummyImpl Implementation

5.90.1 Member Typedef Documentation

5.90.1.1 typedef ModelCheckerDummyImpl Traits< ModelChecker_if >::Implementation

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

- Traits.h

5.91 Traits< ModelComponent > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef CollectorDatafileDefaultImpl1 CollectorImplementation
- typedef CollectorDefaultImpl1 StatisticsCollectorImplementation

5.91.1 Member Typedef Documentation

5.91.1.1 typedef CollectorDatafileDefaultImpl1 Traits< ModelComponent >::CollectorImplementation

5.91.1.2 typedef CollectorDefaultImpl1 Traits< ModelComponent >::StatisticsCollectorImplementation

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

- Traits.h

5.92 Traits< ModelPersistence_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef ModelPersistenceDummyImpl Implementation

5.92.1 Member Typedef Documentation

5.92.1.1 typedef ModelPersistenceDummyImpl Traits< ModelPersistence_if >::Implementation

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

- Traits.h

5.93 Traits< Parser_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef ParserFlexBisonImpl Implementation

5.93.1 Member Typedef Documentation

5.93.1.1 typedef ParserFlexBisonImpl Traits< Parser_if >::Implementation

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

- Traits.h

5.94 Traits< ProcessAnalyser_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef ProcessAnalyserDummyImpl Implementation

5.94.1 Member Typedef Documentation

5.94.1.1 typedef ProcessAnalyserDummyImpl Traits< ProcessAnalyser_if >::Implementation

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

- Traits.h

5.95 Traits< Sampler_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef SamplerDefaultImpl1 Implementation
- typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Parameters

5.95.1 Member Typedef Documentation

5.95.1.1 typedef SamplerDefaultImpl1 Traits< Sampler_if >::Implementation

5.95.1.2 typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Traits< Sampler_if >::Parameters

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

- Traits.h

5.96 Traits< Statistics_if > Struct Template Reference

`#include <Traits.h>`

Public Types

- typedef StatisticsDefaultImpl1 Implementation
- typedef CollectorDefaultImpl1 CollectorImplementation

5.96.1 Member Typedef Documentation

5.96.1.1 typedef CollectorDefaultImpl1 Traits< Statistics_if >::CollectorImplementation

5.96.1.2 typedef StatisticsDefaultImpl1 Traits< Statistics_if >::Implementation

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

- Traits.h

5.97 Util Class Reference

`#include <Util.h>`

Public Types

- enum TimeUnit : int {
  TimeUnit::picosecond = 1, TimeUnit::nanosecond = 2, TimeUnit::microsecond = 3, TimeUnit::milisecond = 4,
  TimeUnit::second = 5, TimeUnit::minute = 6, TimeUnit::hour = 7, TimeUnit::day = 8,
  TimeUnit::week = 9 }
- enum TraceLevel : int {
  TraceLevel::noTraces = 0, TraceLevel::errors = 10, TraceLevel::report = 20, TraceLevel::simulation = 30,
  TraceLevel::transferOnly = 40, TraceLevel::blockArrival = 50, TraceLevel::blockInternal = 60, TraceLevel↩
  ::mostDetailed = 70 }
- typedef unsigned long identitifcation
- typedef unsigned int rank

Static Public Member Functions

- static void IncIndent ()
- static void DecIndent ()
- static std::string Indent ()
- static Util::identitifcation GenerateNewId ()
- static double TimeUnitConvert (Util::TimeUnit timeUnit1, Util::TimeUnit timeUnit2)
- static Util::identitifcation GenerateNewIdOfType (std::string objtyp)
- template<class T >
  static std::string TypeOf ()
- template<class T >
  static Util::identitifcation GenerateNewIdOfType ()

5.97.1 Member Typedef Documentation

5.97.1.1 typedef unsigned long Util::identitifcation

5.97.1.2 typedef unsigned int Util::rank

5.97.2 Member Enumeration Documentation

5.97.2.1 enum Util::TimeUnit : int [strong]

Enumerator

picosecond

nanosecond

microsecond

milisecond

second

minute

hour

day

week

5.97.2.2 enum Util::TraceLevel : int [strong]

Enumerator

noTraces

errors

report

simulation

transferOnly

blockArrival

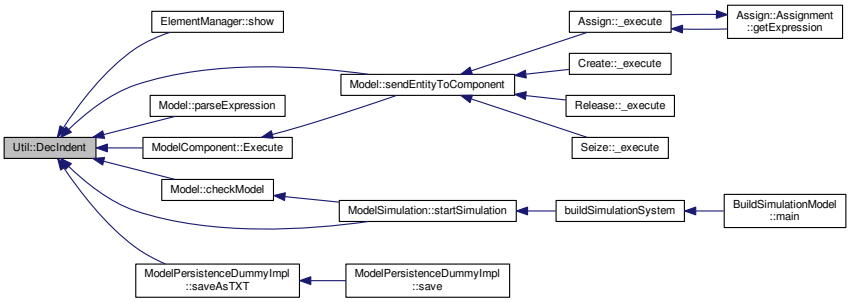
blockInternal

mostDetailed

5.97.3 Member Function Documentation

5.97.3.1 void Util::DecIndent ( ) [static]

Here is the caller graph for this function:



5.97.3.2 Util::identitifcation Util::GenerateNewId ( ) [static]

5.97.3.3 Util::identitifcation Util::GenerateNewIdOfType (std::string *objtyp*) [static]

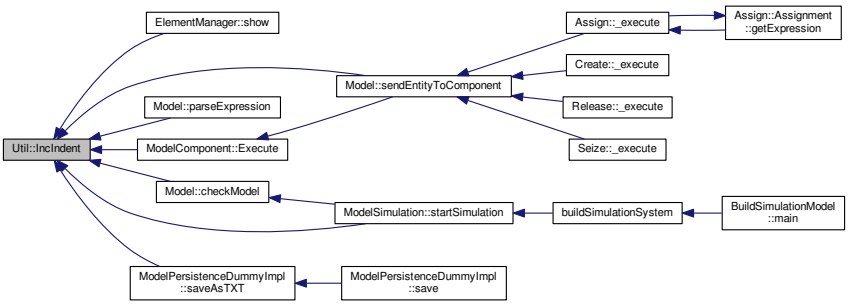
5.97.3.4 template<class T > static Util::identitifcation Util::GenerateNewIdOfType ( ) [inline], [static]

Every component or element has a unique ID for its class, but not unique for other classes. IDs are generated sequentially for each class.

Here is the caller graph for this function:

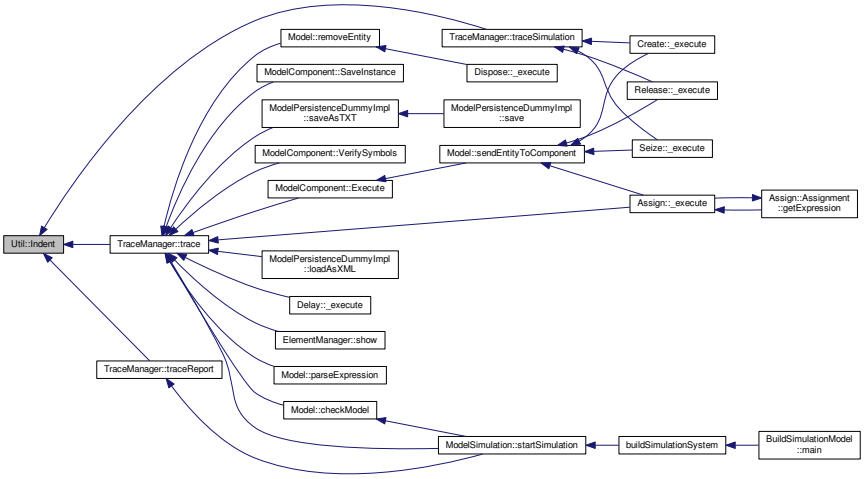
5.97.3.5 void Util::IncIndent ( ) [static]

Here is the caller graph for this function:



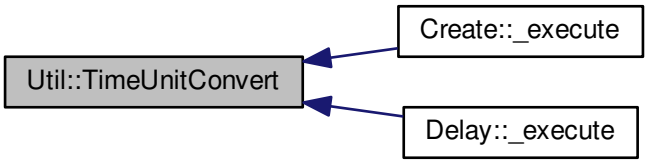
5.97.3.6 std::string Util::Indent ( ) [static]

Here is the caller graph for this function:



5.97.3.7 double Util::TimeUnitConvert (Util::TimeUnit *timeUnit1*, Util::TimeUnit *timeUnit2*) [static]

Here is the caller graph for this function:



5.97.3.8 template<class T > static std::string Util::TypeOf ( ) [inline], [static]

Return the name of the class used as T.

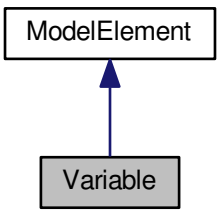
The documentation for this class was generated from the following files:

- Util.h
- Util.cpp

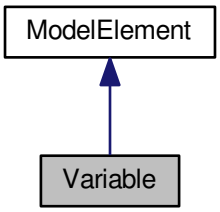
5.98 Variable Class Reference

`#include <Variable.h>`

Inheritance diagram for Variable:



Collaboration diagram for Variable:



Public Member Functions

- Variable ()
- Variable (std::string name)
- Variable (const Variable &orig)
- virtual ~Variable ()
- virtual std::string show ()
- double getValue ()
- double getValue (std::string index)
- void setValue (double value)
- void setValue (std::string index, double value)

Protected Member Functions

- virtual void _loadInstance (std::list< std::string > words)
- virtual std::list< std::string > * _saveInstance ()
- virtual bool _verifySymbols (std::string *errorMessage)

Additional Inherited Members

5.98.1 Constructor & Destructor Documentation

5.98.1.1 Variable::Variable ( )

5.98.1.2 Variable::Variable (std::string *name*)

5.98.1.3 Variable::Variable (const Variable & *orig*)

5.98.1.4 Variable::~Variable ( ) [virtual]

5.98.2 Member Function Documentation

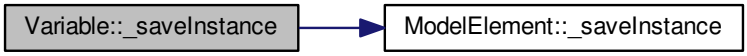
5.98.2.1 void Variable::_loadInstance (std::list< std::string > *words*) [protected], [virtual]

Implements ModelElement.

5.98.2.2 std::list< std::string > * Variable::_saveInstance ( ) [protected], [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



5.98.2.3 bool Variable::_verifySymbols (std::string * *errorMessage*) [protected], [virtual]

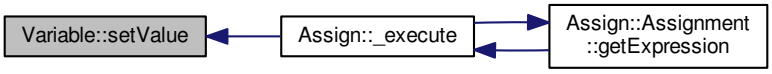
Implements ModelElement.

5.98.2.4 double Variable::getValue ( )

5.98.2.5 double Variable::getValue (std::string *index*)

5.98.2.6 void Variable::setValue (double *value*)

Here is the caller graph for this function:

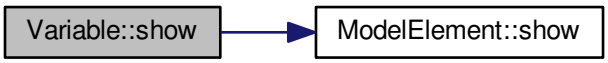


5.98.2.7 void Variable::setValue (std::string *index,* double *value*)

5.98.2.8 std::string Variable::show ( ) [virtual]

Reimplemented from ModelElement.

Here is the call graph for this function:



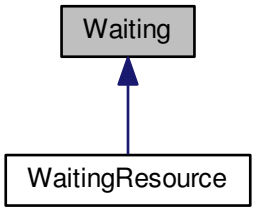
The documentation for this class was generated from the following files:

- Variable.h
- Variable.cpp

5.99 Waiting Class Reference

`#include <Waiting.h>`

Inheritance diagram for Waiting:



Public Member Functions

- Waiting (Entity *entity, ModelComponent *component, double timeStartedWaiting)
- Waiting (const Waiting &orig)
- virtual ~Waiting ()
- virtual std::string show ()
- double getTimeStartedWaiting () const
- ModelComponent * getComponent () const
- Entity * getEntity () const

5.99.1 Constructor & Destructor Documentation

5.99.1.1 Waiting::Waiting (Entity * *entity,* ModelComponent * *component,* double *timeStartedWaiting*)

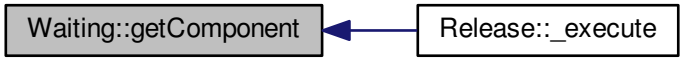
5.99.1.2 Waiting::Waiting (const Waiting & *orig*)

5.99.1.3 Waiting::~Waiting ( ) [virtual]

5.99.2 Member Function Documentation

5.99.2.1 ModelComponent * Waiting::getComponent ( ) const

Here is the caller graph for this function:

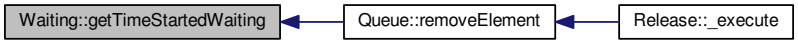


5.99.2.2 Entity * Waiting::getEntity ( ) const

Here is the caller graph for this function:

5.99.2.3 double Waiting::getTimeStartedWaiting ( ) const

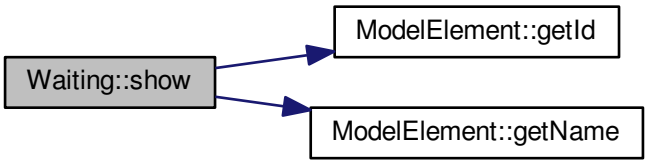
Here is the caller graph for this function:



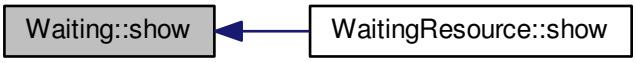
5.99.2.4 std::string Waiting::show ( ) [virtual]

Reimplemented in WaitingResource.

Here is the call graph for this function:



Here is the caller graph for this function:



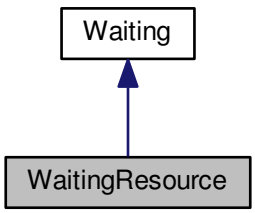
The documentation for this class was generated from the following files:

- Waiting.h
- Waiting.cpp

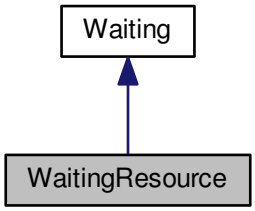
5.100 WaitingResource Class Reference

`#include <WaitingResource.h>`

Inheritance diagram for WaitingResource:



Collaboration diagram for WaitingResource:



Public Member Functions

- WaitingResource (Entity *entity, ModelComponent *component, double timeStartedWaiting, unsigned int quantity)
- WaitingResource (const WaitingResource &orig)
- virtual ~WaitingResource ()
- virtual std::string show ()
- unsigned int getQuantity () const

5.100.1 Constructor & Destructor Documentation

5.100.1.1 WaitingResource::WaitingResource (Entity * *entity,* ModelComponent * *component,* double *timeStartedWaiting,* unsigned int *quantity*)

5.100.1.2 WaitingResource::WaitingResource (const WaitingResource & *orig*)

5.100.1.3 WaitingResource::~WaitingResource ( ) [virtual]

5.100.2 Member Function Documentation

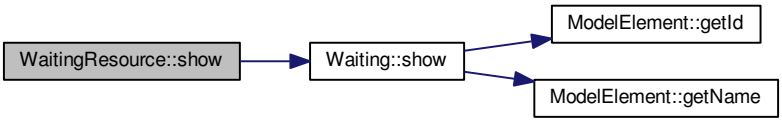
5.100.2.1 unsigned int WaitingResource::getQuantity ( ) const

Here is the caller graph for this function:

5.100.2.2 std::string WaitingResource::show ( ) [virtual]

Reimplemented from Waiting.

Here is the call graph for this function:



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

- WaitingResource.h
- WaitingResource.cpp

Chapter 6

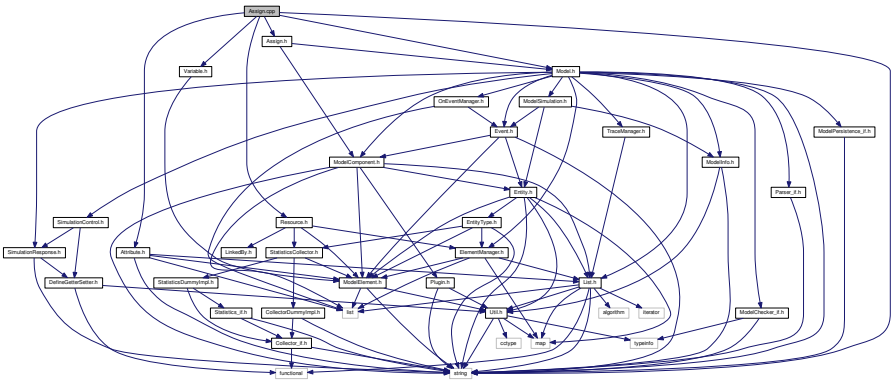
File Documentation

6.1 .dep.inc File Reference

6.2 Assign.cpp File Reference

```
#include <string>
#include "Model.h"
#include "Assign.h"
#include "Variable.h"
#include "Attribute.h"
#include "Resource.h"
```

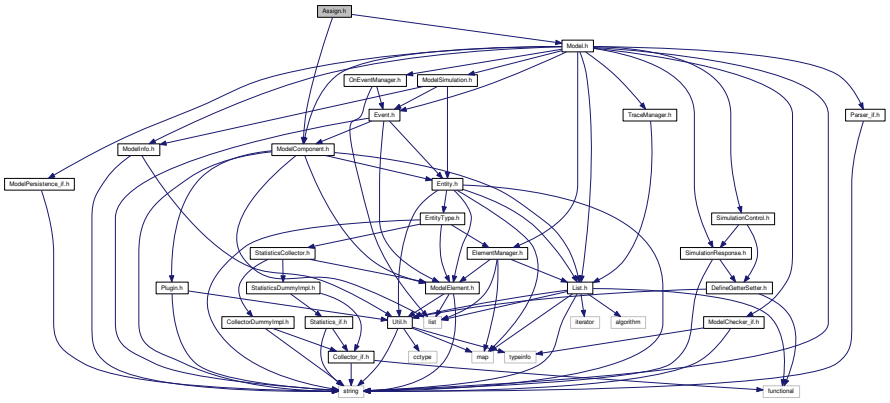
Include dependency graph for Assign.cpp:



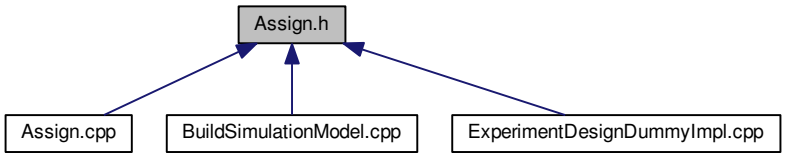
6.3 Assign.h File Reference

```
#include "ModelComponent.h"
#include "Model.h"
```

Include dependency graph for Assign.h:



This graph shows which files directly or indirectly include this file:



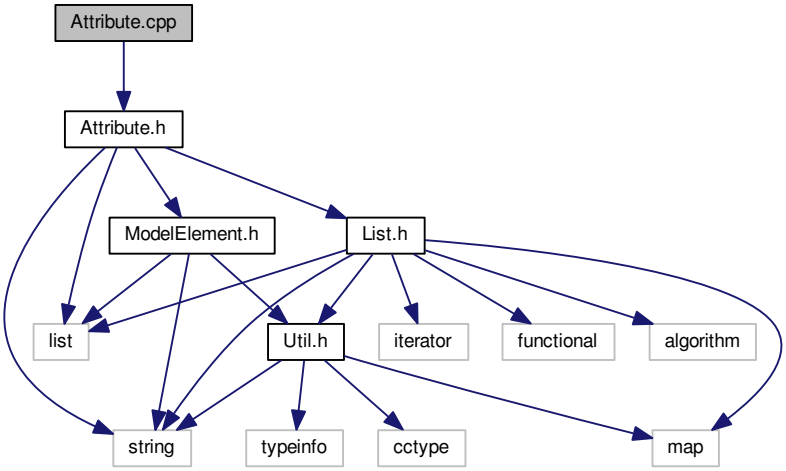
Classes

- class Assign
- class Assign::Assignment

6.4 Attribute.cpp File Reference

```
#include "Attribute.h"
```

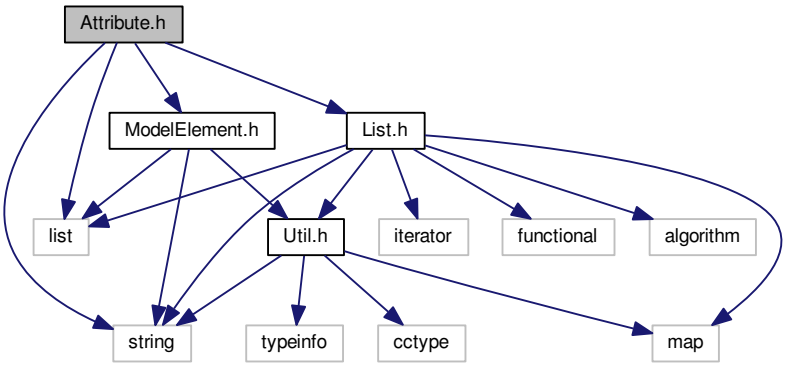
Include dependency graph for Attribute.cpp:



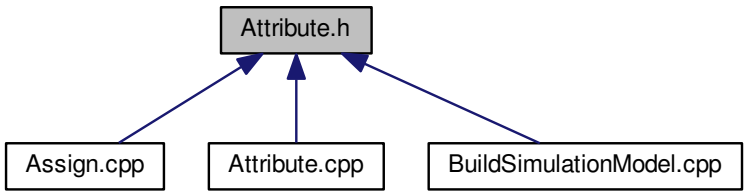
6.5 Attribute.h File Reference

```
#include <string>
#include <list>
#include "List.h"
#include "ModelElement.h"
```

Include dependency graph for Attribute.h:



This graph shows which files directly or indirectly include this file:



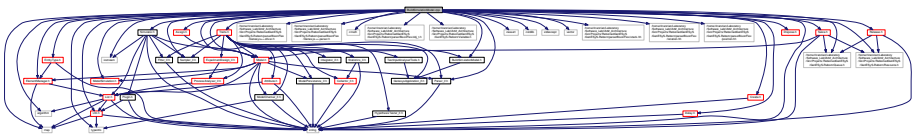
Classes

- class Attribute

6.6 BuildSimulationModel.cpp File Reference

```
#include "BuildSimulationModel.h"
#include "Simulator.h"
#include "Traits.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "ModelSimulation.h"
#include "ElementManager.h"
#include "EntityType.h"
#include "Attribute.h"
```

Include dependency graph for BuildSimulationModel.cpp:



Functions

- void traceHandler (TraceEvent e)
- void traceSimulationHandler (TraceSimulationEvent e)
- void onSimulationStartHandler (SimulationEvent *re)
- void onReplicationStartHandler (SimulationEvent *re)
- void onProcessEventHandler (SimulationEvent *re)
- void onReplicationEndHandler (SimulationEvent *re)
- void onEntityRemoveHandler (SimulationEvent *re)
- void buildModel (Model *model)
- void builVerySimpledModel (Model *model)
- void buildSimulationSystem ()

6.6.1 Function Documentation

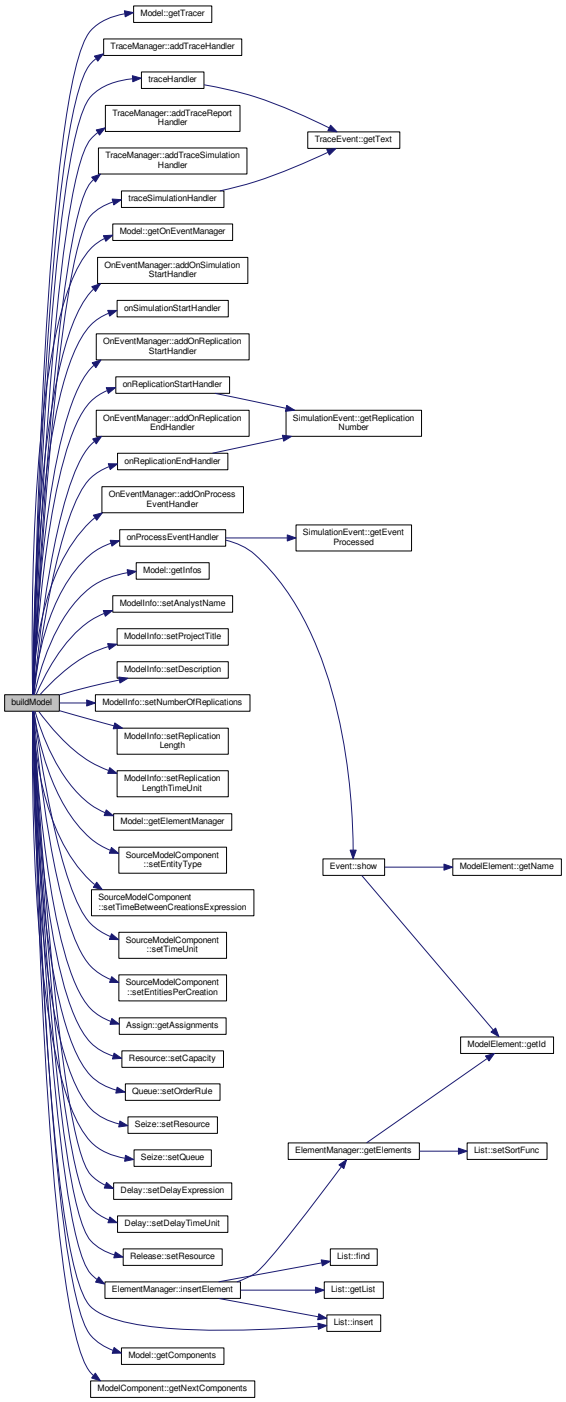
6.6.1.1 void buildModel (Model * *model*)

This function shows an example of how to create a simulation model. It creates some handlers for tracing (debug) and for events, set model infos and than creates the model itself. The model is a composition of components (and elements that they use), connected to form a process/fluxogram

Parameters

model	- The instance returned that will contains the built model

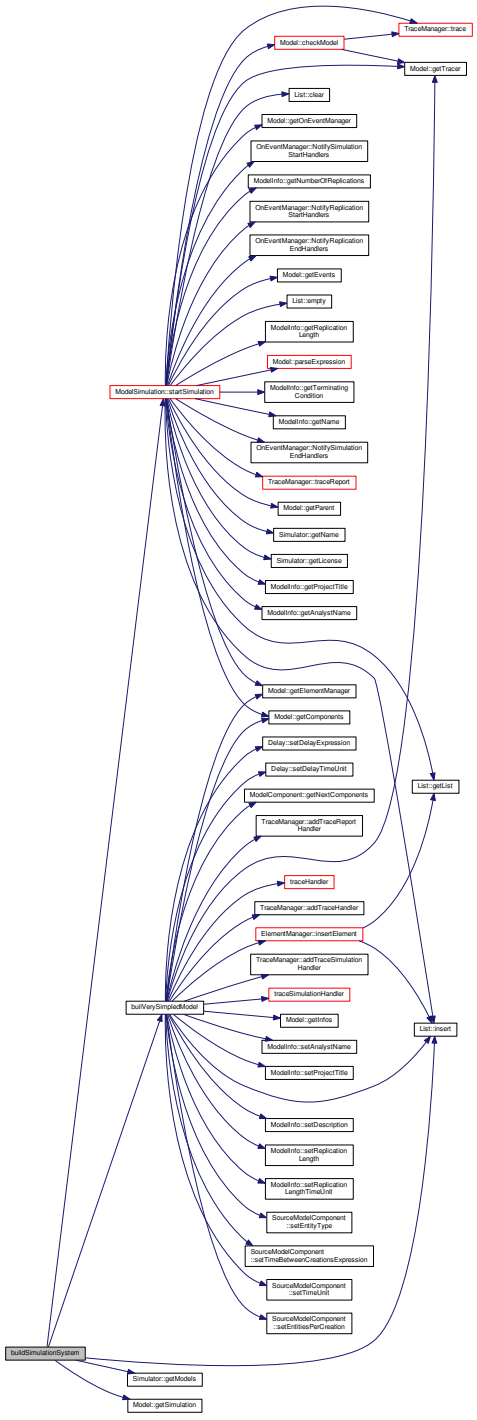
Here is the call graph for this function:



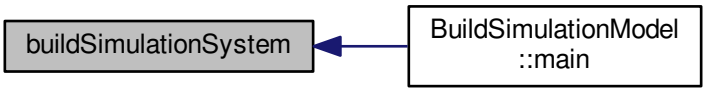
6.6.1.2 void buildSimulationSystem ( )

This is the main function of the BuildSimulationModel application. It instanciates the simulator, builds a simulation model and then simulate that model.

