My Vensin

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

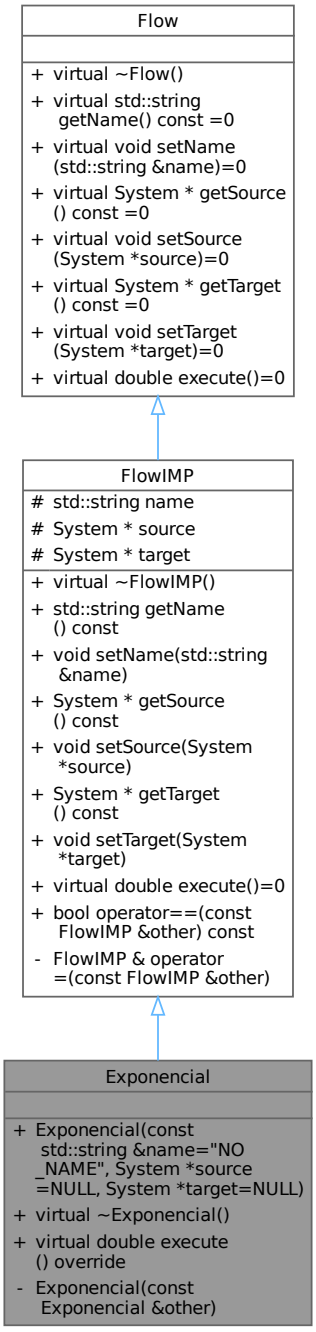
Chapter 4

Class Documentation

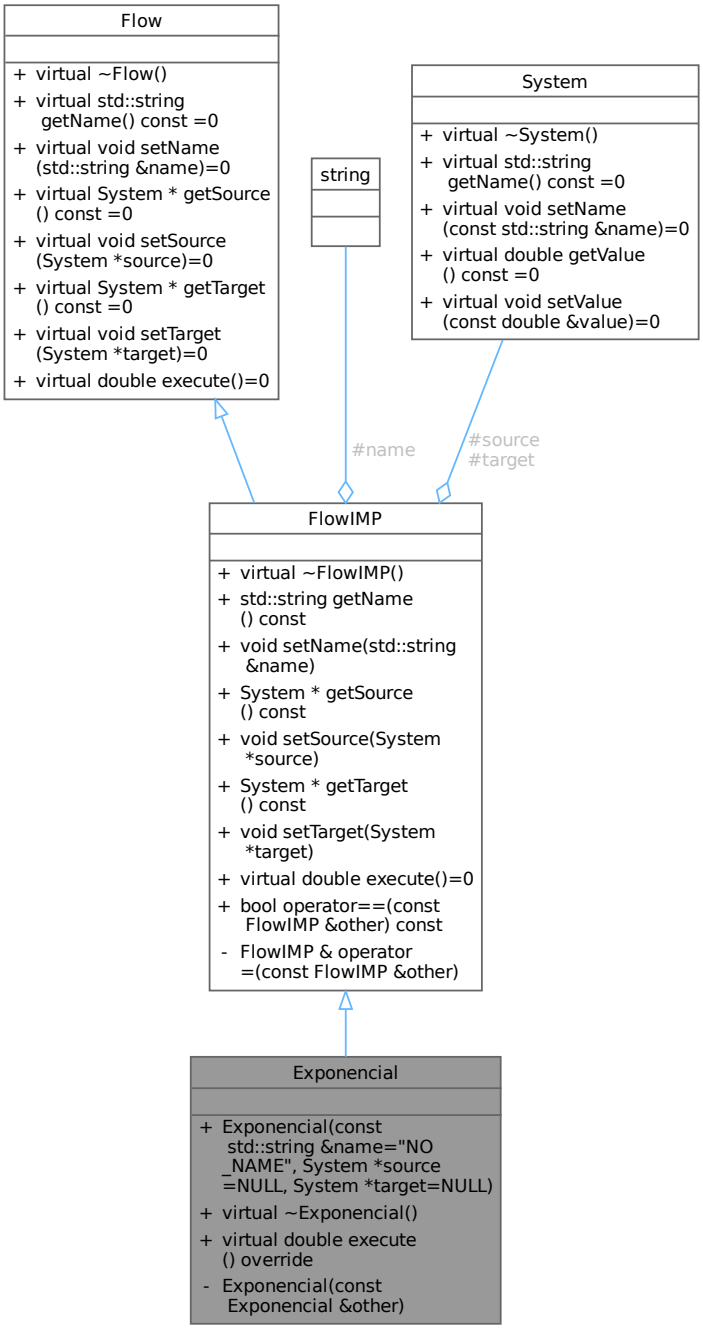
4.1 Exponencial Class Reference

`#include <Exponencial.h>`

Inheritance diagram for Exponencial:



Collaboration diagram for Exponencial:



Public Member Functions

- Exponencial (const std::string &name="NO_NAME", System *source=NULL, System *target=NULL)

  Construct a new Exponencial by name, source and target.
- virtual ~Exponencial ()

  This destructor is a virtual destructor of the Class.
- virtual double execute () override

  Pure virtual method that will contain an equation that will be executed in the flow by the model.

Public Member Functions inherited from FlowIMP

- virtual ~FlowIMP ()

  This destructor is a virtual destructor of the class.
- std::string getName () const

  This method returns the name of a flow.
- void setName (std::string &name)

  This method assigns a string to the name of a flow obj.
- System * getSource () const

  This method returns the source system poiter.
- void setSource (System *source)

  This method assigns a system poiter to the source of a flow obj.
- System * getTarget () const

  This method returns the target system poiter.
- void setTarget (System *target)

  This method assigns a system poiter to the target of a flow obj.
- bool operator== (const FlowIMP &other) const

  This method is overloading the '==' operator, compare two flows objs.