Here is the call graph for this function:

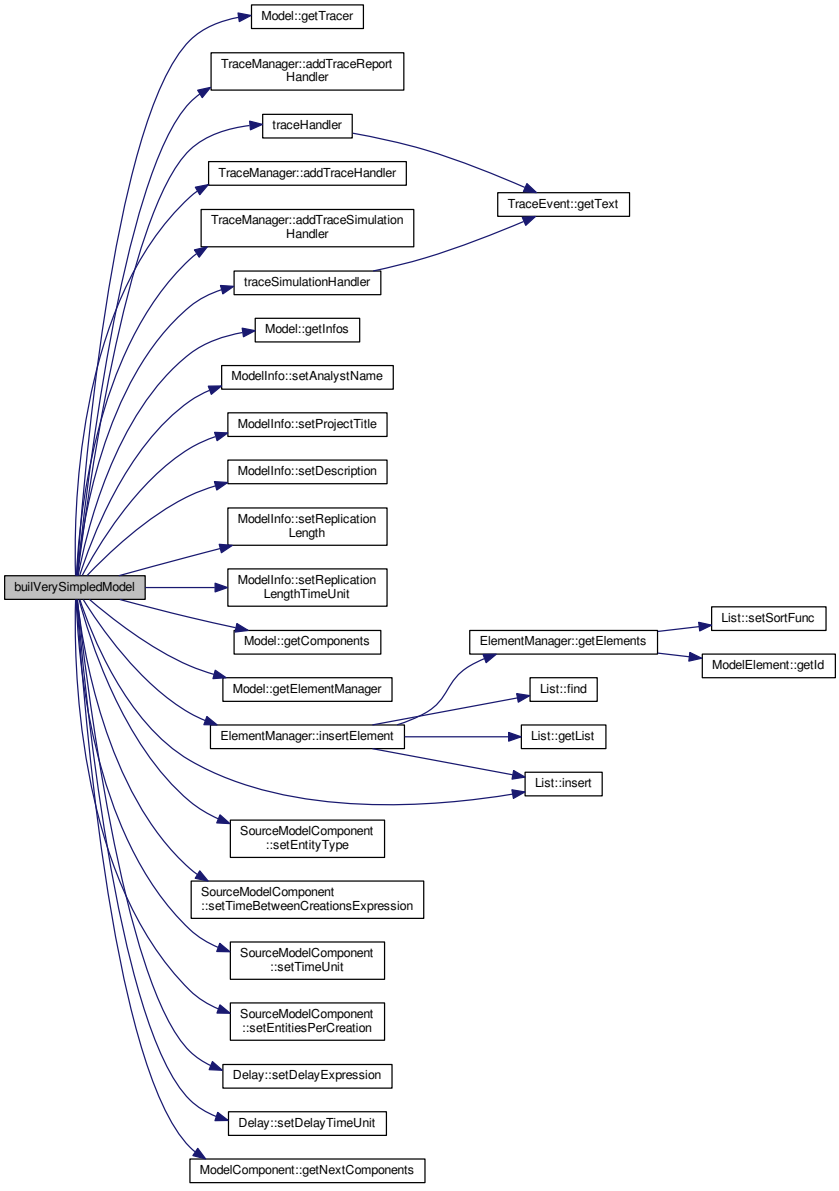


Here is the caller graph for this function:

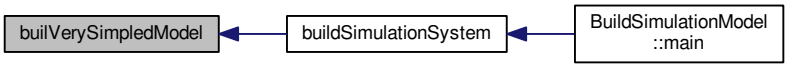


6.6.1.3 void builVerySimpledModel (Model * *model*)

Here is the call graph for this function:

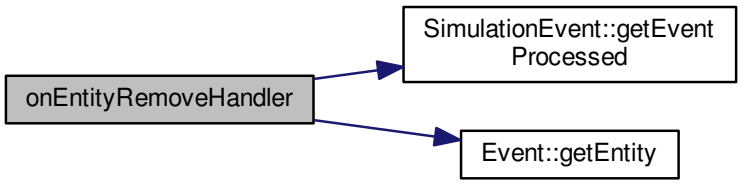


Here is the caller graph for this function:



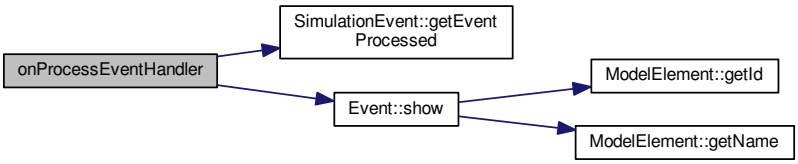
6.6.1.4 void onEntityRemoveHandler (SimulationEvent * re)

Here is the call graph for this function:



6.6.1.5 void onProcessEventHandler (SimulationEvent * re)

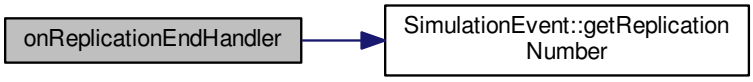
Here is the call graph for this function:



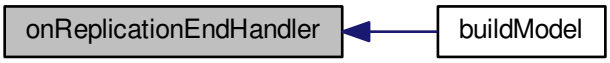
Here is the caller graph for this function:

6.6.1.6 void onReplicationEndHandler (SimulationEvent * re)

Here is the call graph for this function:

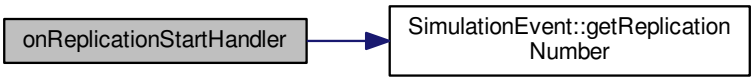


Here is the caller graph for this function:

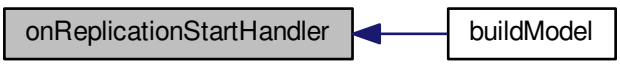


6.6.1.7 void onReplicationStartHandler (SimulationEvent * re)

Here is the call graph for this function:

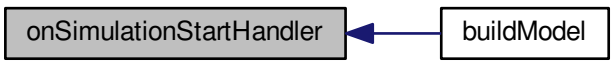


Here is the caller graph for this function:



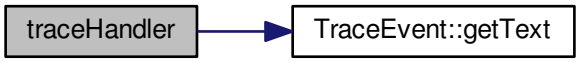
6.6.1.8 void onSimulationStartHandler (SimulationEvent * *re*)

Here is the caller graph for this function:

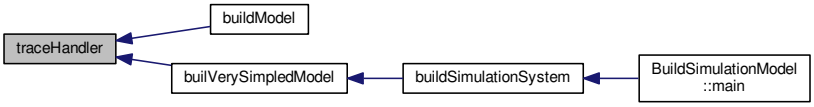


6.6.1.9 void traceHandler (TraceEvent *e*)

Here is the call graph for this function:



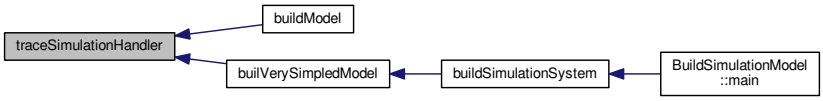
Here is the caller graph for this function:



6.6.1.10 void traceSimulationHandler (TraceSimulationEvent *e*)

Here is the call graph for this function:

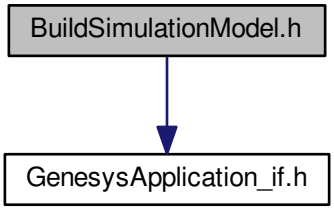
Here is the caller graph for this function:



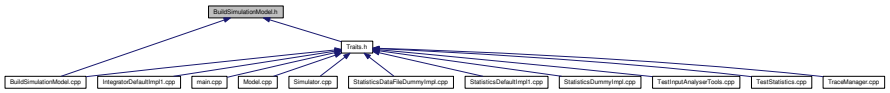
6.7 BuildSimulationModel.h File Reference

```
#include "GenesysApplication_if.h"
```

Include dependency graph for BuildSimulationModel.h:



This graph shows which files directly or indirectly include this file:



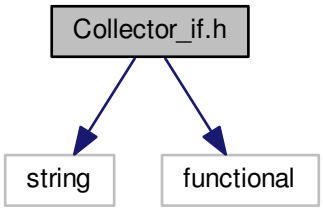
Classes

- class BuildSimulationModel

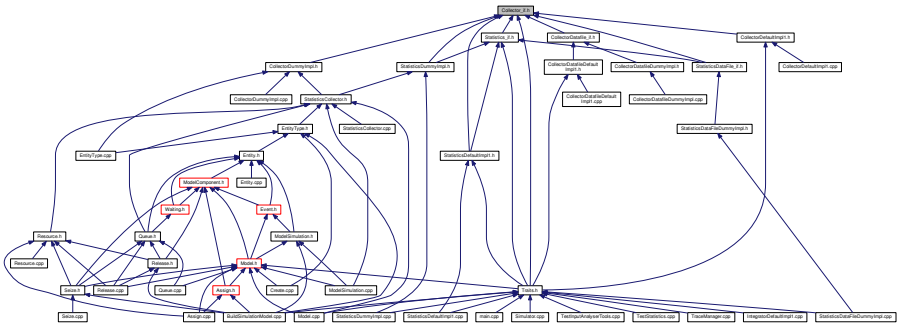
6.8 Collector_if.h File Reference

```
#include <string>
#include <functional>
```

Include dependency graph for Collector_if.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Collector_if

Typedefs

- typedef std::function< void(double) > CollectorAddValueHandler
- typedef std::function< void() > CollectorClearHandler

Functions

- template<typename Class >
  CollectorAddValueHandler SetCollectorAddValueHandler (void(Class::*function)(double), Class *object)
- template<typename Class >
  CollectorClearHandler SetCollectorClearHandler (void(Class::*function)(), Class *object)

6.8.1 Typedef Documentation

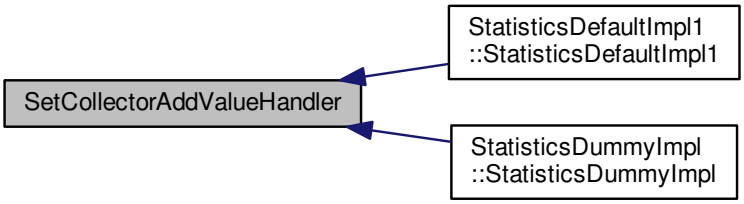
6.8.1.1 typedef std::function<void(double) > CollectorAddValueHandler

6.8.1.2 typedef std::function<void() > CollectorClearHandler

6.8.2 Function Documentation

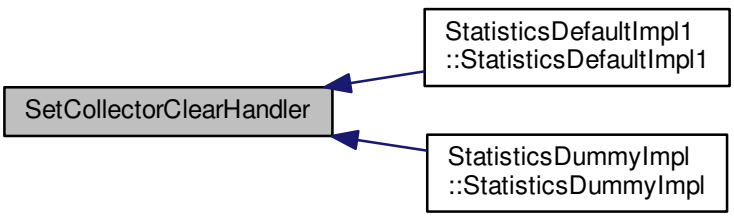
6.8.2.1 template<typename Class > CollectorAddValueHandler SetCollectorAddValueHandler (void(Class::*)(double) *function,* Class * *object*)

Here is the caller graph for this function:



6.8.2.2 template<typename Class > CollectorClearHandler SetCollectorClearHandler (void(Class::*)() *function,* Class * *object*)

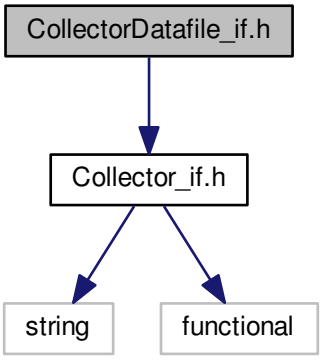
Here is the caller graph for this function:



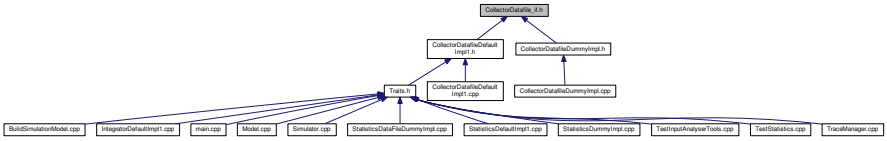
6.9 CollectorDatafile_if.h File Reference

```
#include "Collector_if.h"
```

Include dependency graph for CollectorDatafile_if.h:



This graph shows which files directly or indirectly include this file:



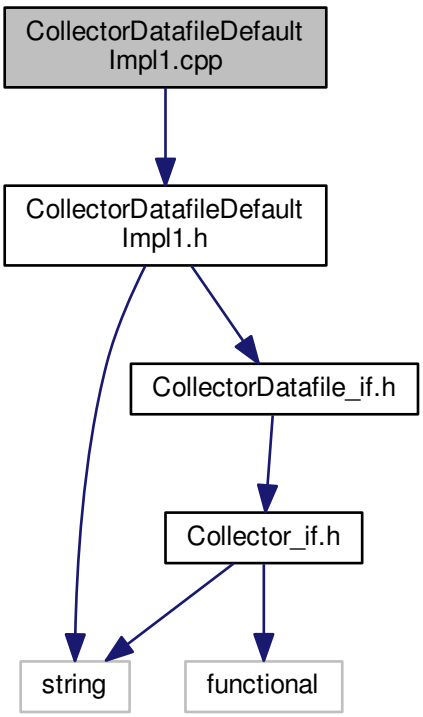
Classes

- class CollectorDatafile_if

6.10 CollectorDatafileDefaultImpl1.cpp File Reference

```
#include "CollectorDatafileDefaultImpl1.h"
```

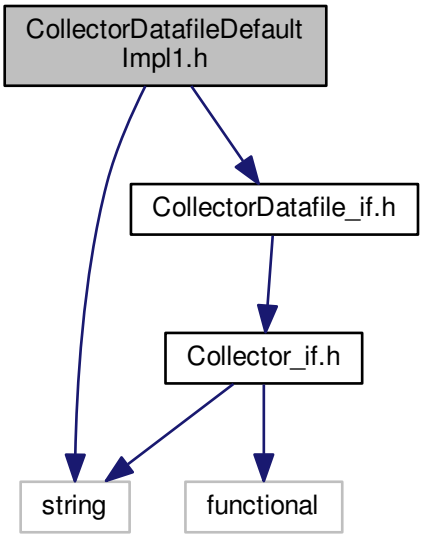
Include dependency graph for CollectorDatafileDefaultImpl1.cpp:



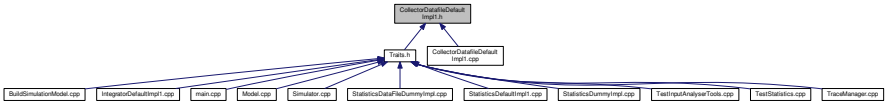
6.11 CollectorDatafileDefaultImpl1.h File Reference

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

Include dependency graph for CollectorDatafileDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



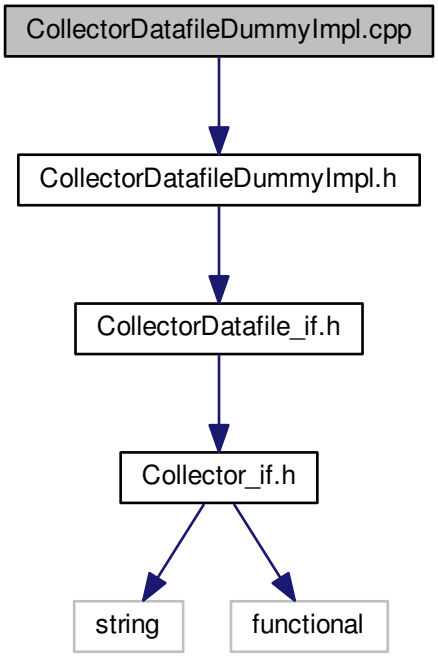
Classes

- class CollectorDatafileDefaultImpl1

6.12 CollectorDatafileDummyImpl.cpp File Reference

```
#include "CollectorDatafileDummyImpl.h"
```

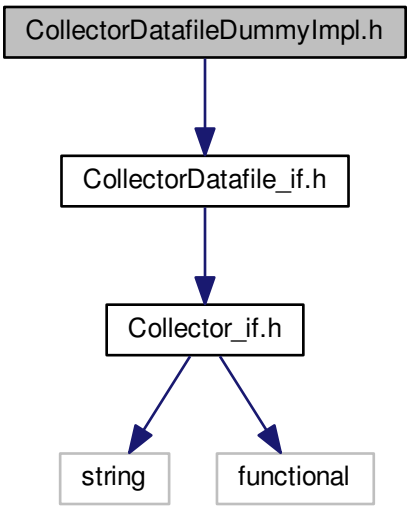
Include dependency graph for CollectorDatafileDummyImpl.cpp:



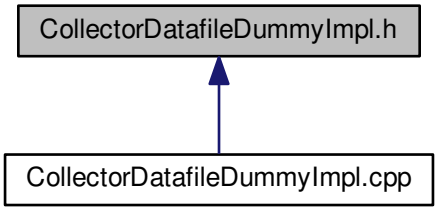
6.13 CollectorDatafileDummyImpl.h File Reference

```
#include "CollectorDatafile_if.h"
```

Include dependency graph for CollectorDatafileDummyImpl.h:



This graph shows which files directly or indirectly include this file:



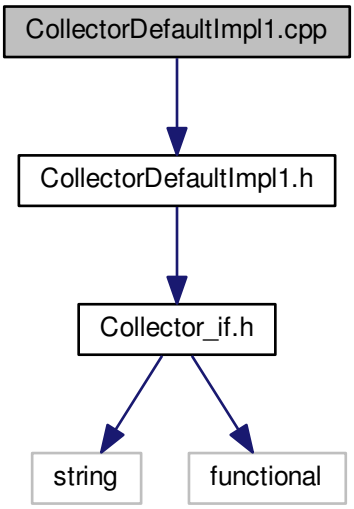
Classes

- class CollectorDatafileDummyImpl

6.14 CollectorDefaultImpl1.cpp File Reference

```
#include "CollectorDefaultImpl1.h"
```

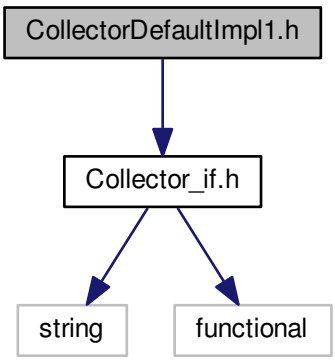
Include dependency graph for CollectorDefaultImpl1.cpp:



6.15 CollectorDefaultImpl1.h File Reference

```
#include "Collector_if.h"
```

Include dependency graph for CollectorDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:

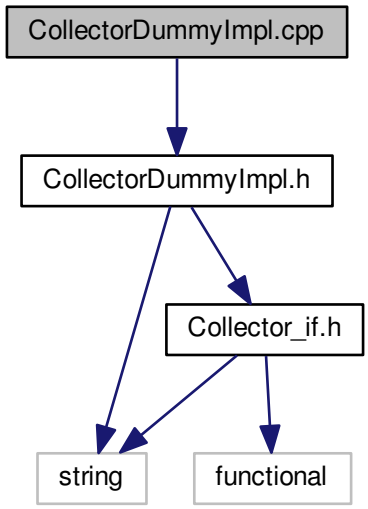
Classes

- class CollectorDefaultImpl1

6.16 CollectorDummyImpl.cpp File Reference

```
#include "CollectorDummyImpl.h"
```

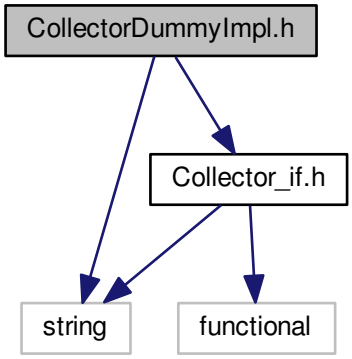
Include dependency graph for CollectorDummyImpl.cpp:



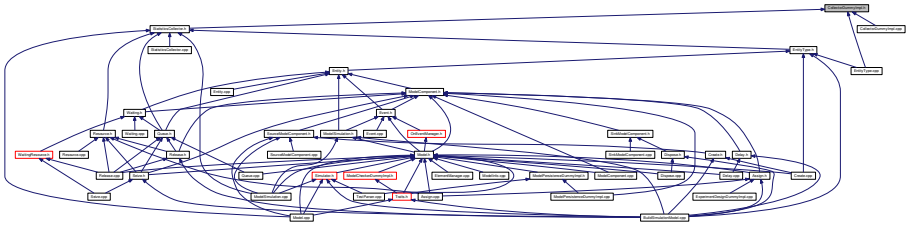
6.17 CollectorDummyImpl.h File Reference

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

Include dependency graph for CollectorDummyImpl.h:



This graph shows which files directly or indirectly include this file:



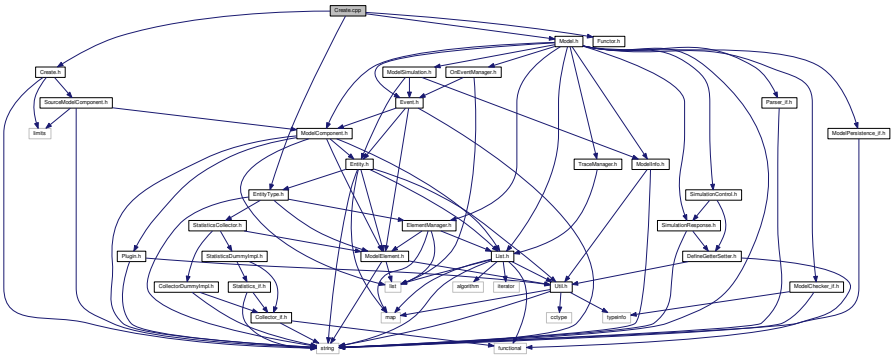
Classes

- class CollectorDummyImpl

6.18 Create.cpp File Reference

```
#include "Create.h"
#include "Model.h"
#include "EntityType.h"
#include "Functor.h"
```

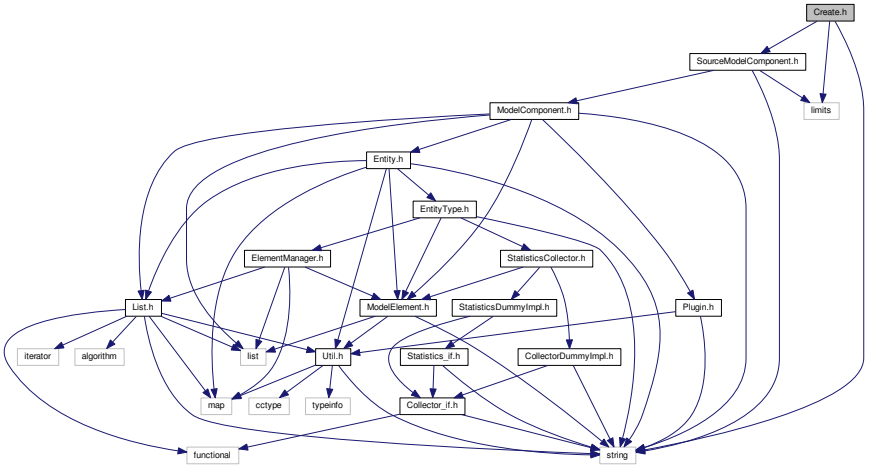
Include dependency graph for Create.cpp:



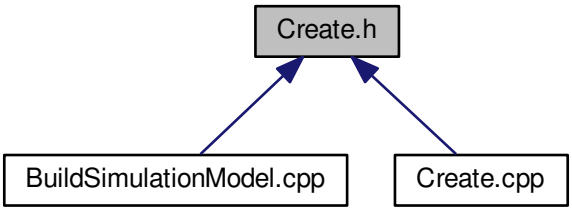
6.19 Create.h File Reference

```
#include <string>
#include <limits>
#include "SourceModelComponent.h"
```

Include dependency graph for Create.h:



This graph shows which files directly or indirectly include this file:



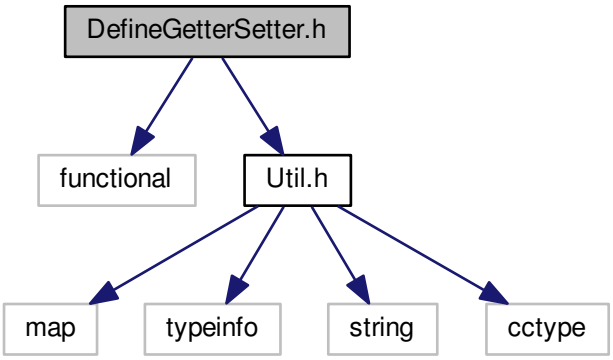
Classes

- class Create

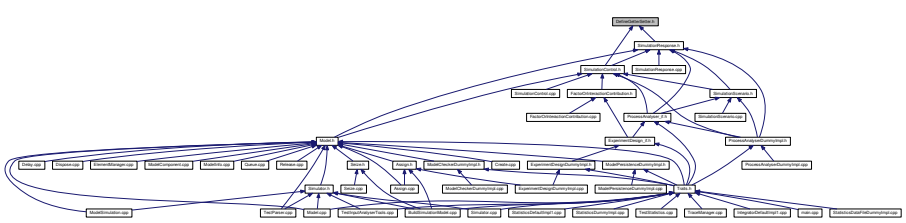
6.20 DefineGetterSetter.h File Reference

```
#include <functional>
#include "Util.h"
```

Include dependency graph for DefineGetterSetter.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef std::function< double() > GetterMember
- typedef std::function< void(double) > SetterMember

Functions

- template<typename Class >
  GetterMember DefineGetterMember (Class *object, double(Class::*function)())
- template<typename Class >
  SetterMember DefineSetterMember (Class *object, void(Class::*function)(double))
- template<typename Class >
  GetterMember DefineGetterMember (Class *object, unsigned int(Class::*function)() const)
- template<typename Class >
  SetterMember DefineSetterMember (Class *object, void(Class::*function)(unsigned int))
- template<typename Class >
  GetterMember DefineGetterMember (Class *object, bool(Class::*function)() const)
- template<typename Class >
  SetterMember DefineSetterMember (Class *object, void(Class::*function)(bool))
- template<typename Class >
  GetterMember DefineGetterMember (Class *object, std::string(Class::*function)() const)
- template<typename Class >
  SetterMember DefineSetterMember (Class *object, void(Class::*function)(std::string))
- template<typename Class >
  GetterMember DefineGetterMember (Class *object, Util::TimeUnit(Class::*function)() const)
- template<typename Class >
  SetterMember DefineSetterMember (Class *object, void(Class::*function)(Util::TimeUnit))

6.20.1 Typedef Documentation

6.20.1.1 typedef std::function<double() > GetterMember

6.20.1.2 typedef std::function<void(double) > SetterMember

6.20.2 Function Documentation

6.20.2.1 template<typename Class > GetterMember DefineGetterMember (Class * *object*, double(Class::*)() *function*)

6.20.2.2 template<typename Class > GetterMember DefineGetterMember (Class * *object*, unsigned int(Class::*)() const *function*)

6.20.2.3 template<typename Class > GetterMember DefineGetterMember (Class * *object*, bool(Class::*)() const *function*)

6.20.2.4 template<typename Class > GetterMember DefineGetterMember (Class * *object*, std::string(Class::*)() const *function*)

6.20.2.5 template<typename Class > GetterMember DefineGetterMember (Class * *object*, Util::TimeUnit(Class::*)() const *function*)

6.20.2.6 template<typename Class > SetterMember DefineSetterMember (Class * *object,* void(Class::*)(double) *function*)

6.20.2.7 template<typename Class > SetterMember DefineSetterMember (Class * *object,* void(Class::*)(unsigned int) *function*)

6.20.2.8 template<typename Class > SetterMember DefineSetterMember (Class * *object,* void(Class::*)(bool) *function*)

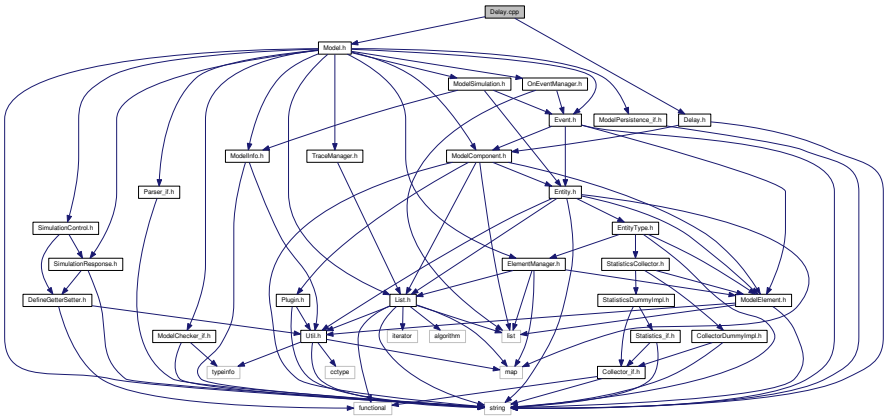
6.20.2.9 template<typename Class > SetterMember DefineSetterMember (Class * *object,* void(Class::*)(std::string) *function*)

6.20.2.10 template<typename Class > SetterMember DefineSetterMember (Class * *object,* void(Class::*)(Util::TimeUnit) *function*)

6.21 Delay.cpp File Reference

```
#include "Delay.h"
#include "Model.h"
```

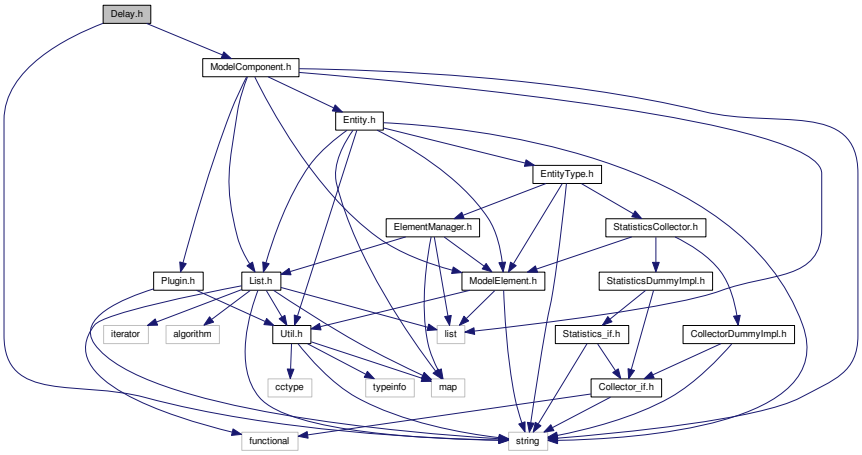
Include dependency graph for Delay.cpp:



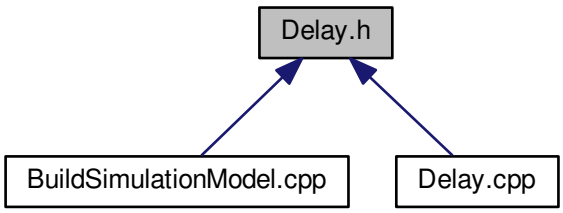
6.22 Delay.h File Reference

```
#include <string>
#include "ModelComponent.h"
```

Include dependency graph for Delay.h:



This graph shows which files directly or indirectly include this file:



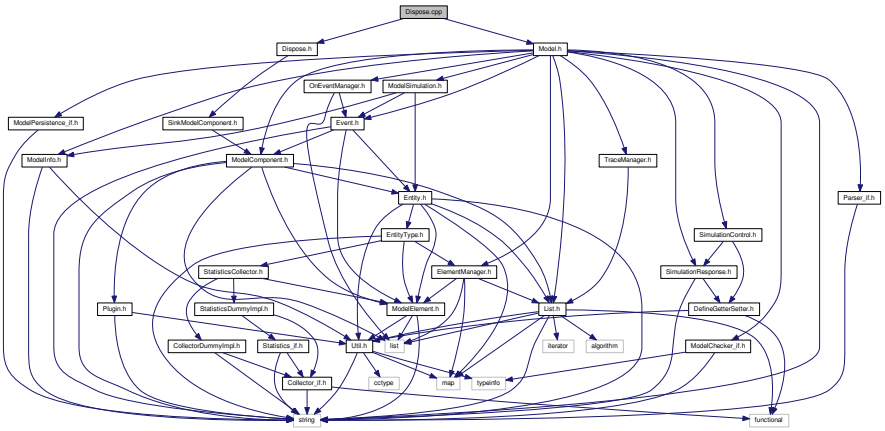
Classes

- class Delay

6.23 Dispose.cpp File Reference

```
#include "Dispose.h"
#include "Model.h"
```

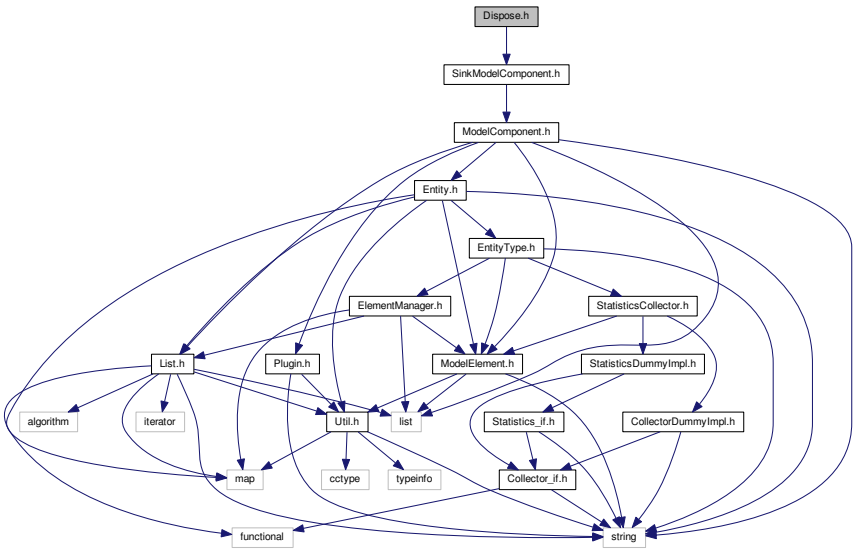
Include dependency graph for Dispose.cpp:



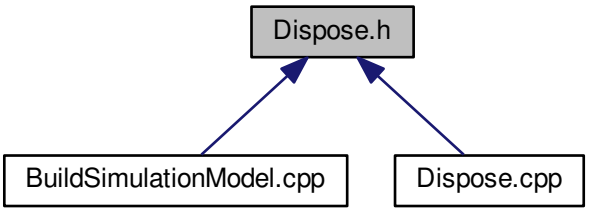
6.24 Dispose.h File Reference

```
#include "SinkModelComponent.h"
```

Include dependency graph for Dispose.h:



This graph shows which files directly or indirectly include this file:



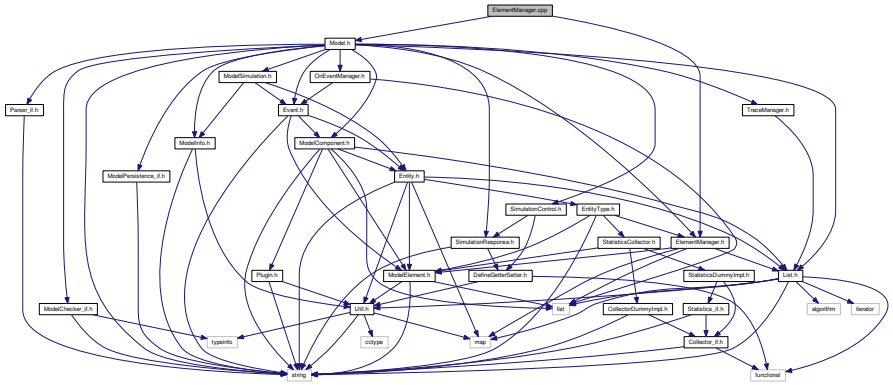
Classes

- class Dispose

6.25 ElementManager.cpp File Reference

```
#include "ElementManager.h"
#include "Model.h"
```

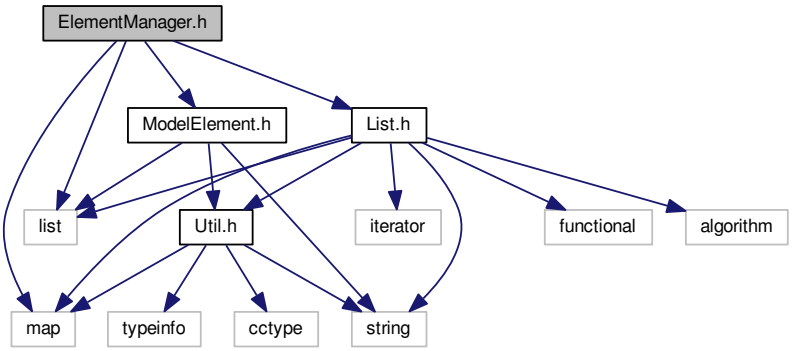
Include dependency graph for ElementManager.cpp:



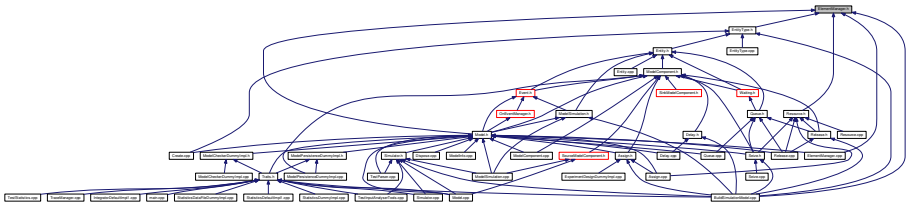
6.26 ElementManager.h File Reference

```
#include <list>
#include <map>
#include "List.h"
#include "ModelElement.h"
```

Include dependency graph for ElementManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ElementManager

6.27 ElementManager_if.h File Reference

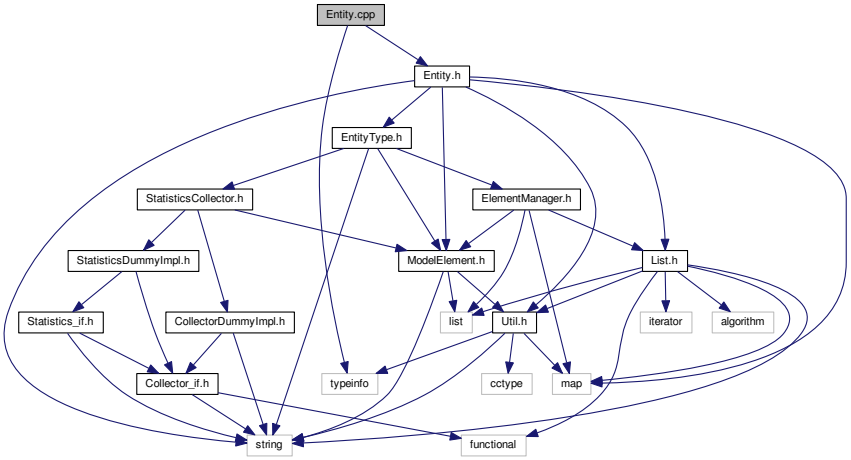
Classes

- class ElementManager_if

6.28 Entity.cpp File Reference

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

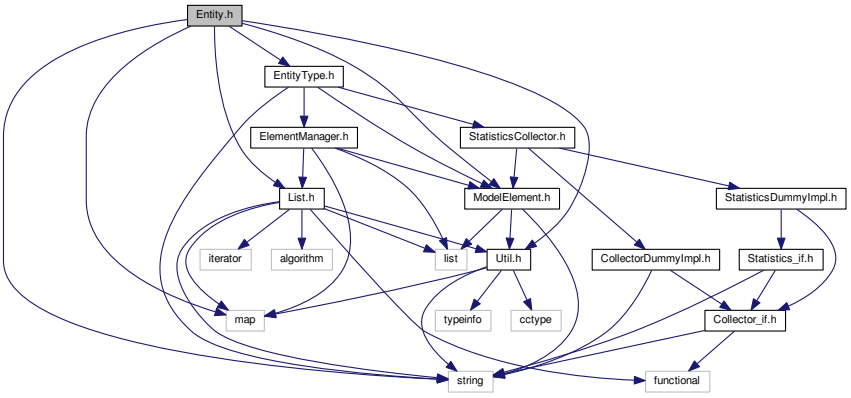
Include dependency graph for Entity.cpp:



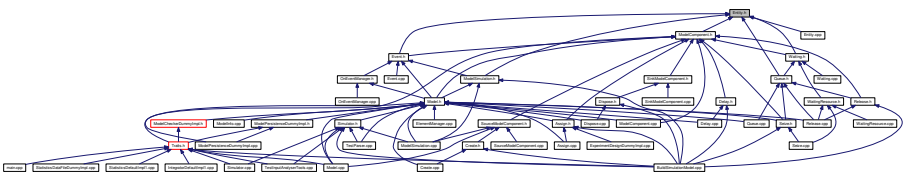
6.29 Entity.h File Reference

```
#include <string>
#include <map>
#include "Util.h"
#include "List.h"
#include "ModelElement.h"
#include "EntityType.h"
```

Include dependency graph for Entity.h:



This graph shows which files directly or indirectly include this file:



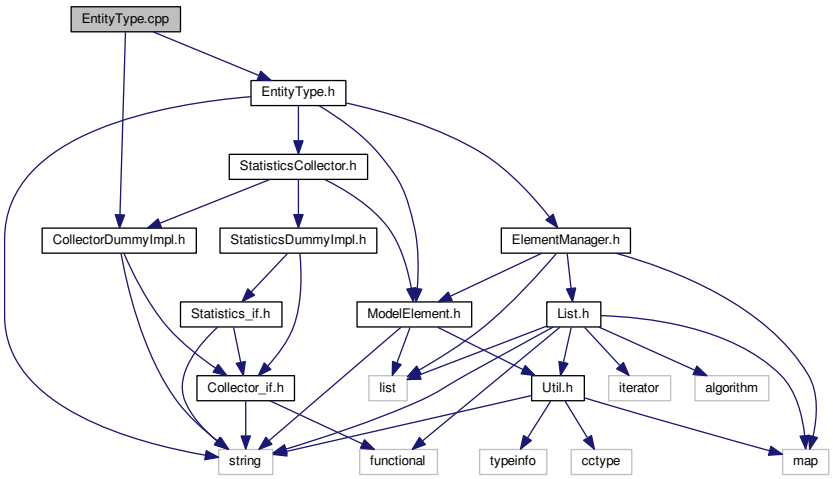
Classes

- class Entity

6.30 EntityType.cpp File Reference

```
#include "EntityType.h"
#include "CollectorDummyImpl.h"
```

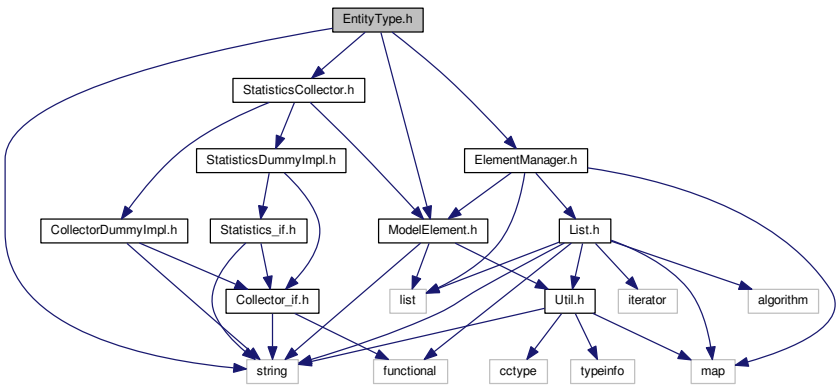
Include dependency graph for EntityType.cpp:



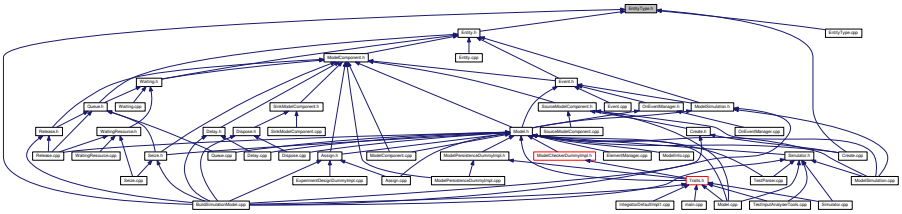
6.31 EntityType.h File Reference

```
#include <string>
#include "ModelElement.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
```

Include dependency graph for EntityType.h:



This graph shows which files directly or indirectly include this file:



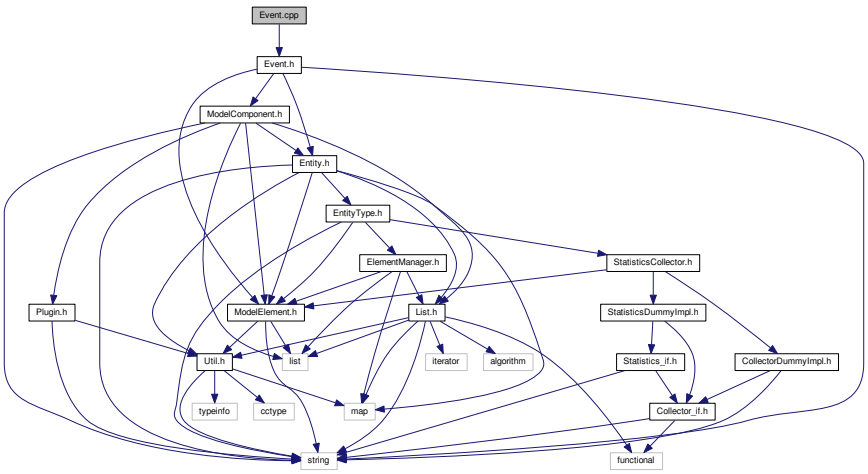
Classes

- class EntityType

6.32 Event.cpp File Reference

```
#include "Event.h"
```

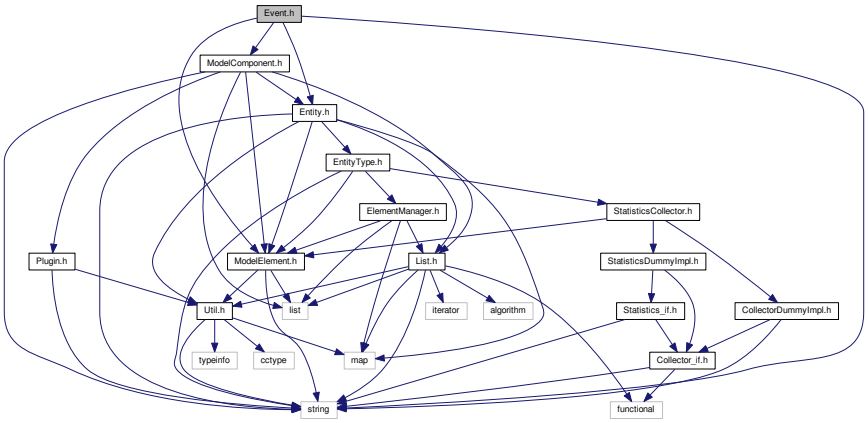
Include dependency graph for Event.cpp:



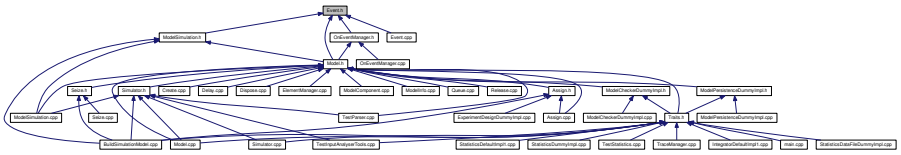
6.33 Event.h File Reference

```
#include <string>
#include "ModelElement.h"
#include "Entity.h"
#include "ModelComponent.h"
```

Include dependency graph for Event.h:



This graph shows which files directly or indirectly include this file:



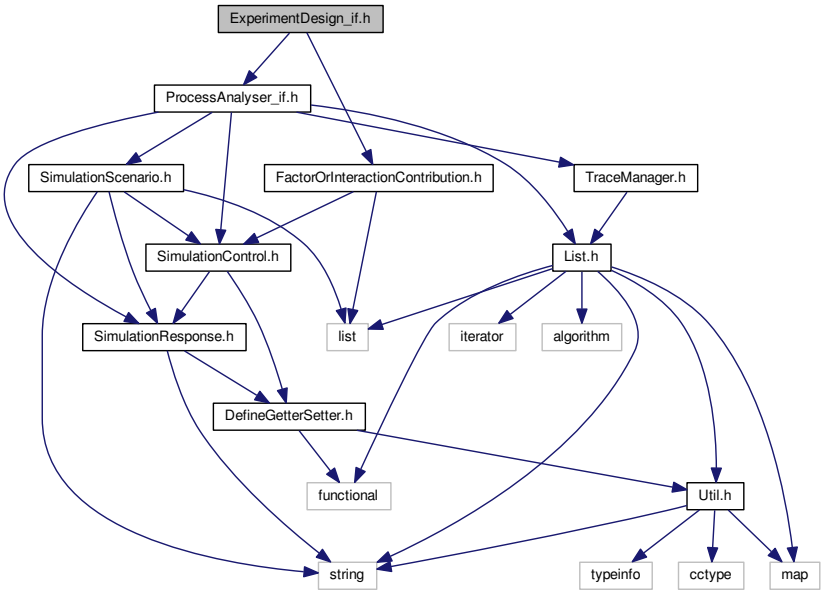
Classes

- class Event

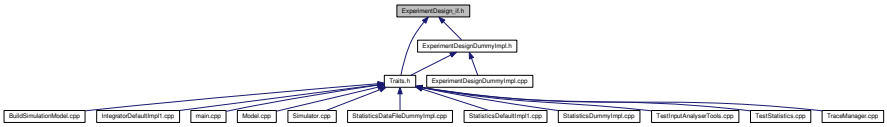
6.34 ExperimentDesign_if.h File Reference

```
#include "FactorOrInteractionContribution.h"
#include "ProcessAnalyser_if.h"
```

Include dependency graph for ExperimentDesign_if.h:



This graph shows which files directly or indirectly include this file:



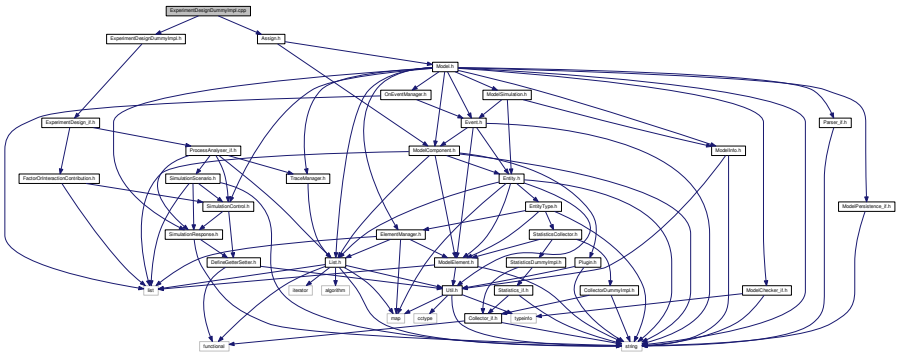
Classes

- class ExperimentDesign_if

6.35 ExperimentDesignDummyImpl.cpp File Reference

```
#include "ExperimentDesignDummyImpl.h"
#include "Assign.h"
```

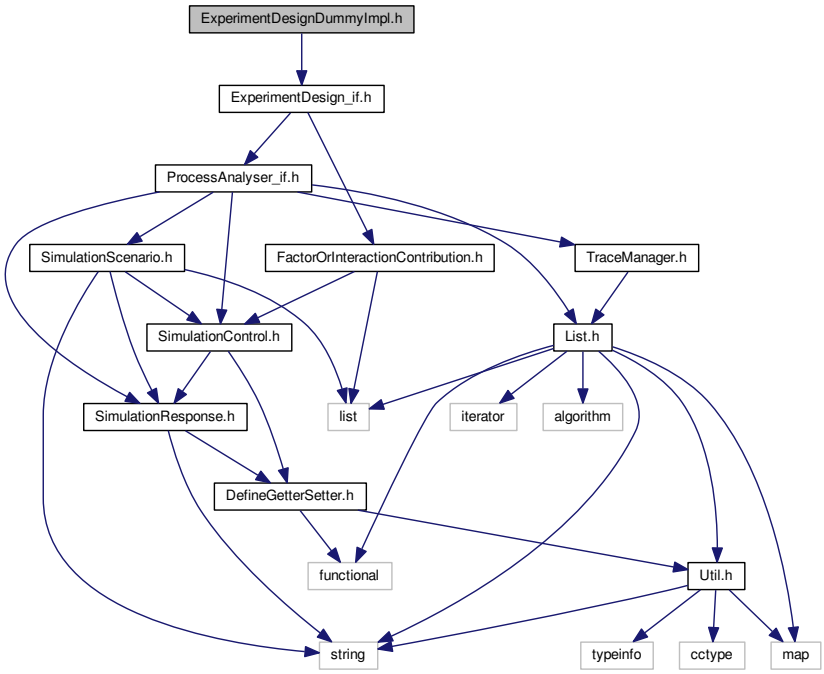
Include dependency graph for ExperimentDesignDummyImpl.cpp:



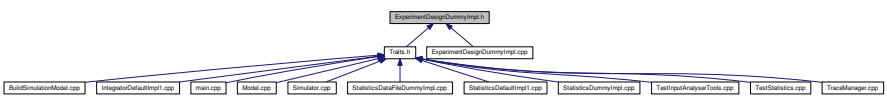
6.36 ExperimentDesignDummyImpl.h File Reference

```
#include "ExperimentDesign_if.h"
```

Include dependency graph for ExperimentDesignDummyImpl.h:



This graph shows which files directly or indirectly include this file:



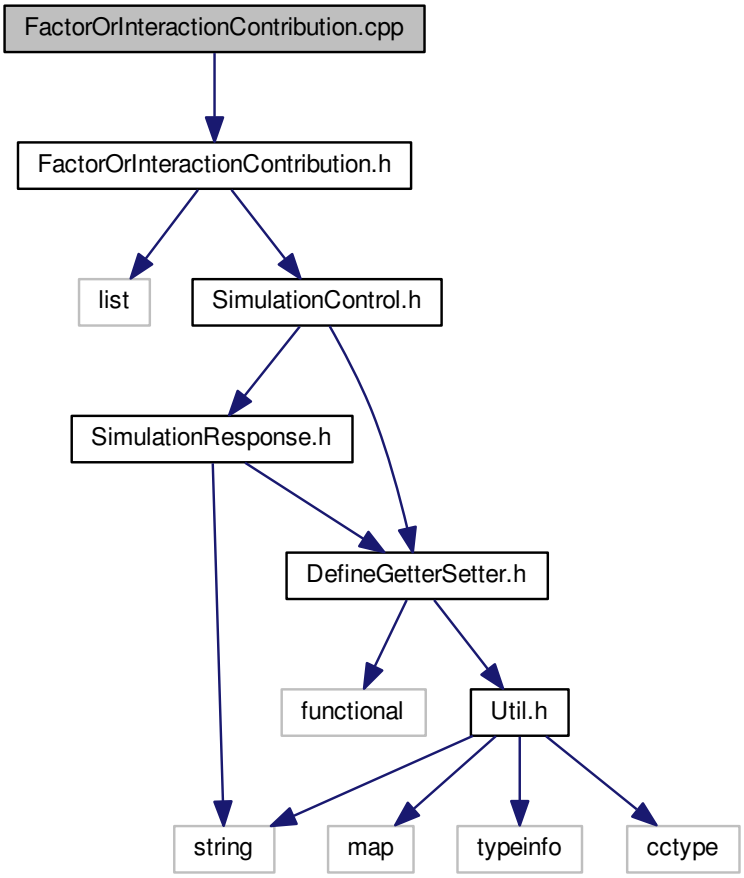
Classes

- class ExperimentDesignDummyImpl

6.37 FactorOrInteractionContribution.cpp File Reference

```
#include "FactorOrInteractionContribution.h"
```

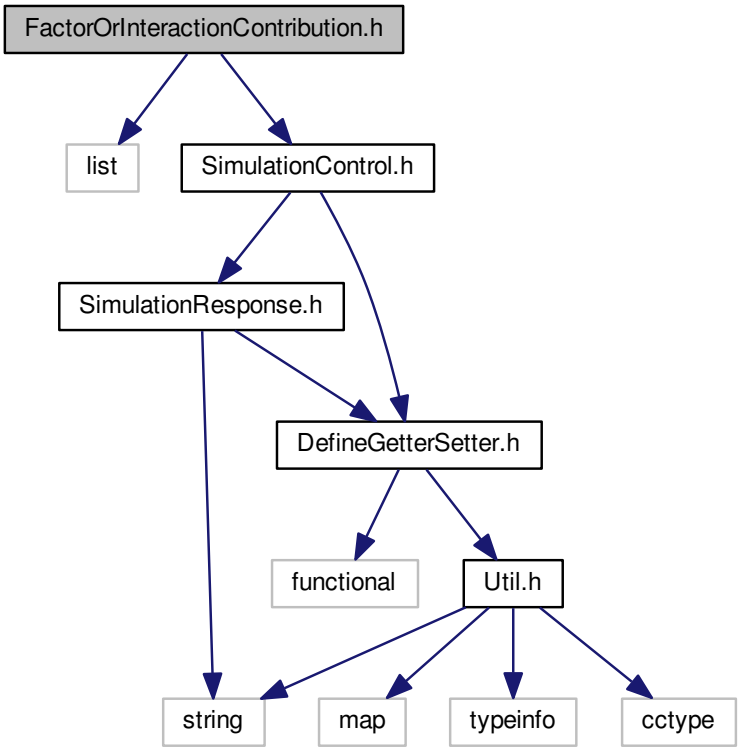
Include dependency graph for FactorOrInteractionContribution.cpp:



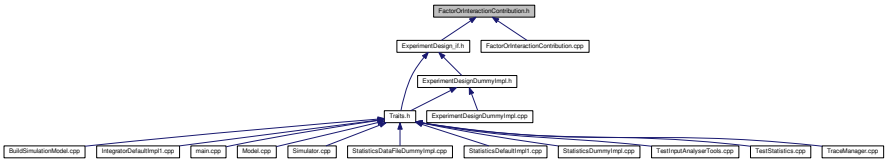
6.38 FactorOrInteractionContribution.h File Reference

```
#include <list>
#include "SimulationControl.h"
```

Include dependency graph for FactorOrInteractionContribution.h:



This graph shows which files directly or indirectly include this file:



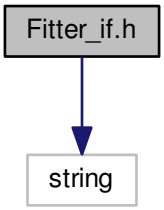
Classes

- class FactorOrInteractionContribution

6.39 Fitter_if.h File Reference

```
#include <string>
```

Include dependency graph for Fitter_if.h:



This graph shows which files directly or indirectly include this file:

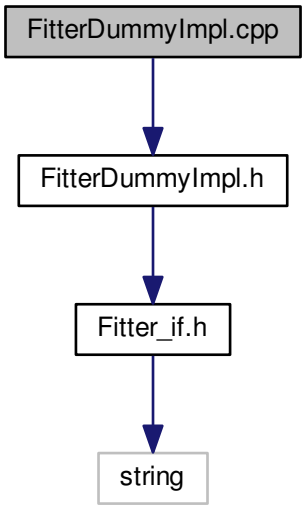
Classes

- class Fitter_if

6.40 FitterDummyImpl.cpp File Reference

```
#include "FitterDummyImpl.h"
```

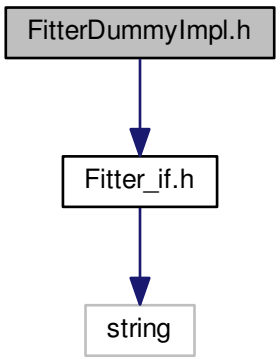
Include dependency graph for FitterDummyImpl.cpp:



6.41 FitterDummyImpl.h File Reference

```
#include "Fitter_if.h"
```

Include dependency graph for FitterDummyImpl.h:



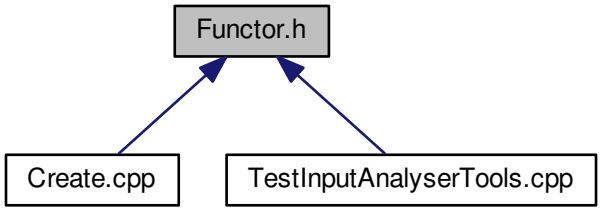
This graph shows which files directly or indirectly include this file:

Classes

- class FitterDummyImpl

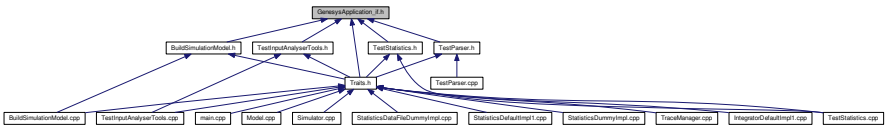
6.42 Functor.h File Reference

This graph shows which files directly or indirectly include this file:



6.43 GenesysApplication_if.h File Reference

This graph shows which files directly or indirectly include this file:



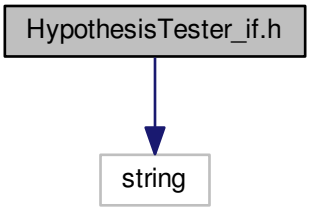
Classes

- class GenesysApplication_if

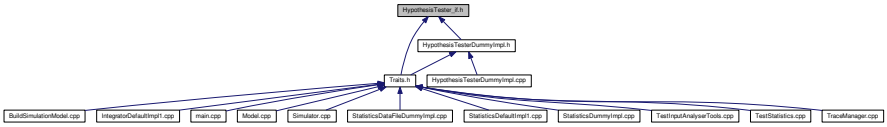
6.44 HypothesisTester_if.h File Reference

```
#include <string>
```

Include dependency graph for HypothesisTester_if.h:



This graph shows which files directly or indirectly include this file:



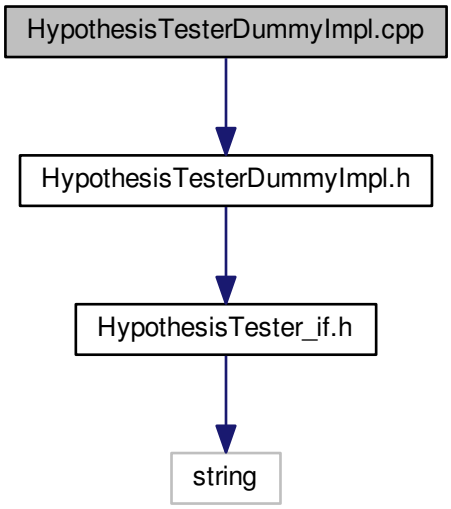
Classes

- class HypothesisTester_if

6.45 HypothesisTesterDummyImpl.cpp File Reference

```
#include "HypothesisTesterDummyImpl.h"
```

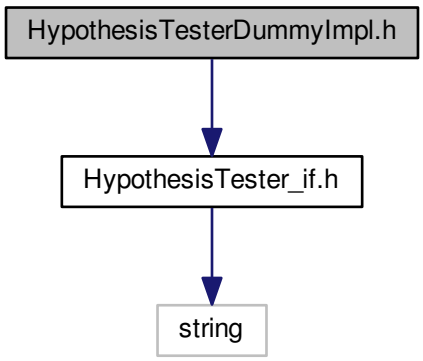
Include dependency graph for HypothesisTesterDummyImpl.cpp:



6.46 HypothesisTesterDummyImpl.h File Reference

```
#include "HypothesisTester_if.h"
```

Include dependency graph for HypothesisTesterDummyImpl.h:



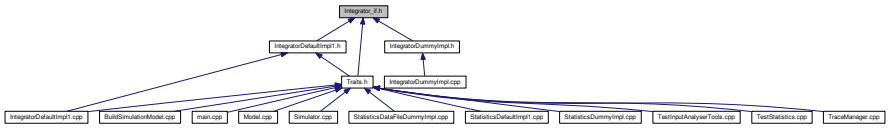
This graph shows which files directly or indirectly include this file:

Classes

- class HypothesisTesterDummyImpl

6.47 Integrator_if.h File Reference

This graph shows which files directly or indirectly include this file:



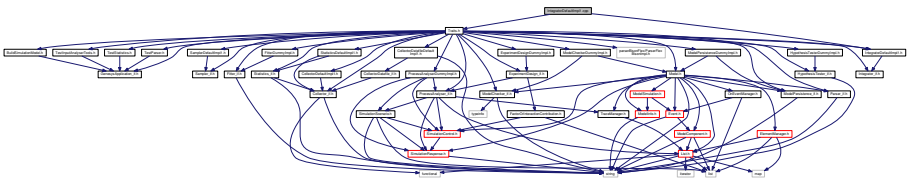
Classes

- class Integrator_if

6.48 IntegratorDefaultImpl1.cpp File Reference

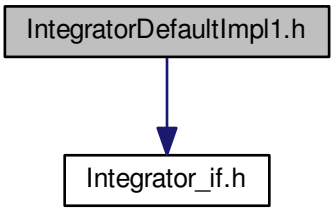
```
#include "IntegratorDefaultImpl1.h"
#include "Traits.h"
```

Include dependency graph for IntegratorDefaultImpl1.cpp:

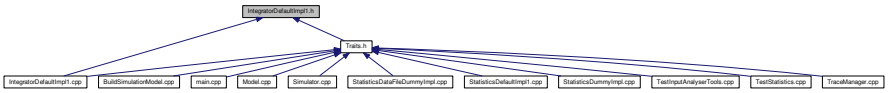


6.49 IntegratorDefaultImpl1.h File Reference

`#include "Integrator_if.h"`
Include dependency graph for IntegratorDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:

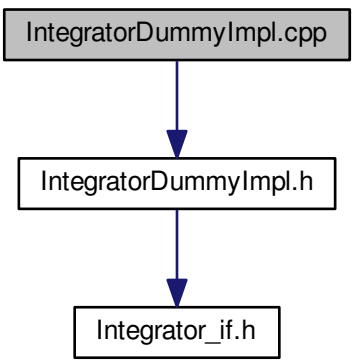


Classes

- class IntegratorDefaultImpl1

6.50 IntegratorDummyImpl.cpp File Reference

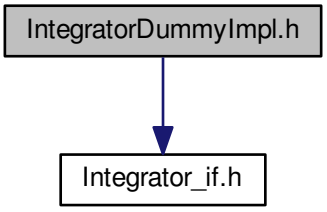
`#include "IntegratorDummyImpl.h"`
Include dependency graph for IntegratorDummyImpl.cpp:



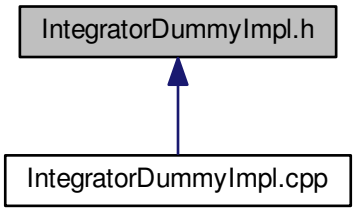
6.51 IntegratorDummyImpl.h File Reference

```
#include "Integrator_if.h"
```

Include dependency graph for IntegratorDummyImpl.h:



This graph shows which files directly or indirectly include this file:



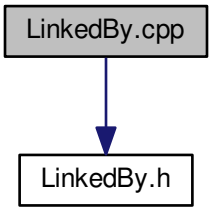
Classes

- class IntegratorDummyImpl

6.52 LinkedBy.cpp File Reference

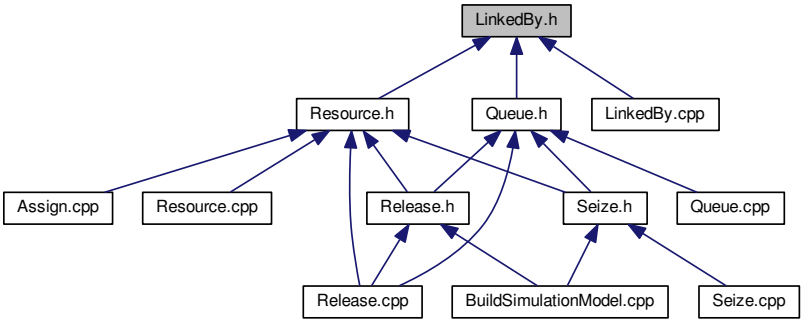
```
#include "LinkedBy.h"
```

Include dependency graph for LinkedBy.cpp:



6.53 LinkedBy.h File Reference

This graph shows which files directly or indirectly include this file:



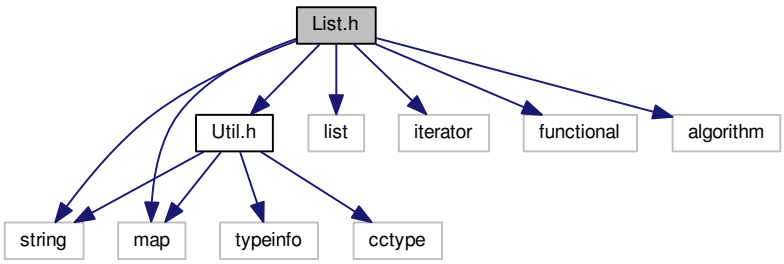
Classes

- class LinkedBy

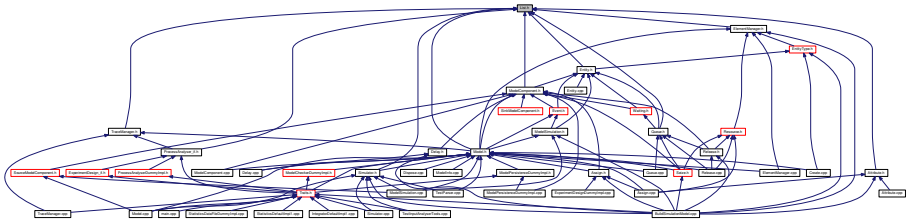
6.54 List.h File Reference

```
#include <string>
#include <list>
#include <map>
#include <iterator>
#include <functional>
#include <algorithm>
#include "Util.h"
```

Include dependency graph for List.h:



This graph shows which files directly or indirectly include this file:



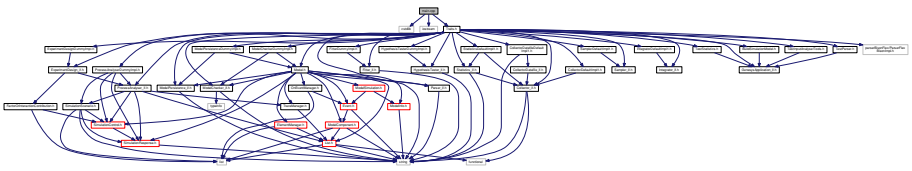
Classes

- class List< T >

6.55 main.cpp File Reference

```
#include <cstdlib>
#include <iostream>
#include "Traits.h"
```

Include dependency graph for main.cpp:



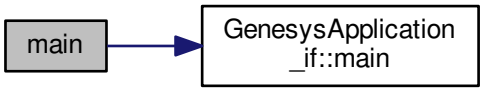
Functions

- int main (int argc, char **argv)

6.55.1 Function Documentation

6.55.1.1 int main (int *argc*, char ** *argv*)

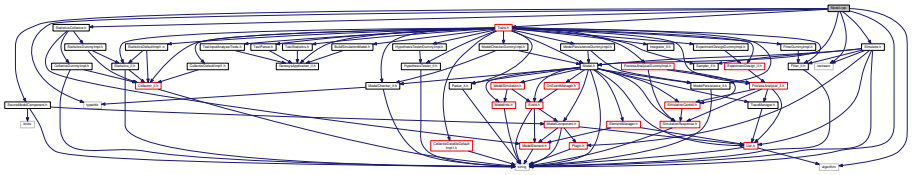
Here is the call graph for this function:



6.56 Model.cpp File Reference

```
#include <typeinfo>
#include <iostream>
#include <algorithm>
#include <string>
#include "Model.h"
#include "SourceModelComponent.h"
#include "Simulator.h"
#include "StatisticsCollector.h"
#include "Traits.h"
```

Include dependency graph for Model.cpp:



Functions

- bool EventCompare (const Event *a, const Event *b)
- double getReplicationLengthNotMemberFunction ()
- void setReplicationLengthNotMemberFunction (double value)

6.56.1 Function Documentation

6.56.1.1 bool EventCompare (const Event * *a*, const Event * *b*)

Here is the call graph for this function:

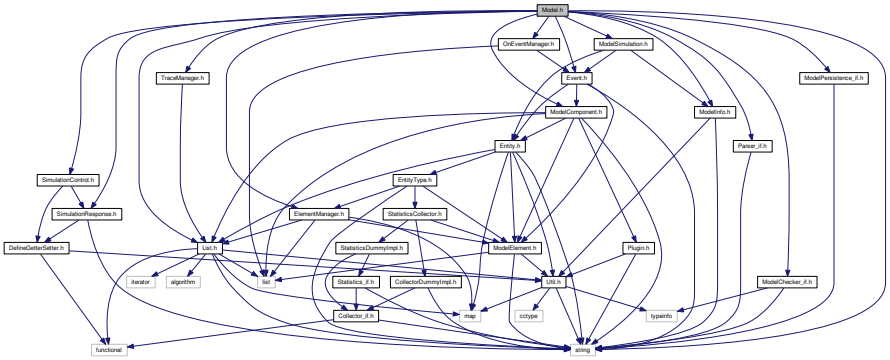
6.56.1.2 double getReplicationLengthNotMemberFunction ( )

6.56.1.3 void setReplicationLengthNotMemberFunction (double *value*)

6.57 Model.h File Reference

```
#include <string>
#include "List.h"
#include "ModelComponent.h"
#include "Event.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "ModelPersistence_if.h"
#include "ElementManager.h"
#include "TraceManager.h"
#include "OnEventManager.h"
#include "ModelInfo.h"
#include "ModelSimulation.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Include dependency graph for Model.h:



This graph shows which files directly or indirectly include this file:

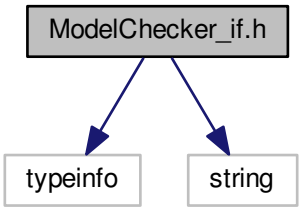
Classes

- class Model

6.58 ModelChecker_if.h File Reference

```
#include <typeinfo>
#include <string>
```

Include dependency graph for ModelChecker_if.h:



This graph shows which files directly or indirectly include this file:

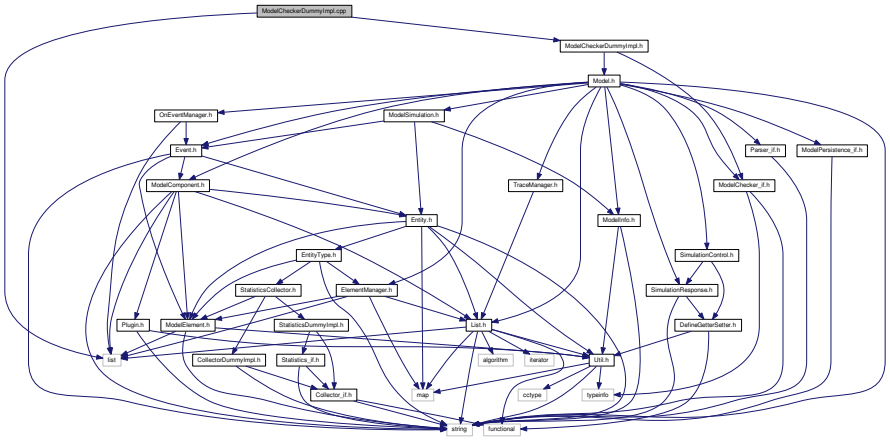
Classes

- class ModelChecker_if

6.59 ModelCheckerDummyImpl.cpp File Reference

```
#include <list>
#include "ModelCheckerDummyImpl.h"
```

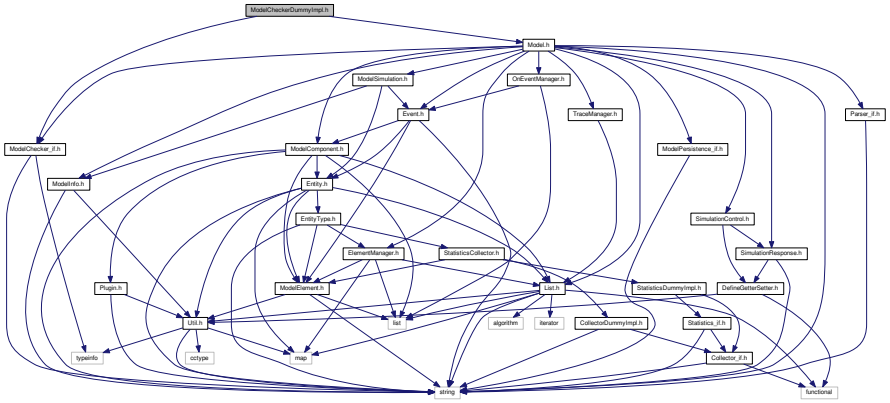
Include dependency graph for ModelCheckerDummyImpl.cpp:



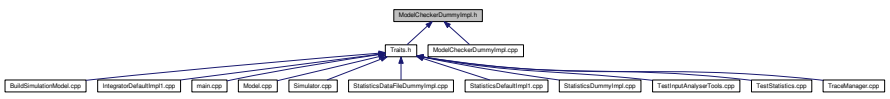
6.60 ModelCheckerDummyImpl.h File Reference

```
#include "ModelChecker_if.h"
#include "Model.h"
```

Include dependency graph for ModelCheckerDummyImpl.h:



This graph shows which files directly or indirectly include this file:



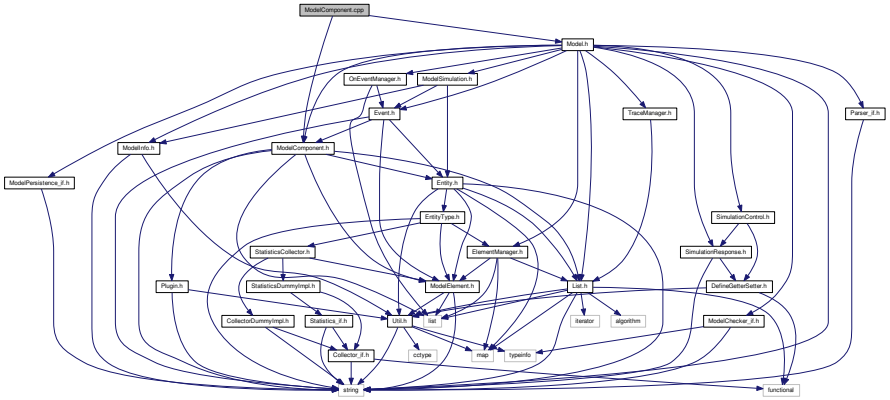
Classes

- class ModelCheckerDummyImpl

6.61 ModelComponent.cpp File Reference

```
#include "ModelComponent.h"
#include "Model.h"
```

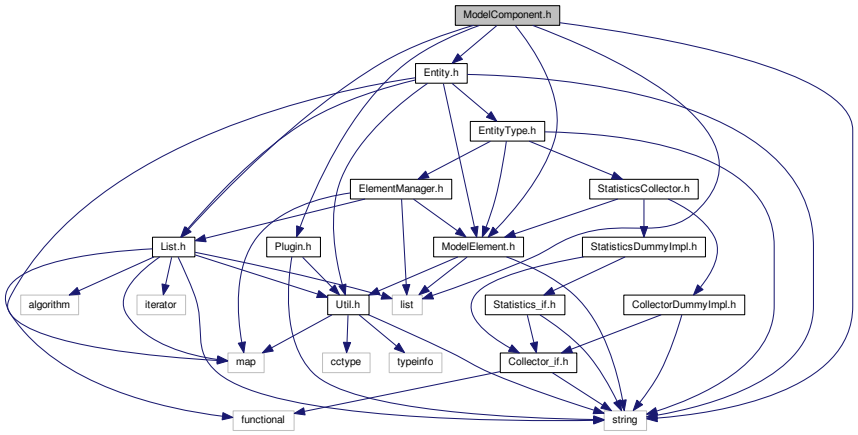
Include dependency graph for ModelComponent.cpp:



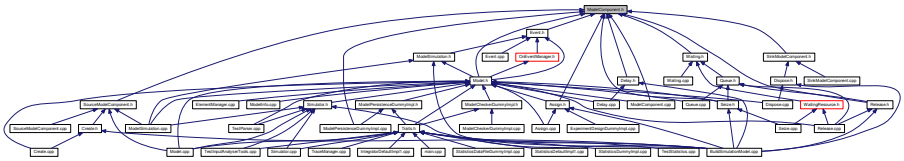
6.62 ModelComponent.h File Reference

```
#include <string>
#include <list>
#include "Plugin.h"
#include "List.h"
#include "Entity.h"
#include "ModelElement.h"
```

Include dependency graph for ModelComponent.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ModelComponent

6.63 ModelComponentManager_if.h File Reference

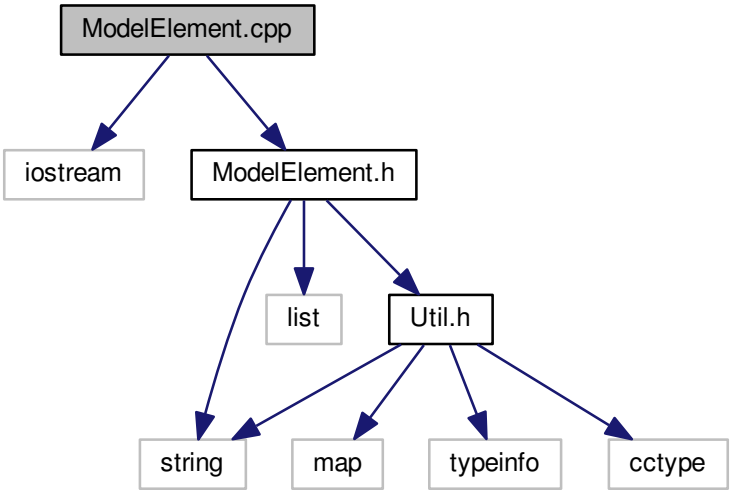
Classes

- class ModelComponentManager_if

6.64 ModelElement.cpp File Reference

```
#include <iostream>
#include "ModelElement.h"
```

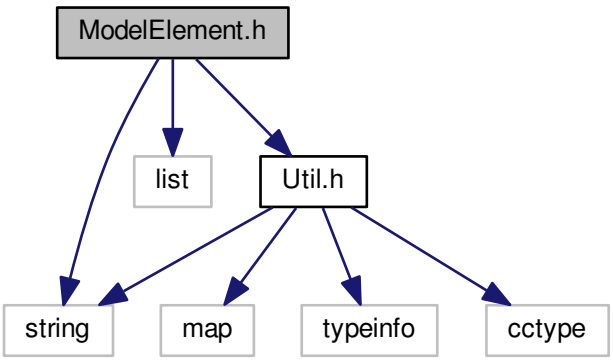
Include dependency graph for ModelElement.cpp:



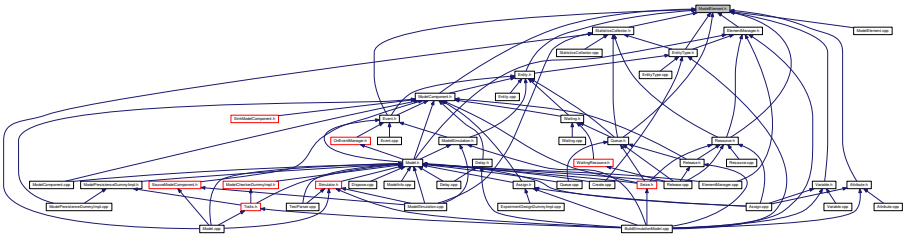
6.65 ModelElement.h File Reference

```
#include <string>
#include <list>
#include "Util.h"
```

Include dependency graph for ModelElement.h:



This graph shows which files directly or indirectly include this file:



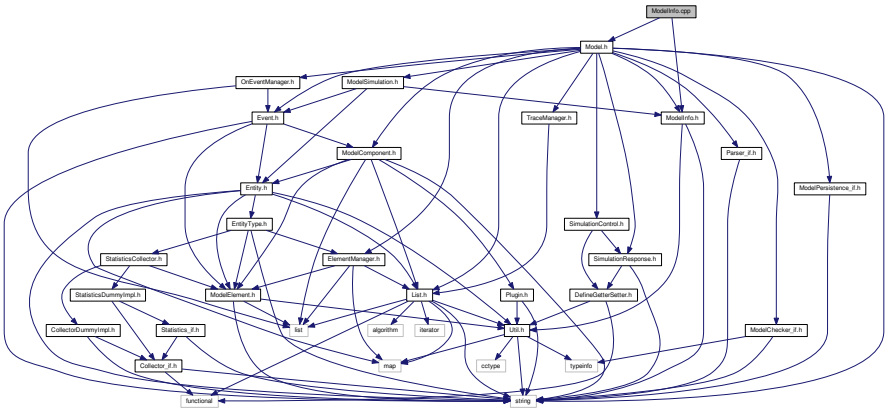
Classes

- class ModelElement

6.66 ModelInfo.cpp File Reference

```
#include "ModelInfo.h"
#include "Model.h"
```

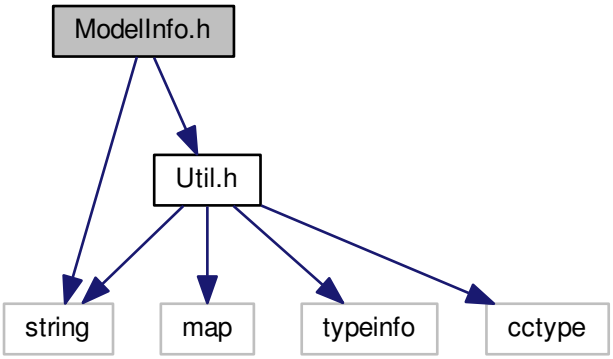
Include dependency graph for ModelInfo.cpp:



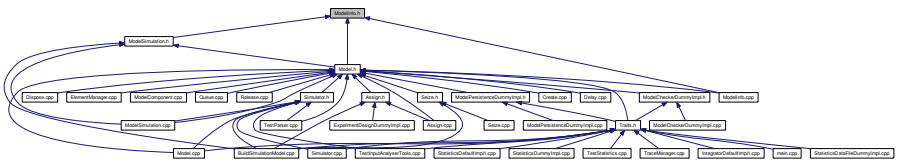
6.67 ModelInfo.h File Reference

```
#include <string>
#include "Util.h"
```

Include dependency graph for ModelInfo.h:



This graph shows which files directly or indirectly include this file:



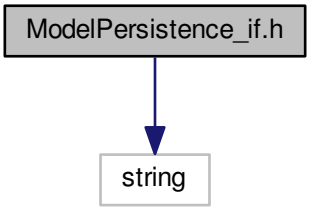
Classes

- class ModelInfo

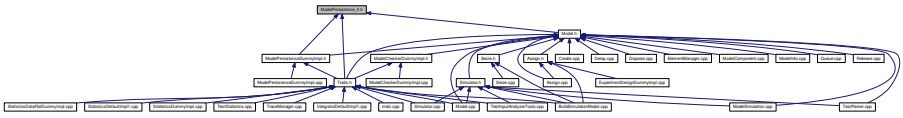
6.68 ModelPersistence_if.h File Reference

```
#include <string>
```

Include dependency graph for ModelPersistence_if.h:



This graph shows which files directly or indirectly include this file:



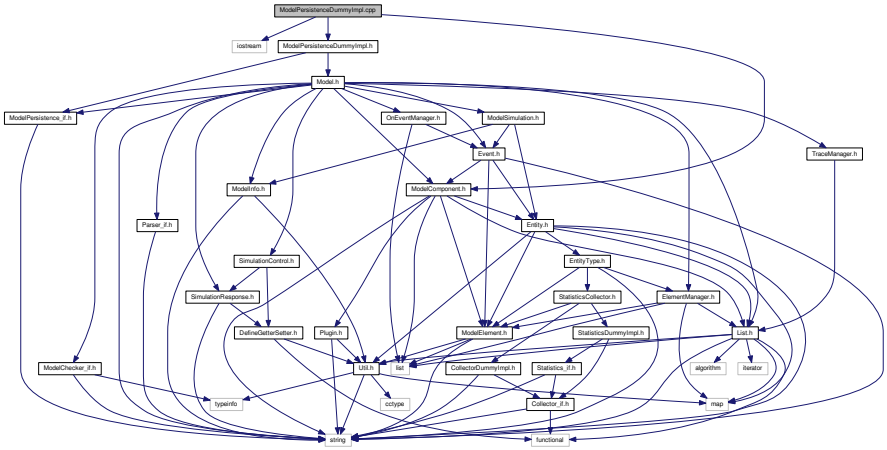
Classes

- class ModelPersistence_if

6.69 ModelPersistenceDummyImpl.cpp File Reference

```
#include <iostream>
#include "ModelPersistenceDummyImpl.h"
#include "ModelComponent.h"
```

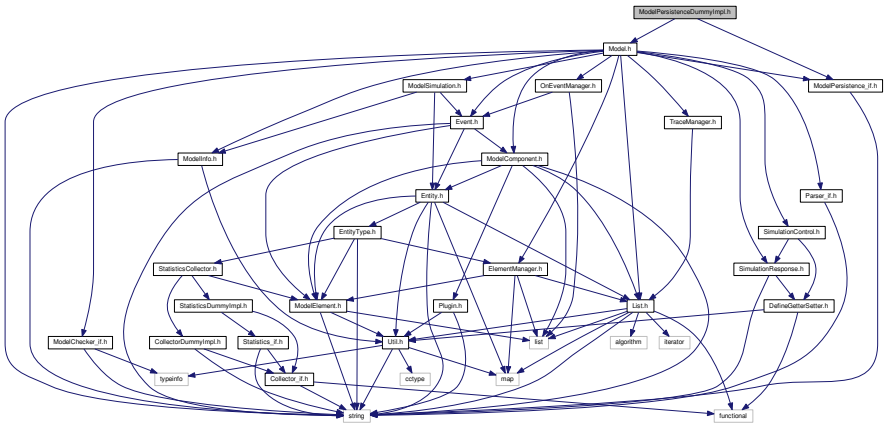
Include dependency graph for ModelPersistenceDummyImpl.cpp:



6.70 ModelPersistenceDummyImpl.h File Reference

```
#include "ModelPersistence_if.h"
#include "Model.h"
```

Include dependency graph for ModelPersistenceDummyImpl.h:



This graph shows which files directly or indirectly include this file:

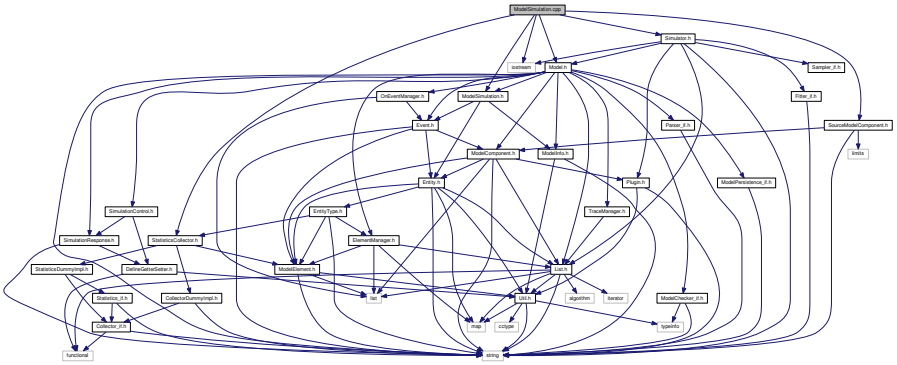
Classes

- class ModelPersistenceDummyImpl

6.71 ModelSimulation.cpp File Reference

```
#include <iostream>
#include "ModelSimulation.h"
#include "Model.h"
#include "Simulator.h"
#include "SourceModelComponent.h"
#include "StatisticsCollector.h"
```

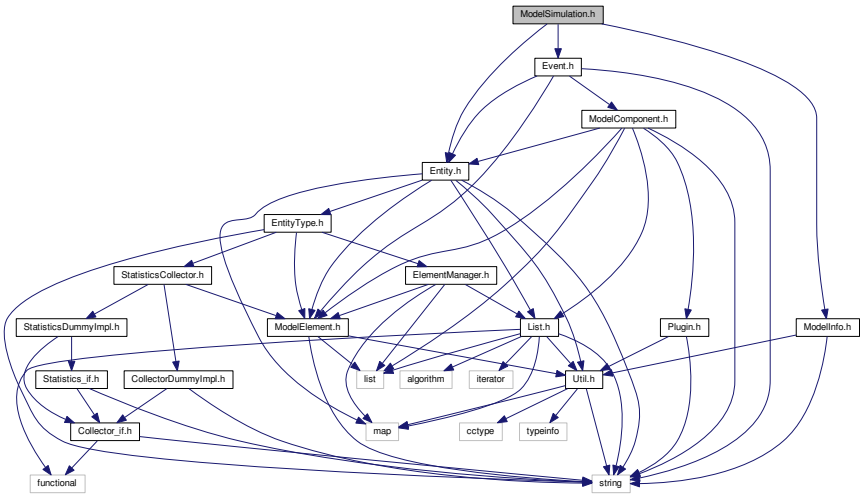
Include dependency graph for ModelSimulation.cpp:



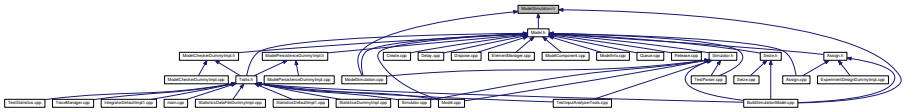
6.72 ModelSimulation.h File Reference

```
#include "Event.h"
#include "Entity.h"
#include "ModelInfo.h"
```

Include dependency graph for ModelSimulation.h:



This graph shows which files directly or indirectly include this file:



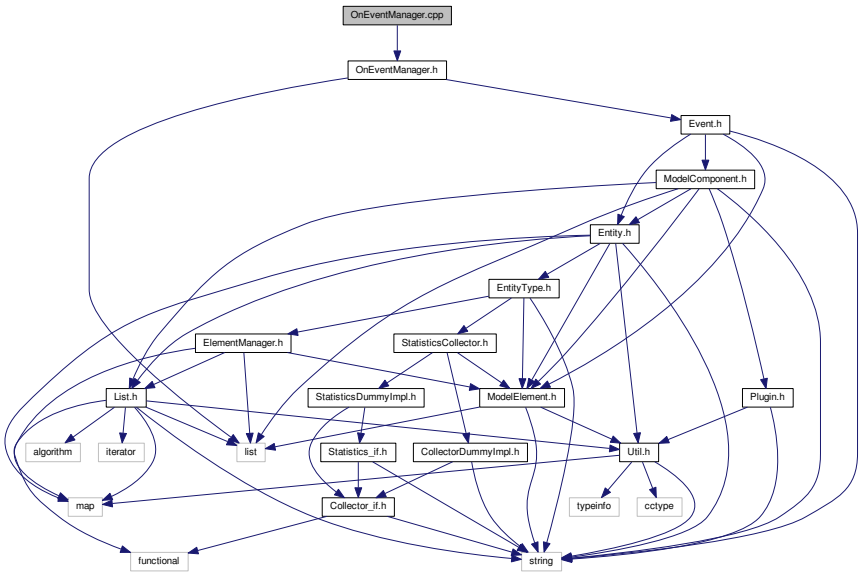
Classes

- class ModelSimulation

6.73 OnEventManager.cpp File Reference

```
#include "OnEventManager.h"
```

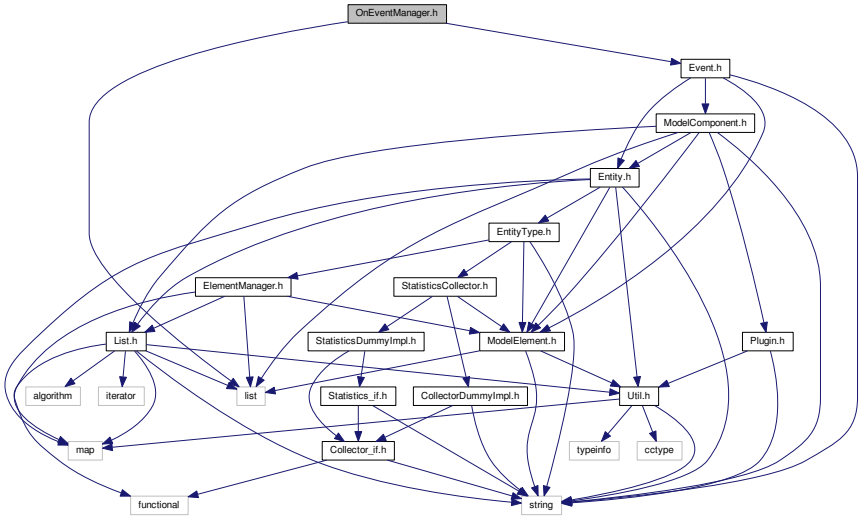
Include dependency graph for OnEventManager.cpp:



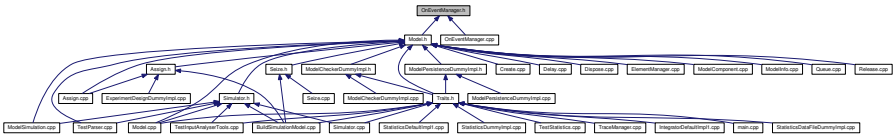
6.74 OnEventManager.h File Reference

```
#include <list>
#include "Event.h"
```

Include dependency graph for OnEventManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SimulationEvent
- class OnEventManager

Typedefs

- typedef void(∗ simulationEventHandler) (SimulationEvent ∗)

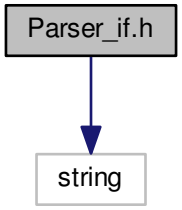
6.74.1 Typedef Documentation

6.74.1.1 typedef void(∗ simulationEventHandler) (SimulationEvent ∗)

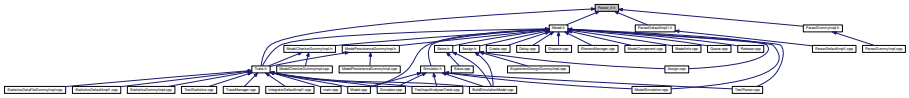
6.75 Parser_if.h File Reference

```
#include <string>
```

Include dependency graph for Parser_if.h:



This graph shows which files directly or indirectly include this file:



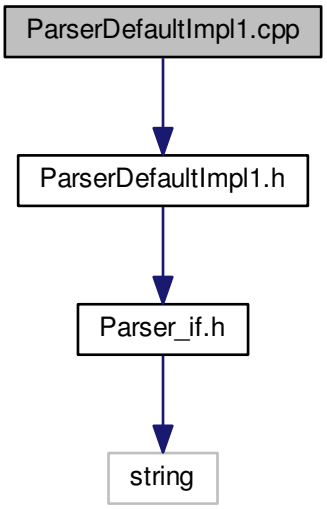
Classes

- class Parser_if

6.76 ParserDefaultImpl1.cpp File Reference

```
#include "ParserDefaultImpl1.h"
```

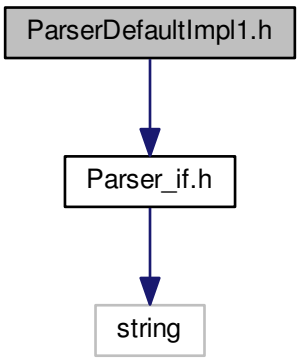
Include dependency graph for ParserDefaultImpl1.cpp:



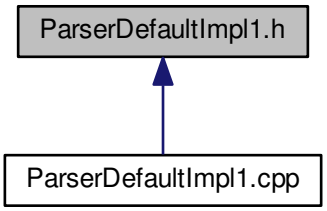
6.77 ParserDefaultImpl1.h File Reference

```
#include "Parser_if.h"
```

Include dependency graph for ParserDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



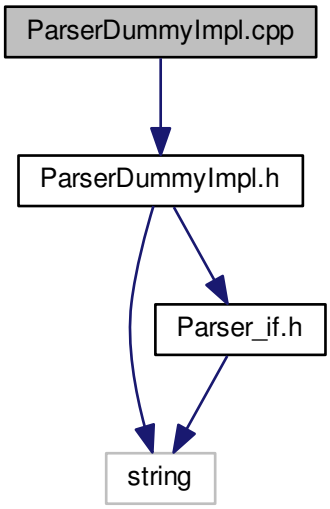
Classes

- class ParserDefaultImpl1

6.78 ParserDummyImpl.cpp File Reference

```
#include "ParserDummyImpl.h"
```

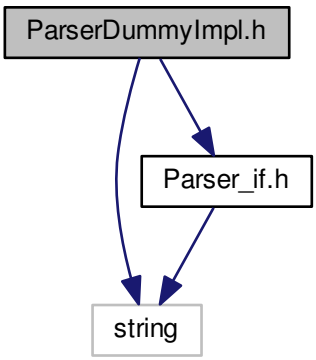
Include dependency graph for ParserDummyImpl.cpp:



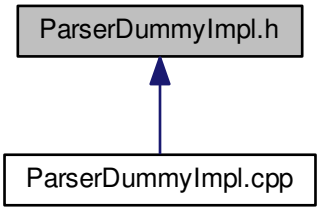
6.79 ParserDummyImpl.h File Reference

```
#include <string>
#include "Parser_if.h"
```

Include dependency graph for ParserDummyImpl.h:



This graph shows which files directly or indirectly include this file:



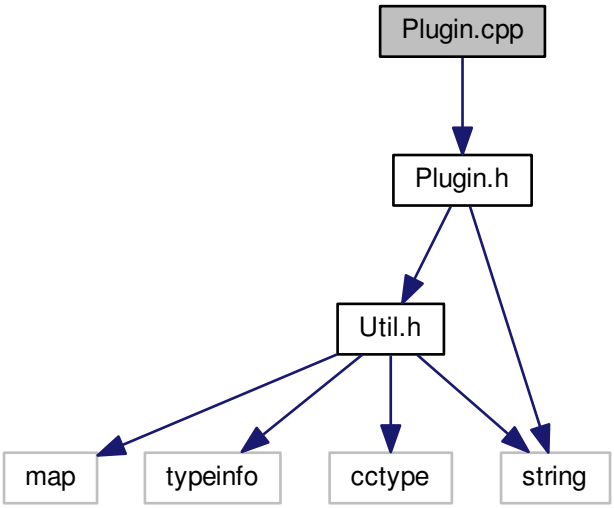
Classes

- class ParserDummyImpl

6.80 Plugin.cpp File Reference

```
#include "Plugin.h"
```

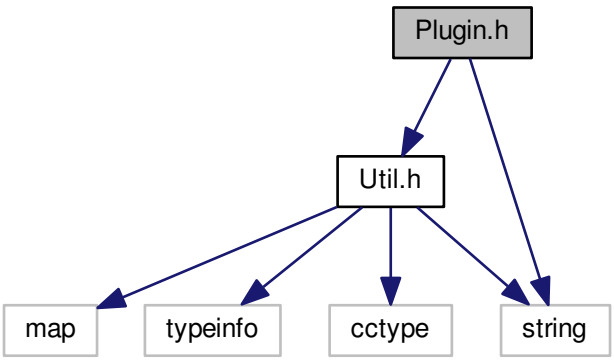
Include dependency graph for Plugin.cpp:



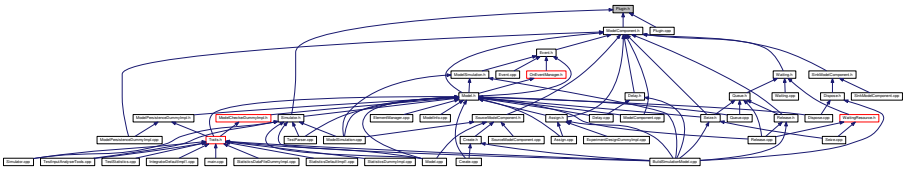
6.81 Plugin.h File Reference

```
#include "Util.h"
#include <string>
```

Include dependency graph for Plugin.h:



This graph shows which files directly or indirectly include this file:



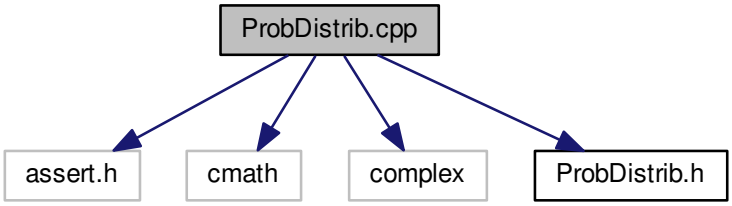
Classes

- class Plugin

6.82 ProbDistrib.cpp File Reference

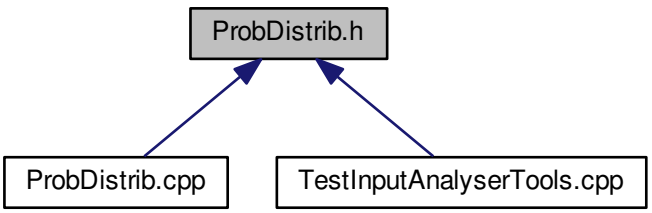
```
#include <assert.h>
#include <cmath>
#include <complex>
#include "ProbDistrib.h"
```

Include dependency graph for ProbDistrib.cpp:



6.83 ProbDistrib.h File Reference

This graph shows which files directly or indirectly include this file:



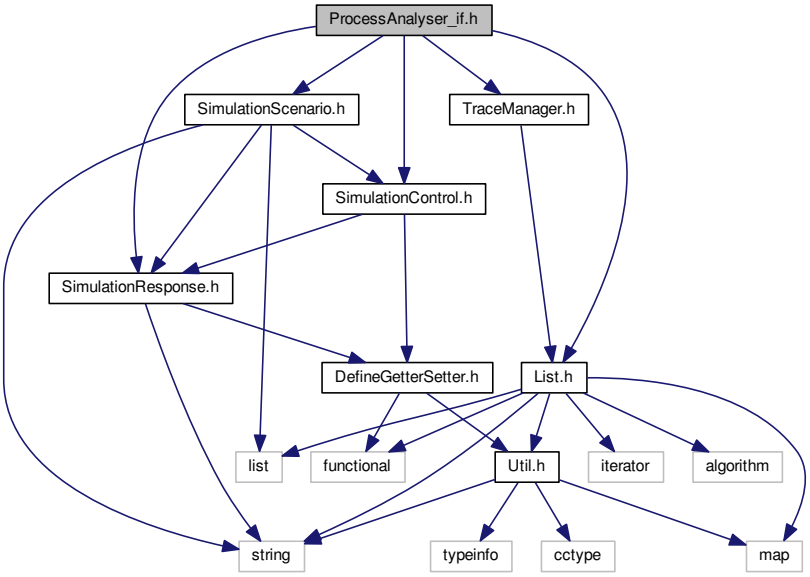
Classes

- class ProbDistrib

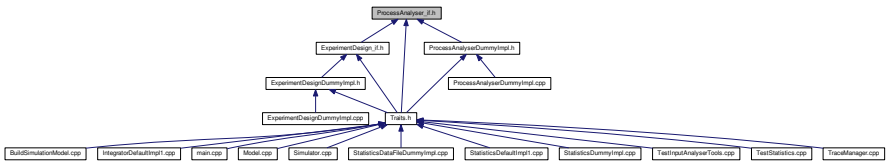
6.84 ProcessAnalyser_if.h File Reference

```
#include "List.h"
#include "SimulationScenario.h"
#include "SimulationControl.h"
#include "SimulationResponse.h"
#include "TraceManager.h"
```

Include dependency graph for ProcessAnalyser_if.h:



This graph shows which files directly or indirectly include this file:



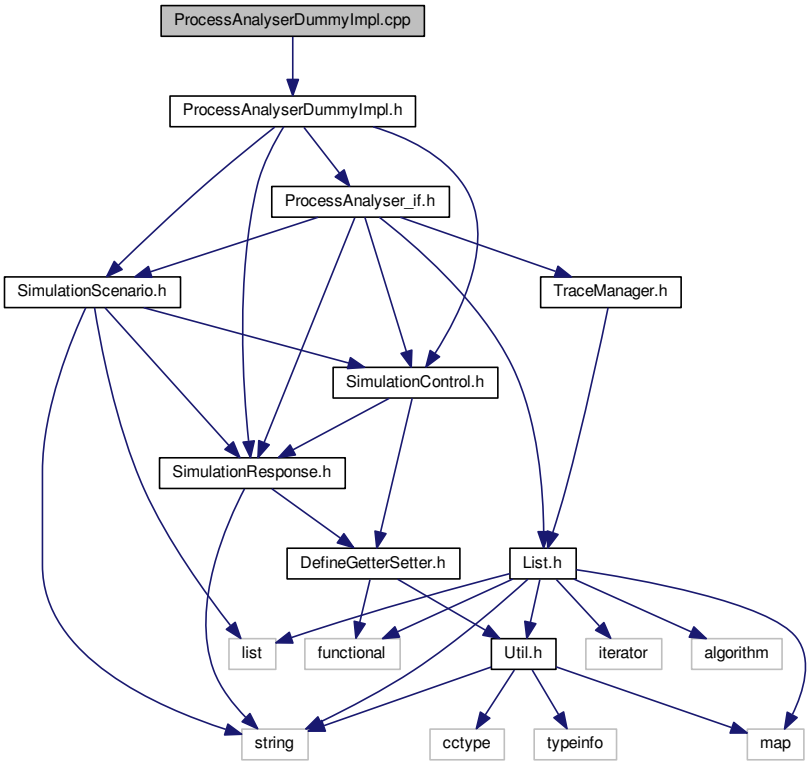
Classes

- class ProcessAnalyser_if

6.85 ProcessAnalyserDummyImpl.cpp File Reference

```
#include "ProcessAnalyserDummyImpl.h"
```

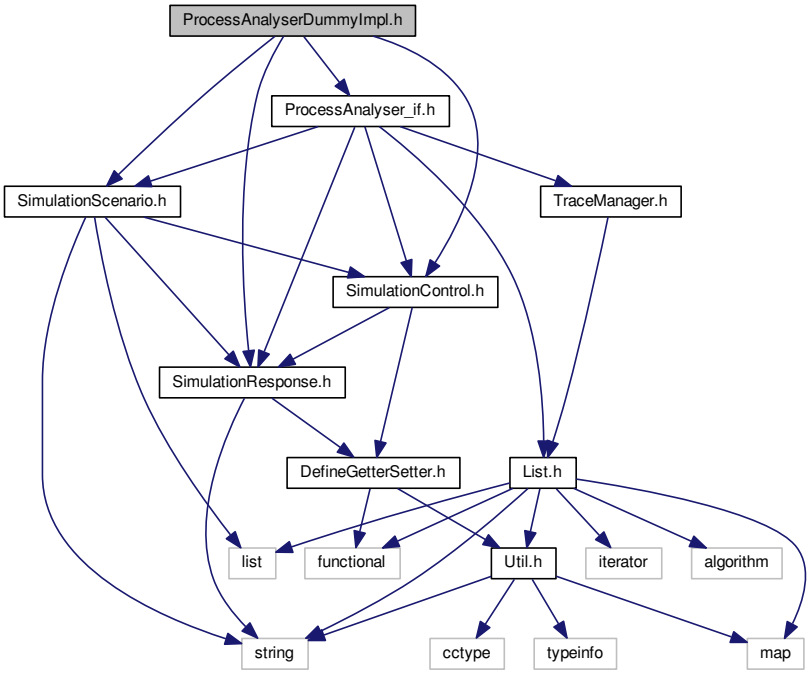
Include dependency graph for ProcessAnalyserDummyImpl.cpp:



6.86 ProcessAnalyserDummyImpl.h File Reference

```
#include "ProcessAnalyser_if.h"
#include "SimulationScenario.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Include dependency graph for ProcessAnalyserDummyImpl.h:



This graph shows which files directly or indirectly include this file:

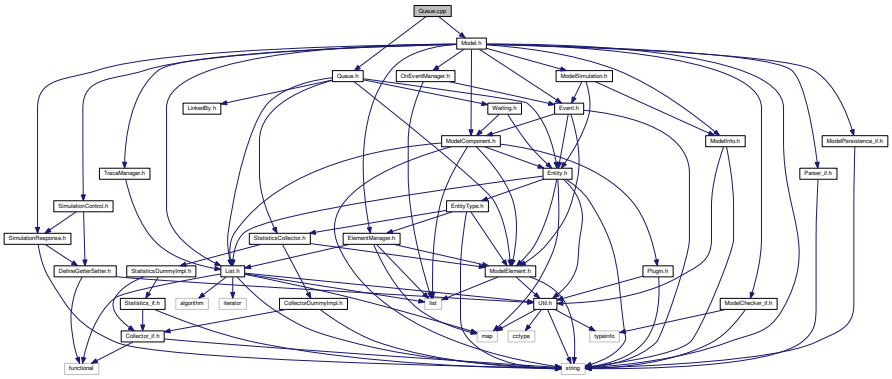
Classes

- class ProcessAnalyserDummyImpl

6.87 Queue.cpp File Reference

```
#include "Queue.h"
#include "Model.h"
```

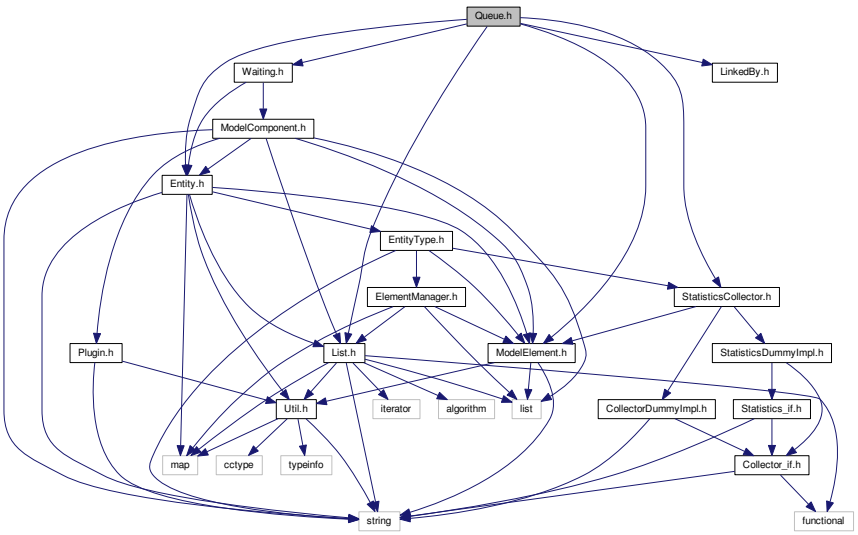
Include dependency graph for Queue.cpp:



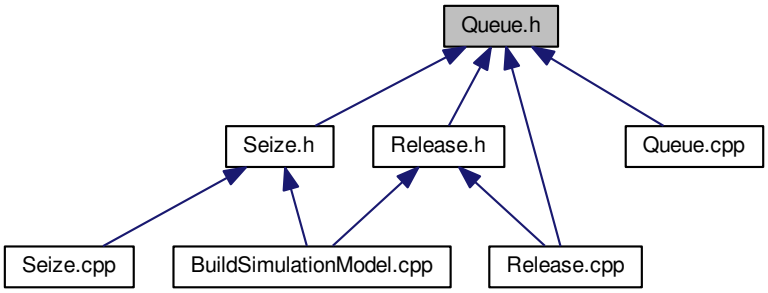
6.88 Queue.h File Reference

```
#include "ModelElement.h"
#include "LinkedBy.h"
#include "List.h"
#include "Entity.h"
#include "Waiting.h"
#include "StatisticsCollector.h"
```

Include dependency graph for Queue.h:



This graph shows which files directly or indirectly include this file:



Classes

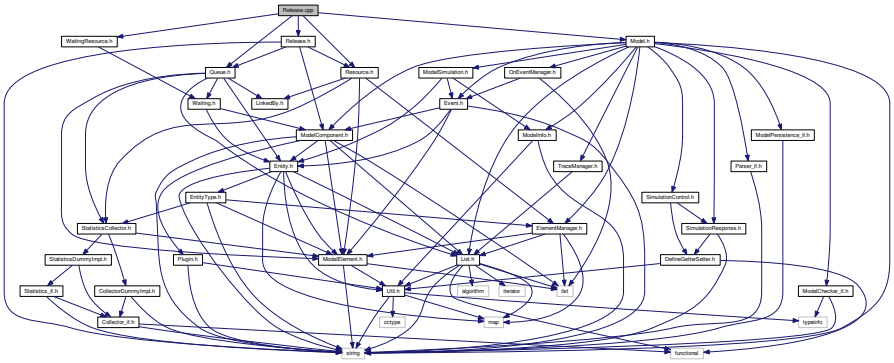
- class Queue

6.89 README.md File Reference

6.90 Release.cpp File Reference

```
#include "Release.h"
#include "Model.h"
#include "WaitingResource.h"
#include "Queue.h"
#include "Resource.h"
```

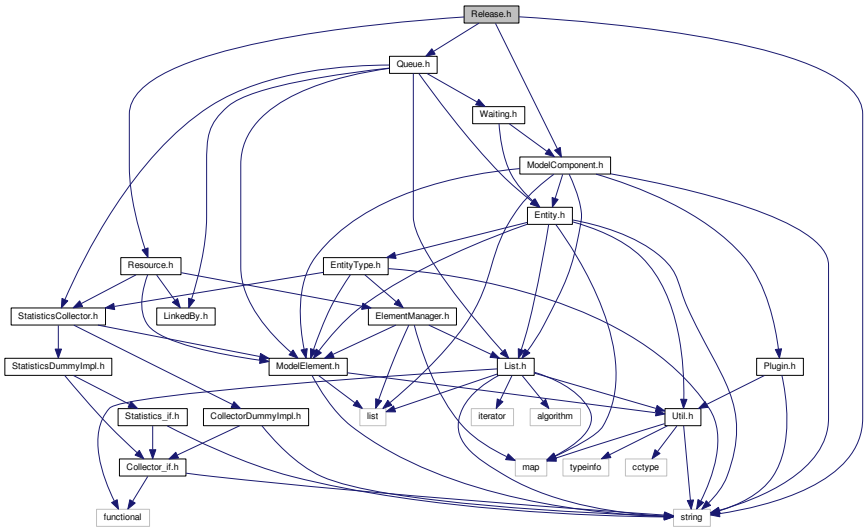
Include dependency graph for Release.cpp:



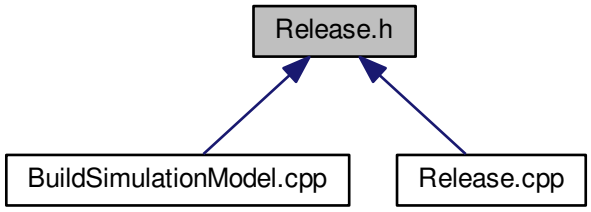
6.91 Release.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Resource.h"
#include "Queue.h"
```

Include dependency graph for Release.h:



This graph shows which files directly or indirectly include this file:



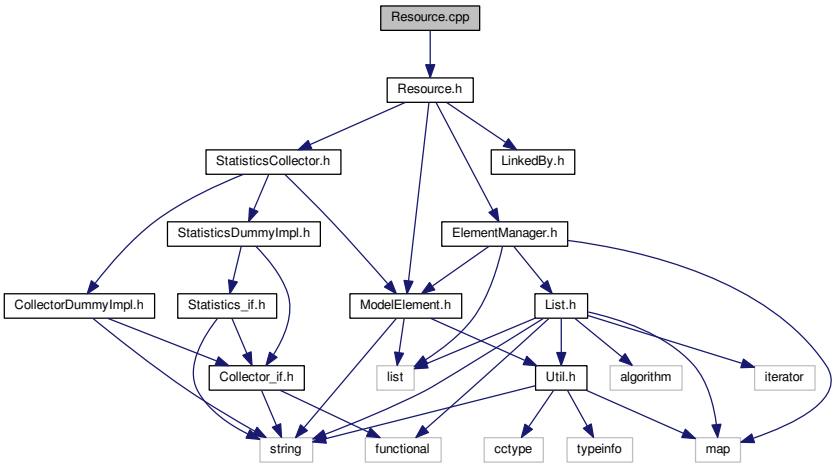
Classes

- class Release

6.92 Resource.cpp File Reference

```
#include "Resource.h"
```

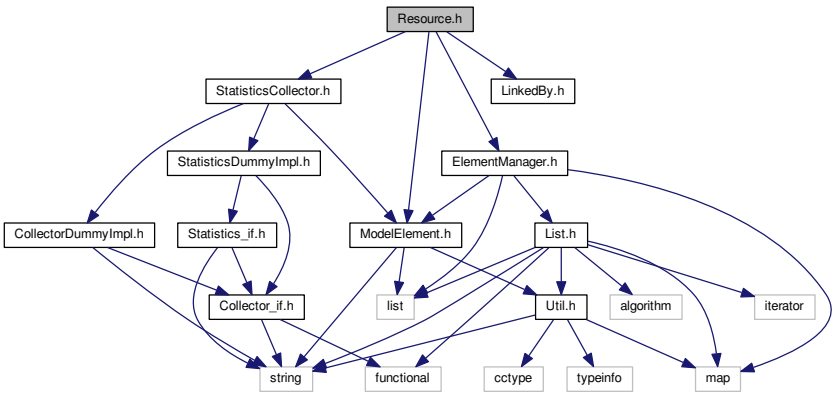
Include dependency graph for Resource.cpp:



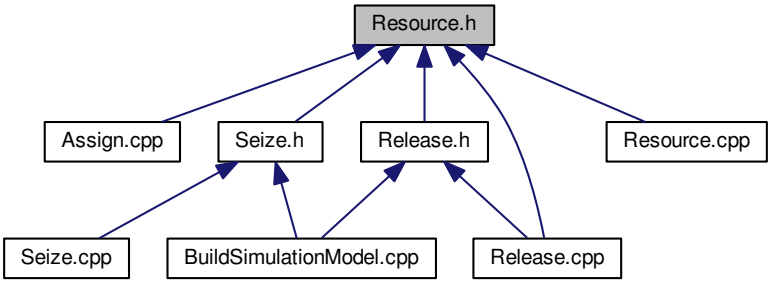
6.93 Resource.h File Reference

```
#include "ModelElement.h"
#include "LinkedBy.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
```

Include dependency graph for Resource.h:



This graph shows which files directly or indirectly include this file:

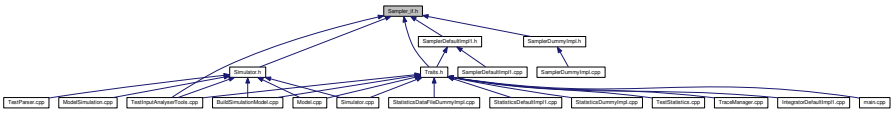


Classes

- class Resource

6.94 Sampler_if.h File Reference

This graph shows which files directly or indirectly include this file:



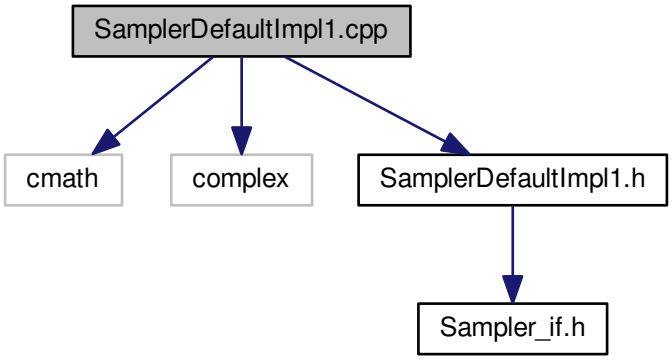
Classes

- class Sampler_if
- class Sampler_if::RNG_Parameters

6.95 SamplerDefaultImpl1.cpp File Reference

```
#include <cmath>
#include <complex>
#include "SamplerDefaultImpl1.h"
```

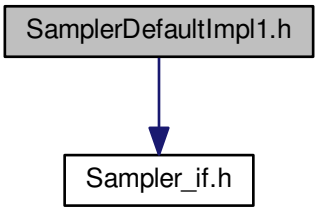
Include dependency graph for SamplerDefaultImpl1.cpp:



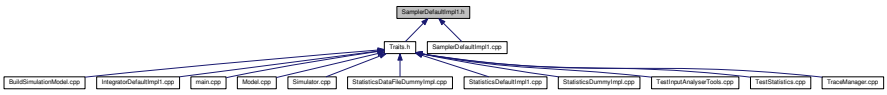
6.96 SamplerDefaultImpl1.h File Reference

```
#include "Sampler_if.h"
```

Include dependency graph for SamplerDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



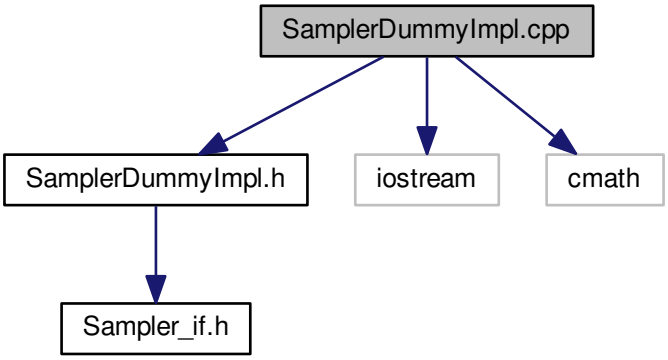
Classes

- class SamplerDefaultImpl1
- class SamplerDefaultImpl1::DefaultImpl1RNG_Parameters

6.97 SamplerDummyImpl.cpp File Reference

```
#include "SamplerDummyImpl.h"
#include <iostream>
#include <cmath>
```

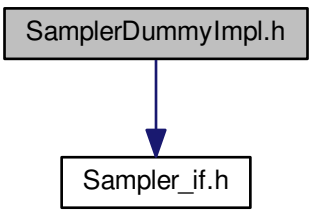
Include dependency graph for SamplerDummyImpl.cpp:



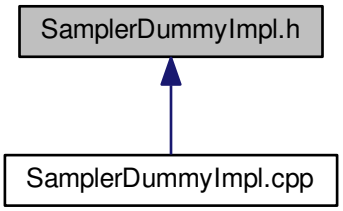
6.98 SamplerDummyImpl.h File Reference

```
#include "Sampler_if.h"
```

Include dependency graph for SamplerDummyImpl.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SamplerDummyImpl
- class SamplerDummyImpl::MyRNG_Parameters

6.99 ScenarioExperiment_if.h File Reference

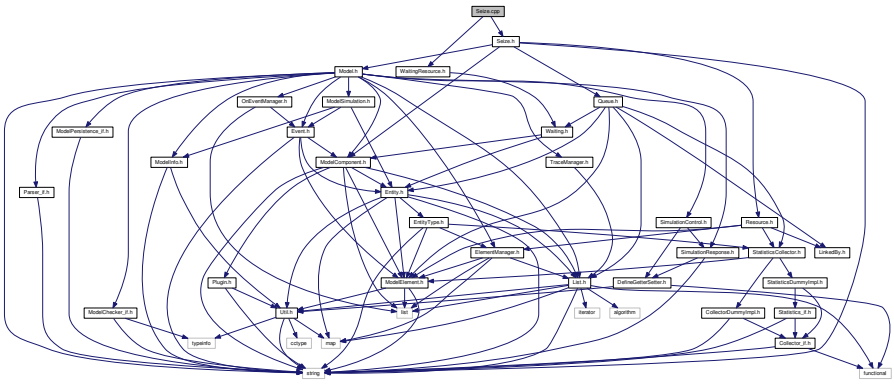
Classes

- class ScenarioExperiment_if

6.100 Seize.cpp File Reference

```
#include "Seize.h"
#include "WaitingResource.h"
```

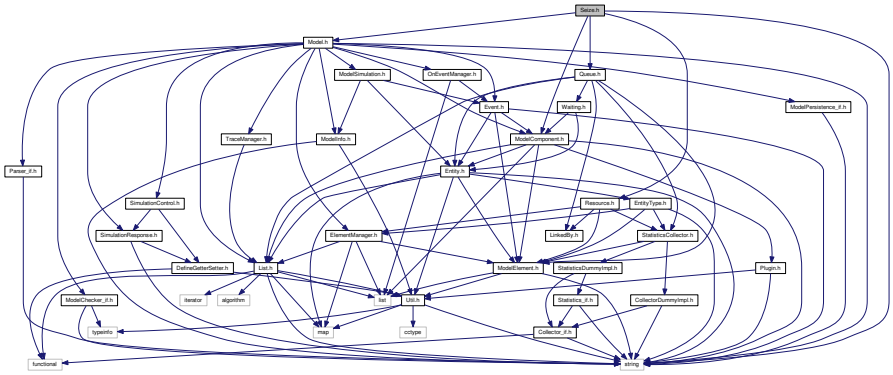
Include dependency graph for Seize.cpp:



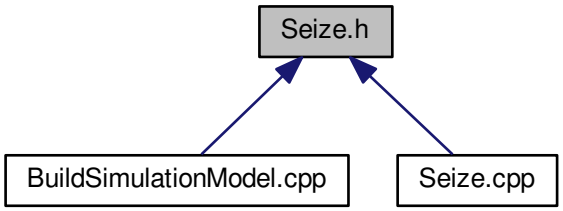
6.101 Seize.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Model.h"
#include "Resource.h"
#include "Queue.h"
```

Include dependency graph for Seize.h:



This graph shows which files directly or indirectly include this file:



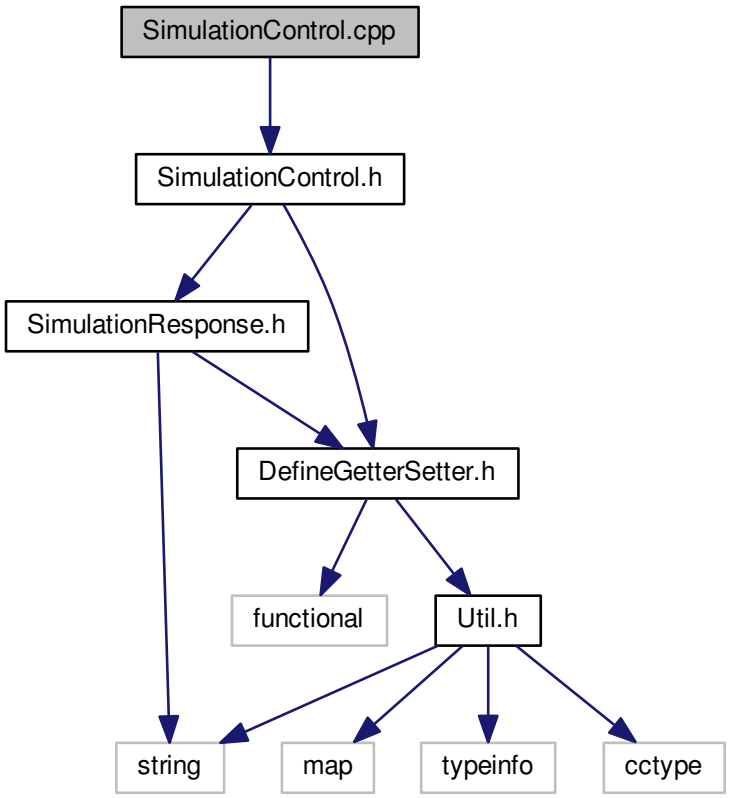
Classes

- class Seize

6.102 SimulationControl.cpp File Reference

```
#include "SimulationControl.h"
```

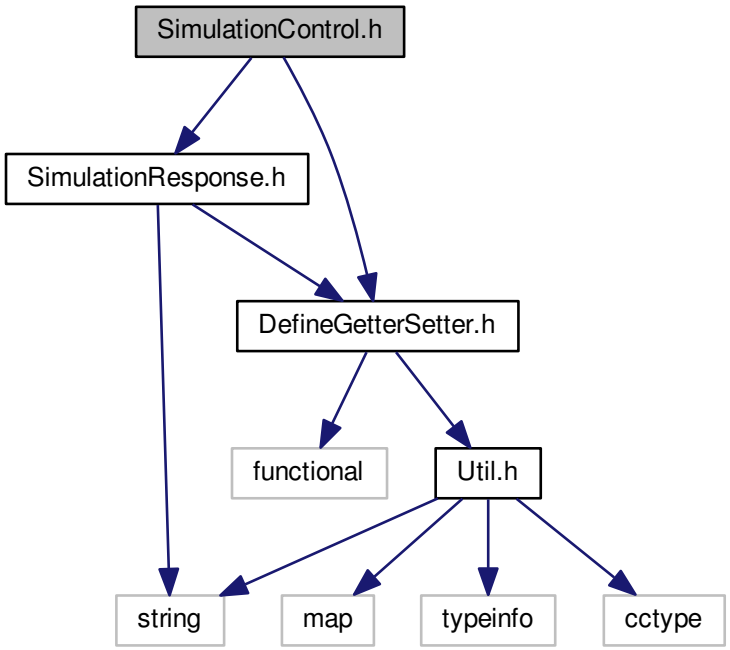
Include dependency graph for SimulationControl.cpp:



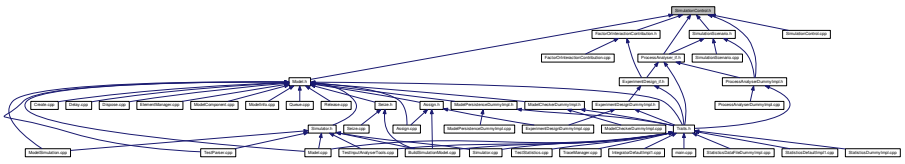
6.103 SimulationControl.h File Reference

```
#include "SimulationResponse.h"
#include "DefineGetterSetter.h"
```

Include dependency graph for SimulationControl.h:



This graph shows which files directly or indirectly include this file:



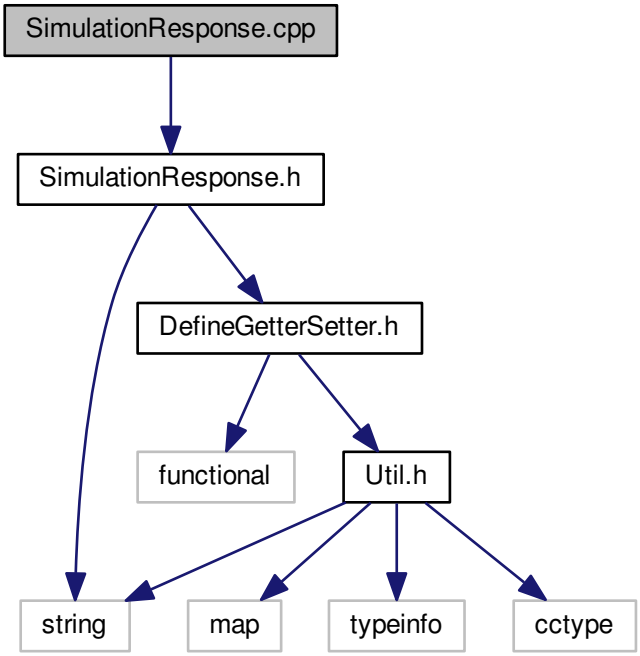
Classes

- class SimulationControl

6.104 SimulationResponse.cpp File Reference

```
#include "SimulationResponse.h"
```

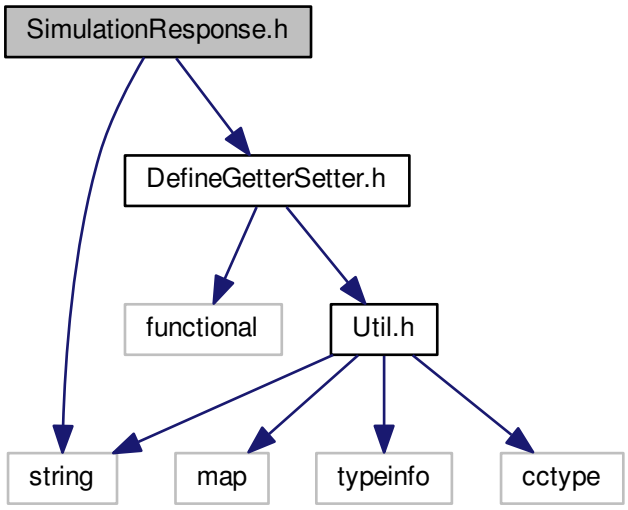
Include dependency graph for SimulationResponse.cpp:



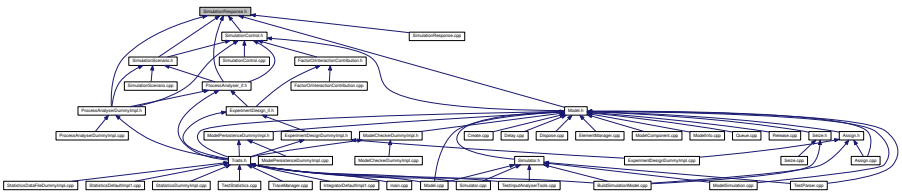
6.105 SimulationResponse.h File Reference

```
#include <string>
#include "DefineGetterSetter.h"
```

Include dependency graph for SimulationResponse.h:



This graph shows which files directly or indirectly include this file:



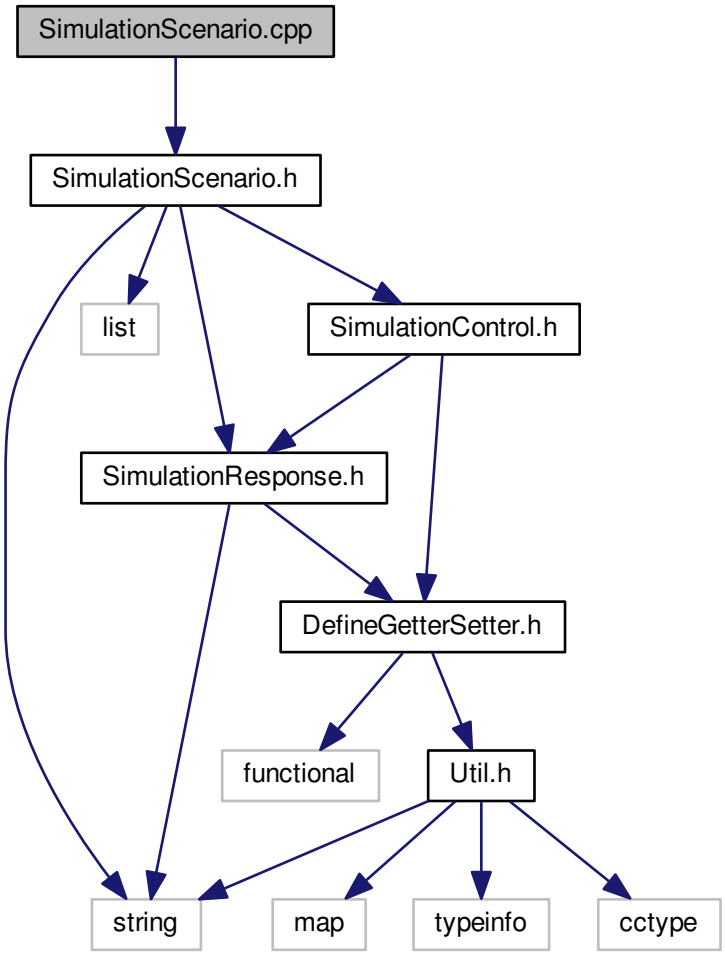
Classes

- class SimulationResponse

6.106 SimulationScenario.cpp File Reference

```
#include "SimulationScenario.h"
```

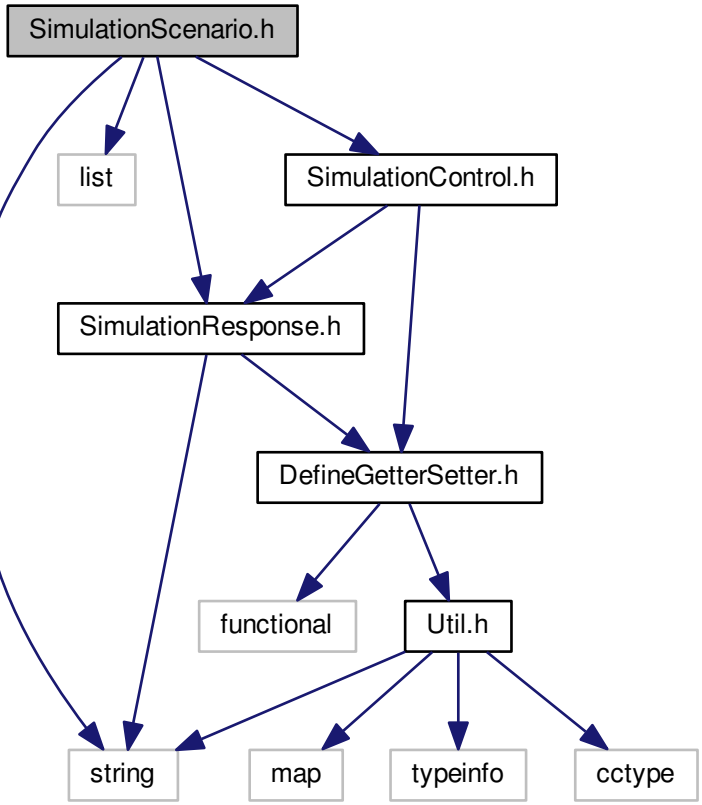
Include dependency graph for SimulationScenario.cpp:



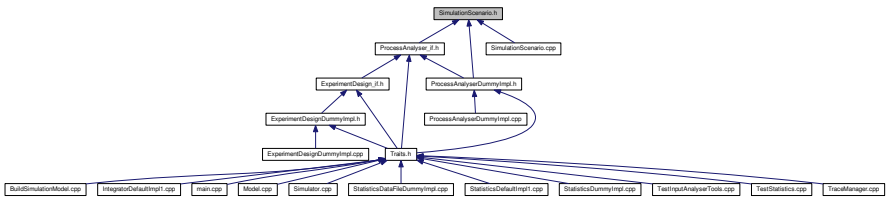
6.107 SimulationScenario.h File Reference

```
#include <string>
#include <list>
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Include dependency graph for SimulationScenario.h:



This graph shows which files directly or indirectly include this file:



Classes

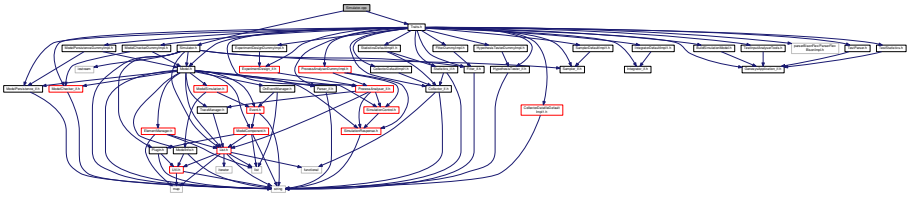
- class SimulationScenario

6.108 Simulator.cpp File Reference

```
#include "Simulator.h"
```

```
#include "Traits.h"
```

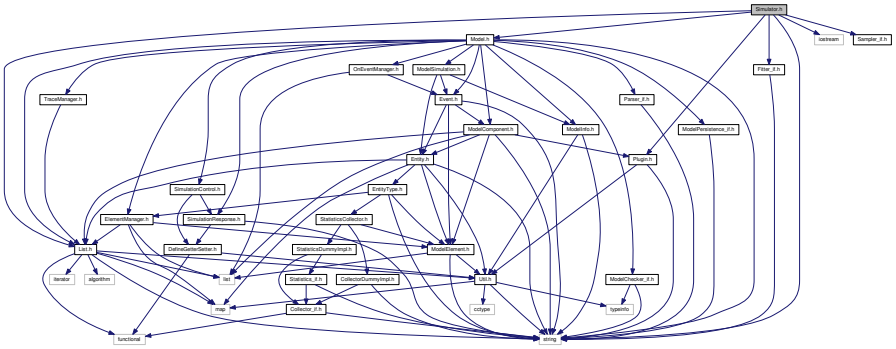
Include dependency graph for Simulator.cpp:



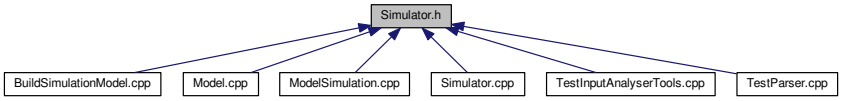
6.109 Simulator.h File Reference

```
#include <string>
#include <iostream>
#include "Model.h"
#include "Plugin.h"
#include "List.h"
#include "Fitter_if.h"
#include "Sampler_if.h"
```

Include dependency graph for Simulator.h:



This graph shows which files directly or indirectly include this file:



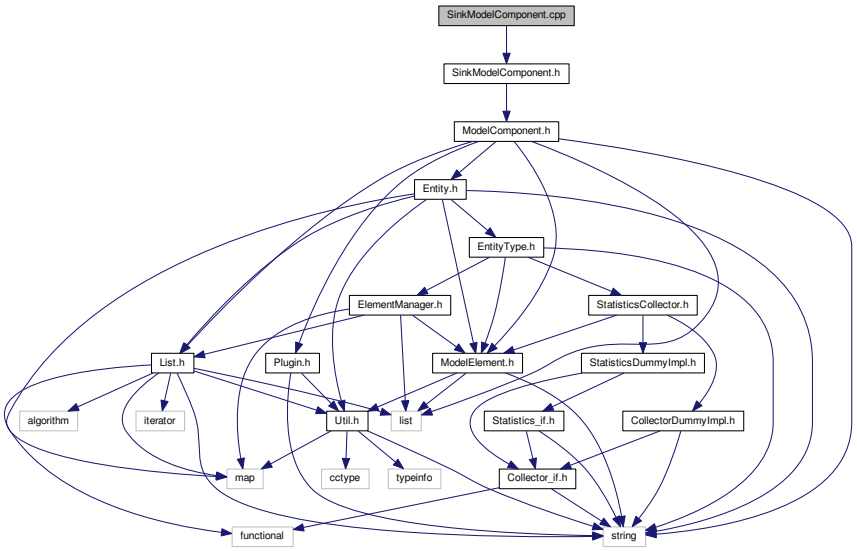
Classes

- class Simulator

6.110 SinkModelComponent.cpp File Reference

```
#include "SinkModelComponent.h"
```

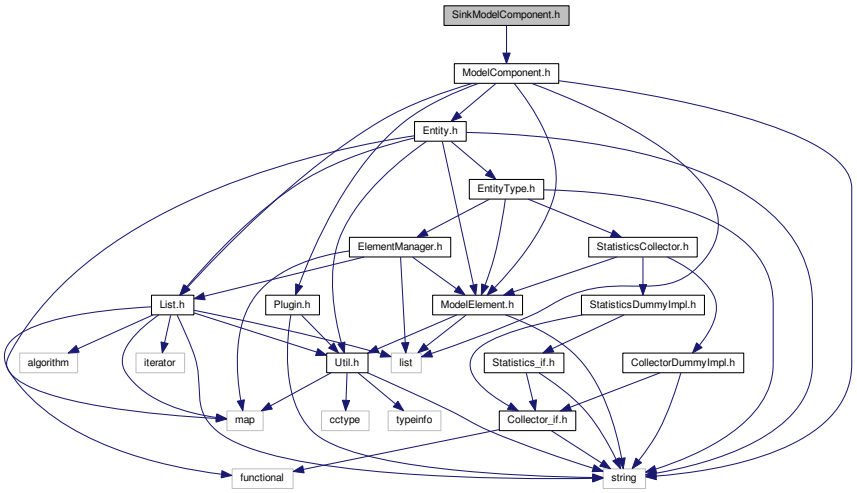
Include dependency graph for SinkModelComponent.cpp:



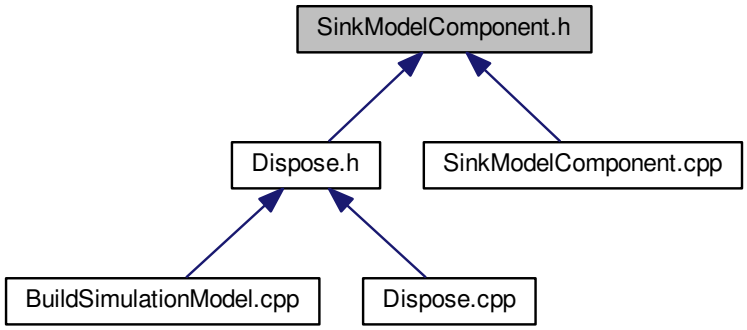
6.111 SinkModelComponent.h File Reference

```
#include "ModelComponent.h"
```

Include dependency graph for SinkModelComponent.h:



This graph shows which files directly or indirectly include this file:



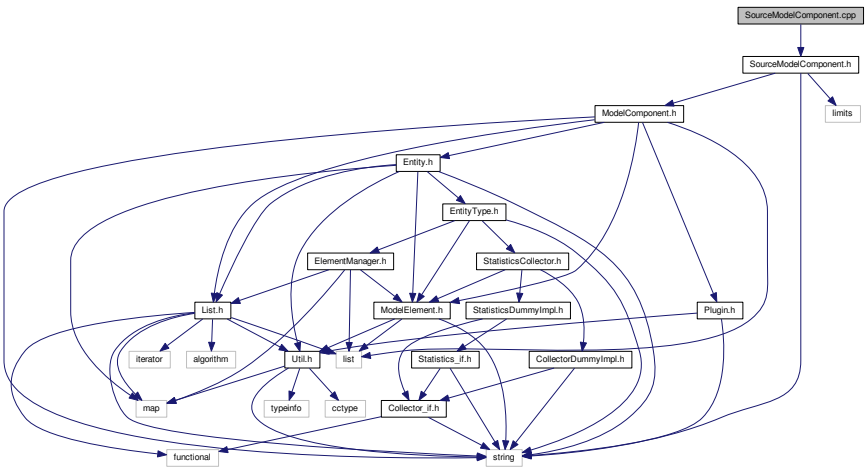
Classes

- class SinkModelComponent

6.112 SourceModelComponent.cpp File Reference

```
#include "SourceModelComponent.h"
```

Include dependency graph for SourceModelComponent.cpp:

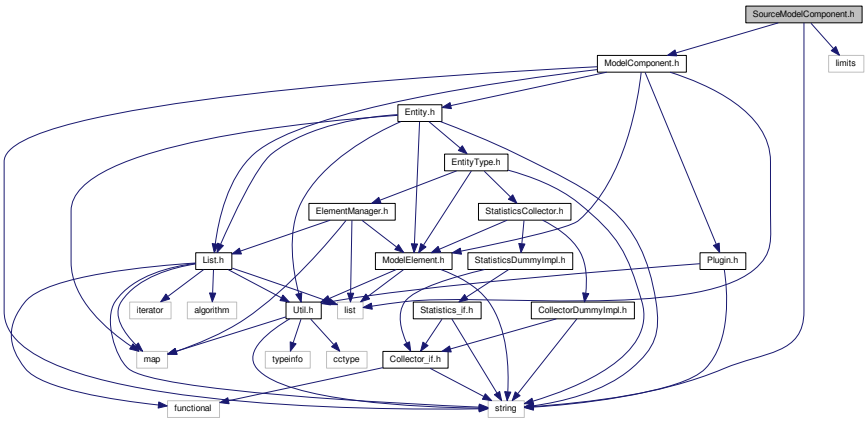


6.113 SourceModelComponent.h File Reference

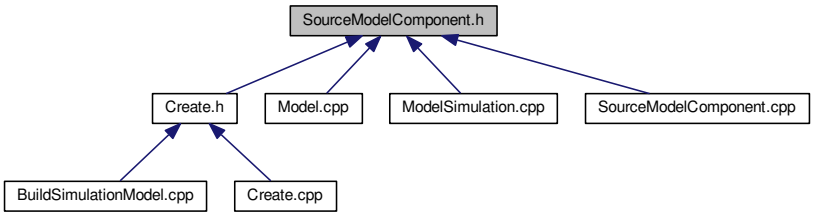
```
#include "ModelComponent.h"
```

```
#include <string>
#include <limits>
```

Include dependency graph for SourceModelComponent.h:



This graph shows which files directly or indirectly include this file:



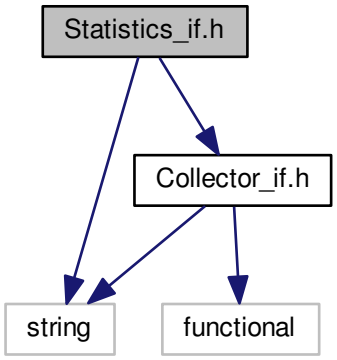
Classes

- class SourceModelComponent

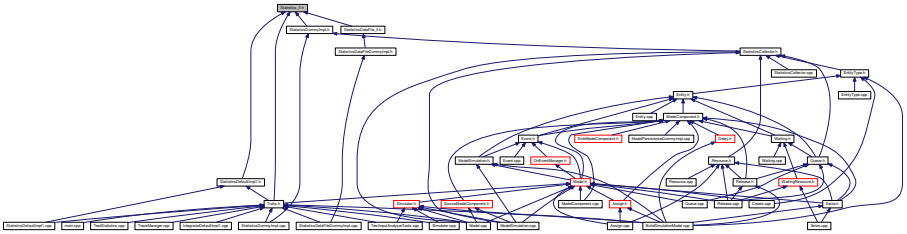
6.114 Statistics_if.h File Reference

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

Include dependency graph for Statistics_if.h:



This graph shows which files directly or indirectly include this file:



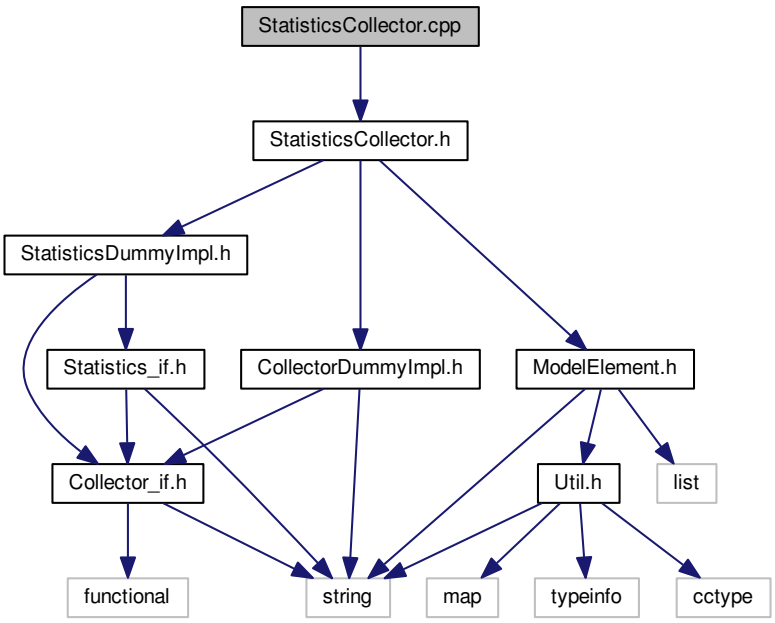
Classes

- class Statistics_if

6.115 StatisticsCollector.cpp File Reference

```
#include "StatisticsCollector.h"
```

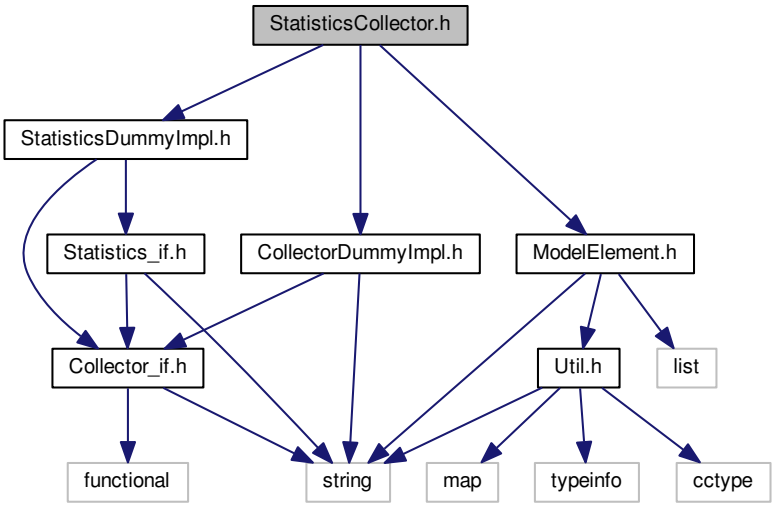
Include dependency graph for StatisticsCollector.cpp:



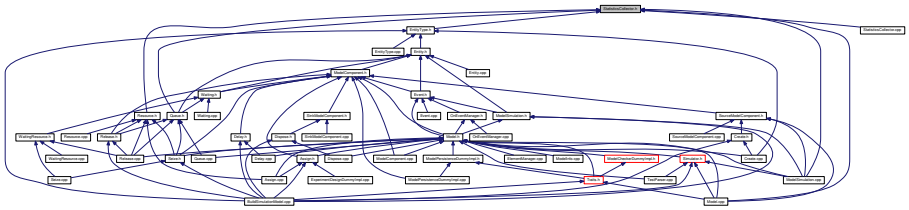
6.116 StatisticsCollector.h File Reference

```
#include "StatisticsDummyImpl.h"
#include "CollectorDummyImpl.h"
#include "ModelElement.h"
```

Include dependency graph for StatisticsCollector.h:



This graph shows which files directly or indirectly include this file:



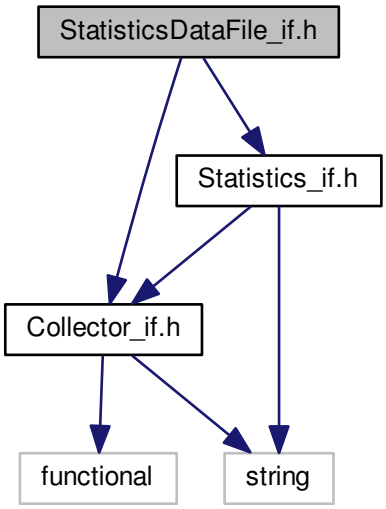
Classes

- class StatisticsCollector

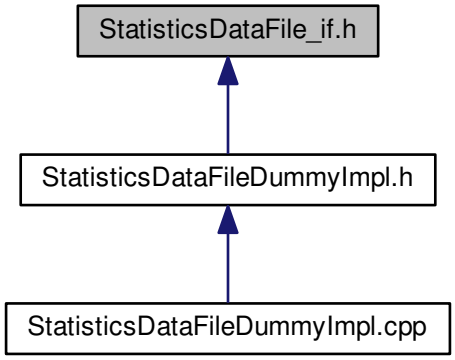
6.117 StatisticsDataFile_if.h File Reference

```
#include "Collector_if.h"
#include "Statistics_if.h"
```

Include dependency graph for StatisticsDataFile_if.h:



This graph shows which files directly or indirectly include this file:



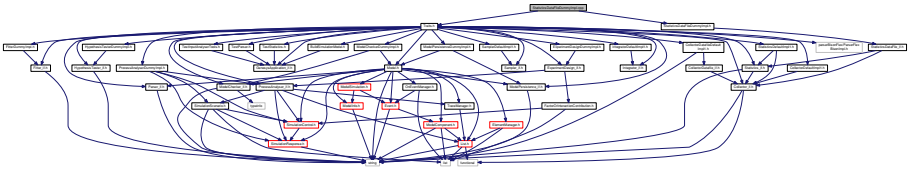
Classes

- class StatisticsDatafile_if

6.118 StatisticsDataFileDummyImpl.cpp File Reference

```
#include "StatisticsDataFileDummyImpl.h"
#include "Traits.h"
```

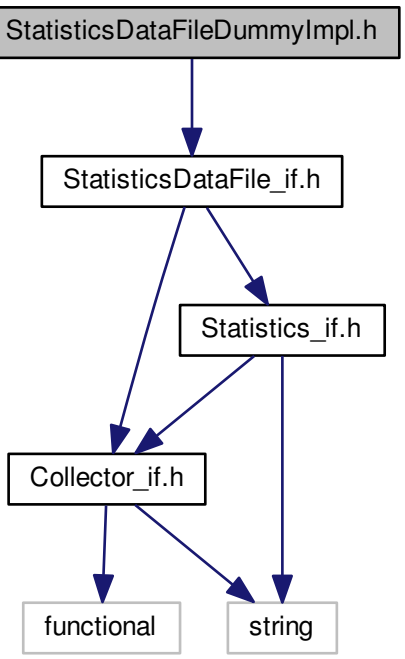
Include dependency graph for StatisticsDataFileDummyImpl.cpp:



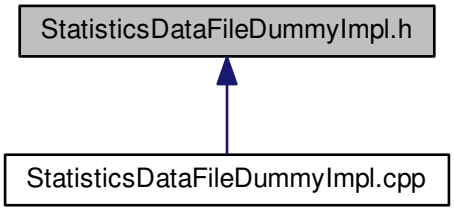
6.119 StatisticsDataFileDummyImpl.h File Reference

```
#include "StatisticsDataFile_if.h"
```

Include dependency graph for StatisticsDataFileDummyImpl.h:



This graph shows which files directly or indirectly include this file:



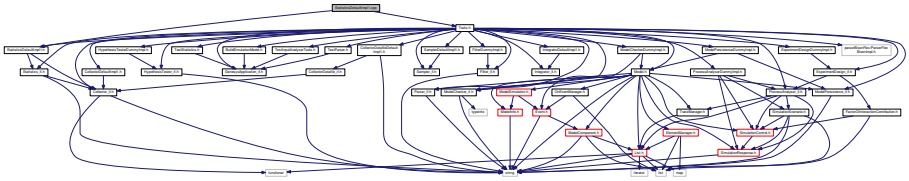
Classes

- class StatisticsDataFileDummyImpl

6.120 StatisticsDefaultImpl1.cpp File Reference

```
#include "StatisticsDefaultImpl1.h"
#include "Traits.h"
```

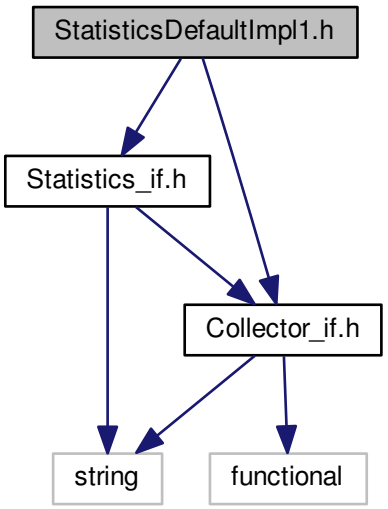
Include dependency graph for StatisticsDefaultImpl1.cpp:



6.121 StatisticsDefaultImpl1.h File Reference

```
#include "Statistics_if.h"
#include "Collector_if.h"
```

Include dependency graph for StatisticsDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:

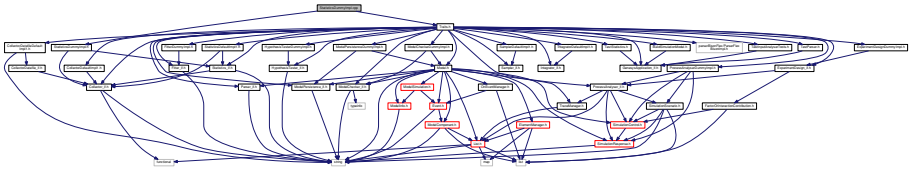
Classes

- class StatisticsDefaultImpl1

6.122 StatisticsDummyImpl.cpp File Reference

```
#include "StatisticsDummyImpl.h"
#include "Traits.h"
```

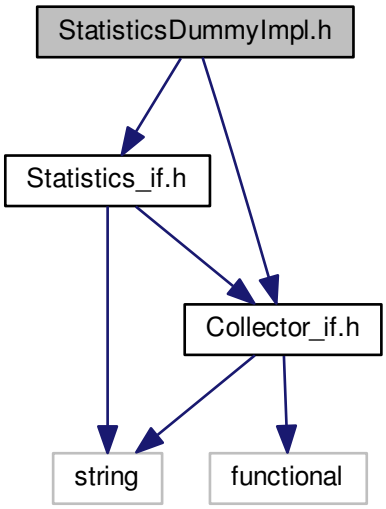
Include dependency graph for StatisticsDummyImpl.cpp:



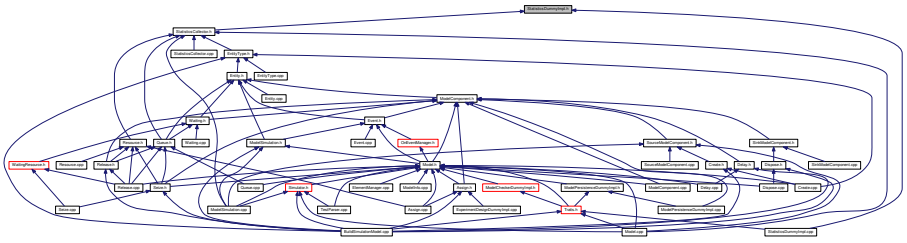
6.123 StatisticsDummyImpl.h File Reference

```
#include "Statistics_if.h"
#include "Collector_if.h"
```

Include dependency graph for StatisticsDummyImpl.h:



This graph shows which files directly or indirectly include this file:



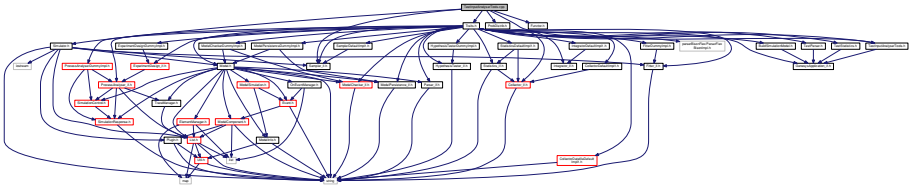
Classes

- class StatisticsDummyImpl

6.124 TestInputAnalyserTools.cpp File Reference

```
#include "TestInputAnalyserTools.h"
#include "Simulator.h"
#include "Sampler_if.h"
#include "ProbDistrib.h"
#include "Traits.h"
#include "Functor.h"
```

Include dependency graph for TestInputAnalyserTools.cpp:



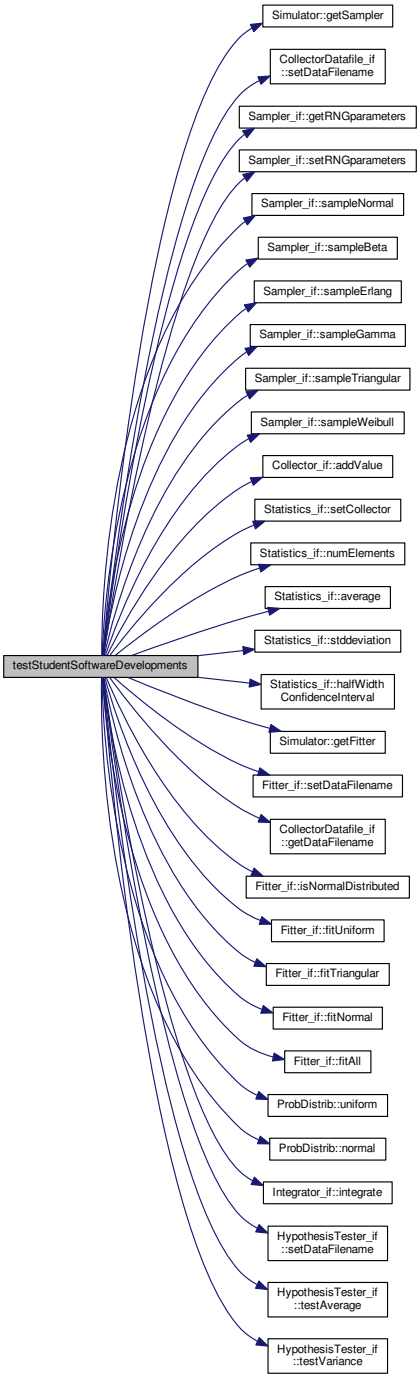
Functions

- void testStudentSoftwareDevelopments ()

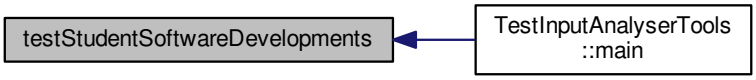
6.124.1 Function Documentation

6.124.1.1 void testStudentSoftwareDevelopments ( )

Here is the call graph for this function:



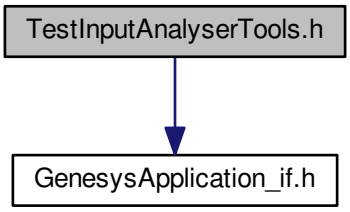
Here is the caller graph for this function:



6.125 TestInputAnalyserTools.h File Reference

```
#include "GenesysApplication_if.h"
```

Include dependency graph for TestInputAnalyserTools.h:



This graph shows which files directly or indirectly include this file:

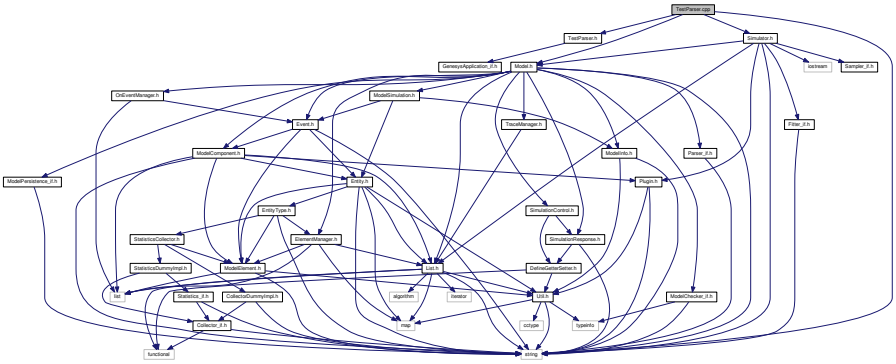
Classes

- class TestInputAnalyserTools

6.126 TestParser.cpp File Reference

```
#include <string>
#include "TestParser.h"
#include "Model.h"
#include "Simulator.h"
```

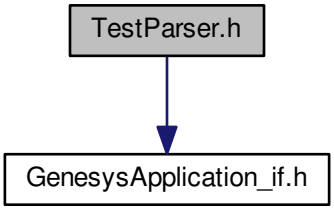
Include dependency graph for TestParser.cpp:



6.127 TestParser.h File Reference

```
#include "GenesysApplication_if.h"
```

Include dependency graph for TestParser.h:



This graph shows which files directly or indirectly include this file:

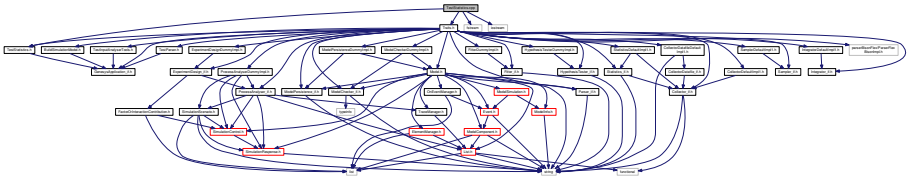
Classes

- class TestParser

6.128 TestStatistics.cpp File Reference

```
#include "TestStatistics.h"
#include <fstream>
#include <iostream>
#include "Traits.h"
```

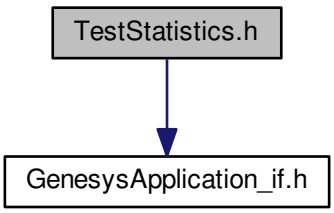
Include dependency graph for TestStatistics.cpp:



6.129 TestStatistics.h File Reference

```
#include "GenesysApplication_if.h"
```

Include dependency graph for TestStatistics.h:



This graph shows which files directly or indirectly include this file:

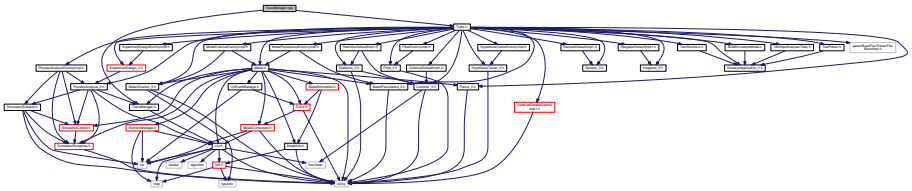
Classes

- class TestStatistics

6.130 TraceManager.cpp File Reference

```
#include "TraceManager.h"
#include "Traits.h"
```

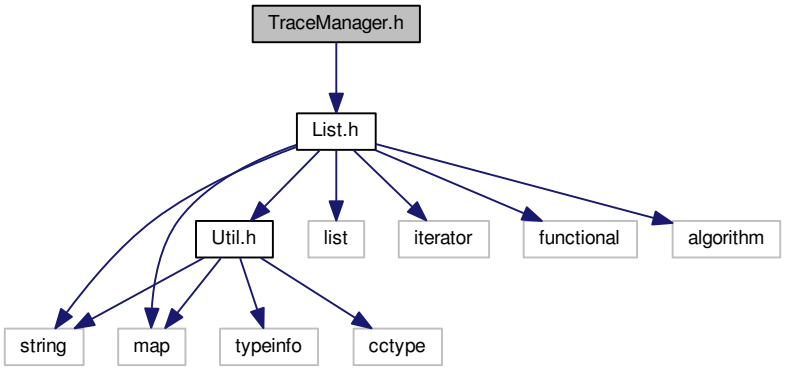
Include dependency graph for TraceManager.cpp:



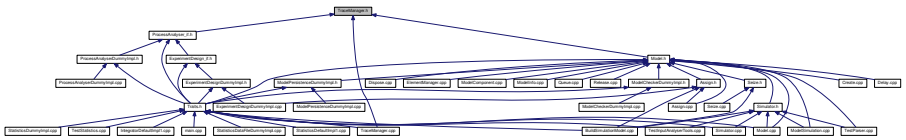
6.131 TraceManager.h File Reference

```
#include "List.h"
```

Include dependency graph for TraceManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TraceEvent
- class TraceErrorEvent
- class TraceSimulationEvent
- class TraceSimulationProcess
- class TraceManager

Typedefs

- typedef void(∗ traceListener) (TraceEvent)
- typedef void(∗ traceErrorListener) (TraceErrorEvent)
- typedef void(∗ traceSimulationListener) (TraceSimulationEvent)
- typedef void(∗ traceSimulationProcessListener) (TraceSimulationProcess)

6.131.1 Typedef Documentation

6.131.1.1 typedef void(∗ traceErrorListener) (TraceErrorEvent)

6.131.1.2 typedef void(∗ traceListener) (TraceEvent)

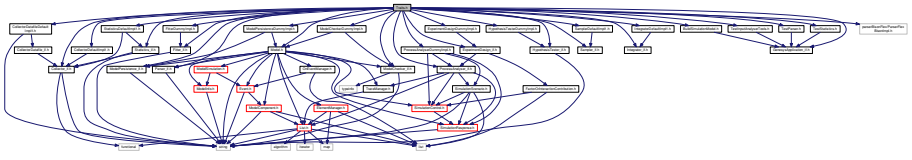
6.131.1.3 typedef void(∗ traceSimulationListener) (TraceSimulationEvent)

6.131.1.4 typedef void(∗ traceSimulationProcessListener) (TraceSimulationProcess)

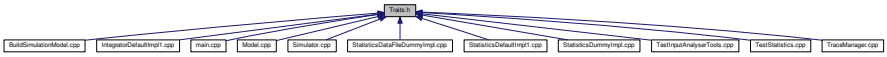
6.132 Traits.h File Reference

```
#include "Model.h"
#include "Collector_if.h"
#include "Sampler_if.h"
#include "Fitter_if.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "Statistics_if.h"
#include "Integrator_if.h"
#include "HypothesisTester_if.h"
#include "ModelPersistence_if.h"
#include "GenesysApplication_if.h"
#include "ProcessAnalyser_if.h"
#include "ExperimentDesign_if.h"
#include "BuildSimulationModel.h"
#include "TestInputAnalyserTools.h"
#include "TestParser.h"
#include "TestStatistics.h"
#include "FitterDummyImpl.h"
#include "ModelCheckerDummyImpl.h"
#include "ExperimentDesignDummyImpl.h"
#include "ProcessAnalyserDummyImpl.h"
#include "HypothesisTesterDummyImpl.h"
#include "ModelPersistenceDummyImpl.h"
#include "CollectorDefaultImpl1.h"
#include "CollectorDatafileDefaultImpl1.h"
#include "StatisticsDefaultImpl1.h"
#include "IntegratorDefaultImpl1.h"
#include "SamplerDefaultImpl1.h"
#include "parserBisonFlex/ParserFlexBisonImpl.h"
```

Include dependency graph for Traits.h:



This graph shows which files directly or indirectly include this file:



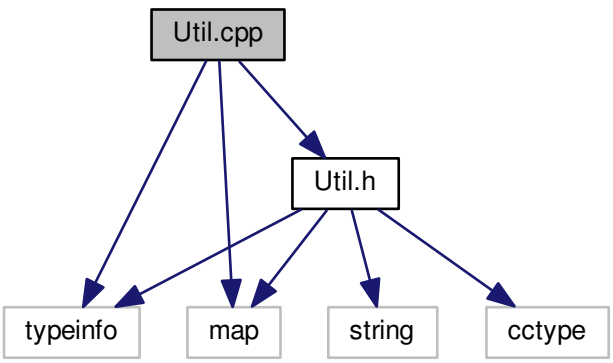
Classes

- struct Traits< T >
- struct Traits< GenesysApplication_if >
- struct Traits< Model >
- struct Traits< ModelPersistence_if >
- struct Traits< ModelComponent >
- struct Traits< ModelChecker_if >
- struct Traits< Parser_if >
- struct Traits< Collector_if >
- struct Traits< Statistics_if >
- struct Traits< Integrator_if >
- struct Traits< Sampler_if >
- struct Traits< Fitter_if >
- struct Traits< HypothesisTester_if >
- struct Traits< ExperimentDesign_if >
- struct Traits< ProcessAnalyser_if >

6.133 Util.cpp File Reference

```
#include <typeinfo>
#include <map>
#include "Util.h"
```

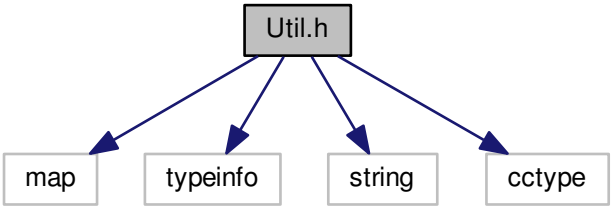
Include dependency graph for Util.cpp:



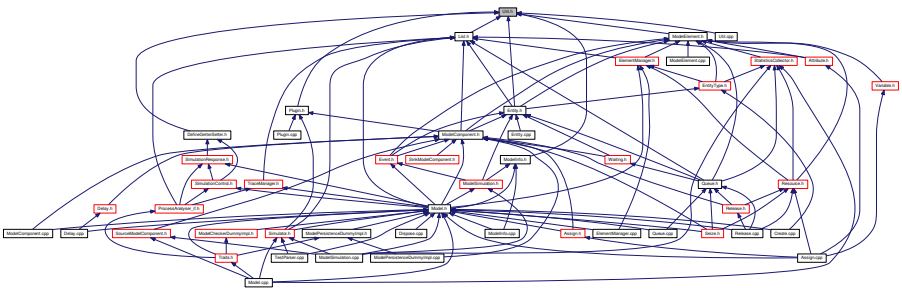
6.134 Util.h File Reference

```
#include <map>
#include <typeinfo>
#include <string>
#include <cctype>
```

Include dependency graph for Util.h:



This graph shows which files directly or indirectly include this file:



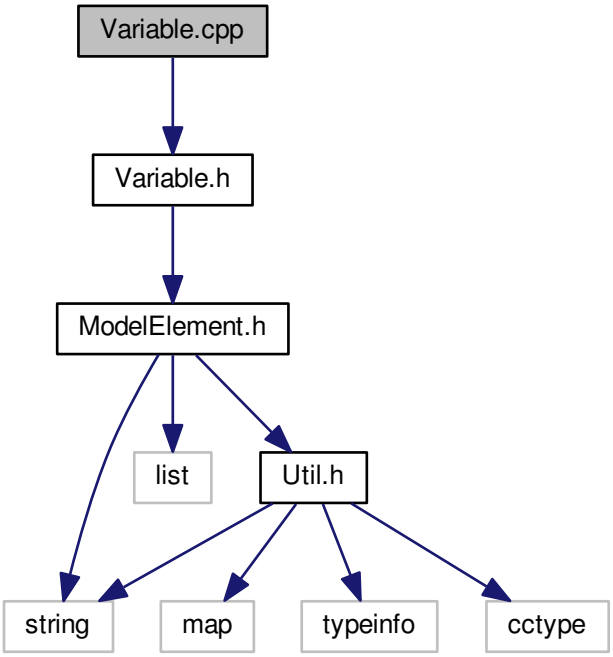
Classes

- class Util

6.135 Variable.cpp File Reference

```
#include "Variable.h"
```

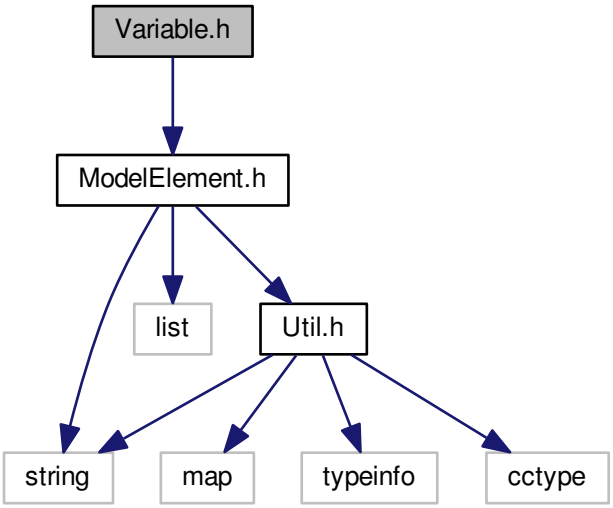
Include dependency graph for Variable.cpp:



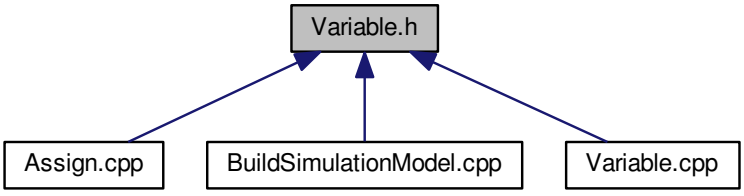
6.136 Variable.h File Reference

```
#include "ModelElement.h"
```

Include dependency graph for Variable.h:



This graph shows which files directly or indirectly include this file:



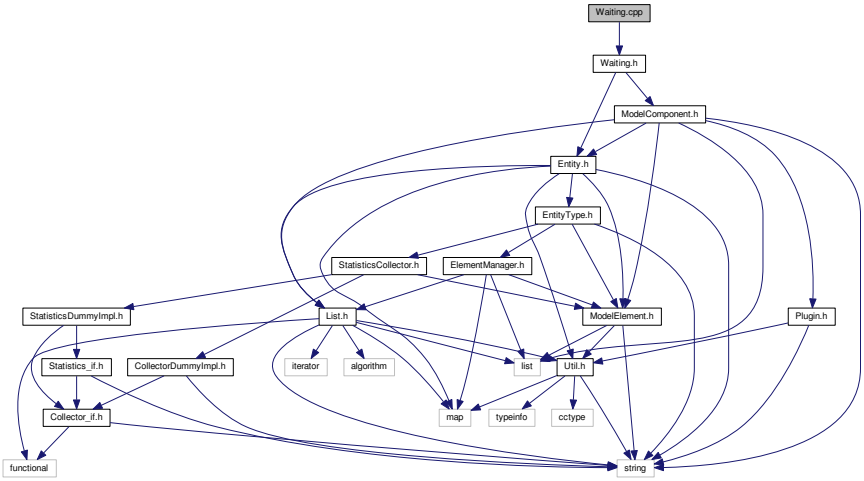
Classes

- class Variable

6.137 Waiting.cpp File Reference

```
#include "Waiting.h"
```

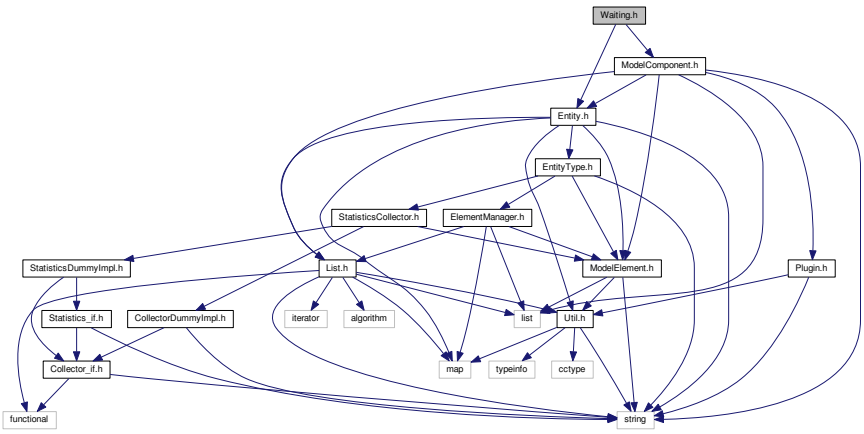
Include dependency graph for Waiting.cpp:



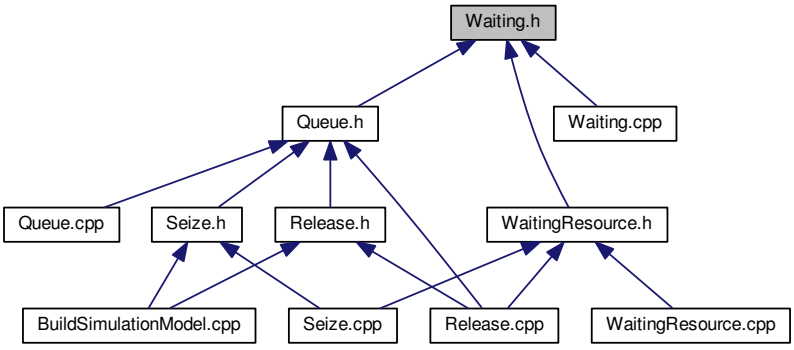
6.138 Waiting.h File Reference

```
#include "Entity.h"
#include "ModelComponent.h"
```

Include dependency graph for Waiting.h:



This graph shows which files directly or indirectly include this file:



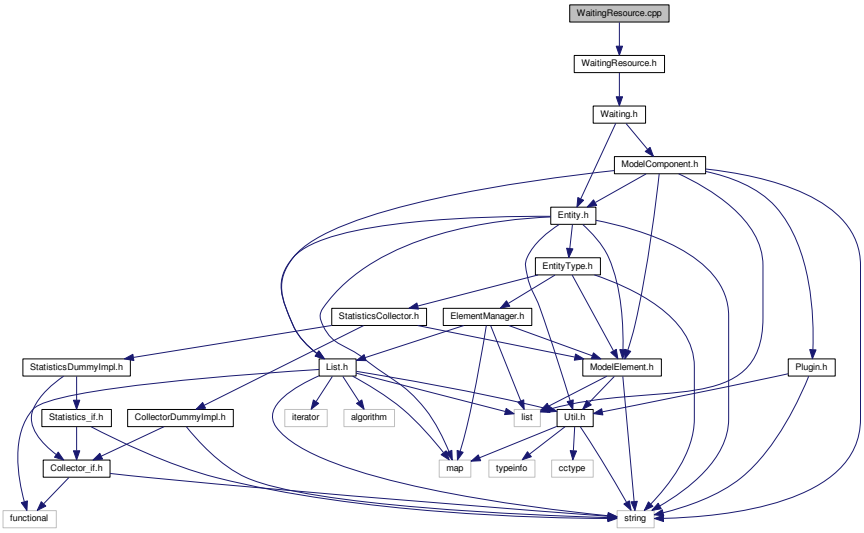
Classes

- class Waiting

6.139 WaitingResource.cpp File Reference

```
#include "WaitingResource.h"
```

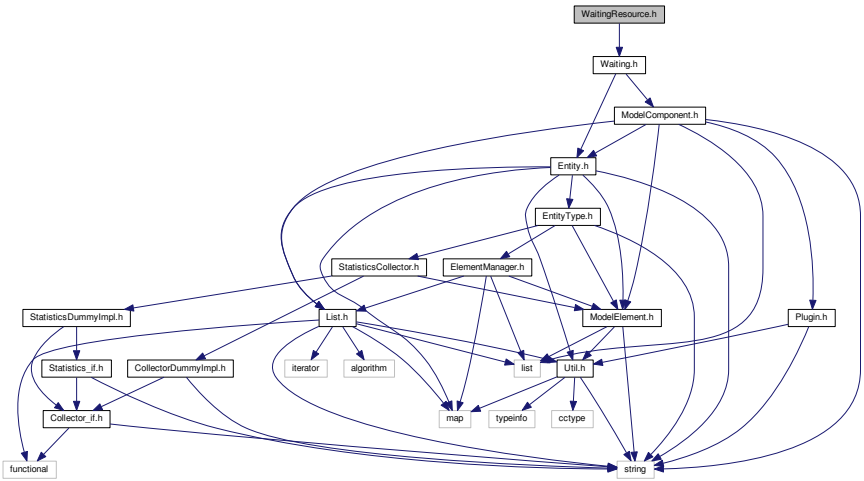
Include dependency graph for WaitingResource.cpp:



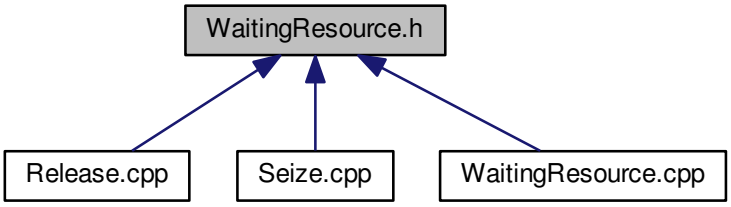
6.140 WaitingResource.h File Reference

```
#include "Waiting.h"
```

Include dependency graph for WaitingResource.h:



This graph shows which files directly or indirectly include this file:



Classes

- class WaitingResource