Public Member Functions inherited from Flow

- virtual ~Flow ()

  This destructor is a virtual destructor of the class.

Private Member Functions

- Exponencial (const Exponencial &other)

  Construct a new Exponencial by a obj.

Additional Inherited Members

Protected Attributes inherited from FlowIMP

- std::string name
- System * source
- System * target

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Exponencial() [1/2]

```
Exponencial::Exponencial (
            const Exponencial & other )  [private]
```

Construct a new Exponencial by a obj.

Parameters

other	Exponencial obj

```
00011                                                {
00012     this->name = other.name;
00013     this->source = other.source;
00014     this->target = other.target;
00015 }
```

References FlowIMP::name, FlowIMP::source, and FlowIMP::target.

4.1.1.2 Exponencial() [2/2]

```
Exponencial::Exponencial (
            const std::string & name = "NO_NAME",
            System * source = NULL,
            System * target = NULL )
```

Construct a new Exponencial by name, source and target.

Parameters

name	string with default value "NO_NAME"
source	System pointer with default value NULL
target	System pointer with default value NULL

```
00004                                                                                {
00005     this->name = name;
00006     this->source = source;
00007     this->target = target;
00008 }
```

References FlowIMP::name, FlowIMP::source, and FlowIMP::target.

4.1.1.3 ~Exponencial()

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

This destructor is a virtual destructor of the Class.

```
00018 {}
```

4.1.2 Member Function Documentation

4.1.2.1 execute()

```
double Exponencial::execute ( )  [override], [virtual]
```

Pure virtual method that will contain an equation that will be executed in the flow by the model.

Returns

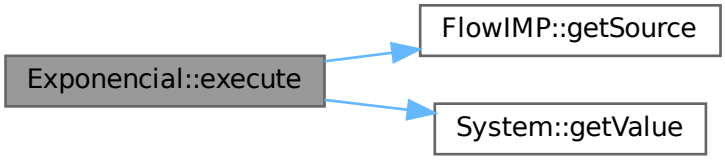
double

Implements FlowIMP.

```
00020                               {
00021     return getSource()->getValue() * 0.01;
00022 }
```

References FlowIMP::getSource(), and System::getValue().

Here is the call graph for this function:



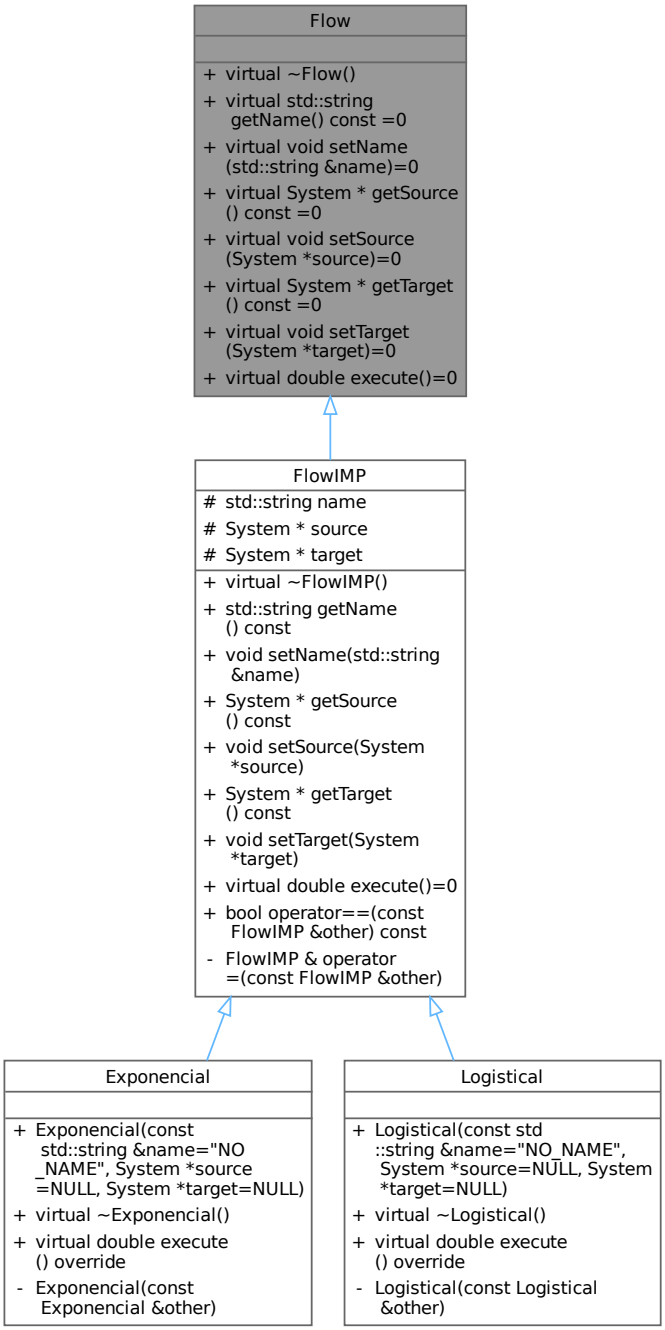
The documentation for this class was generated from the following files:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_↩ tests/src/Exponencial.h
- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_↩ tests/src/Exponencial.cpp

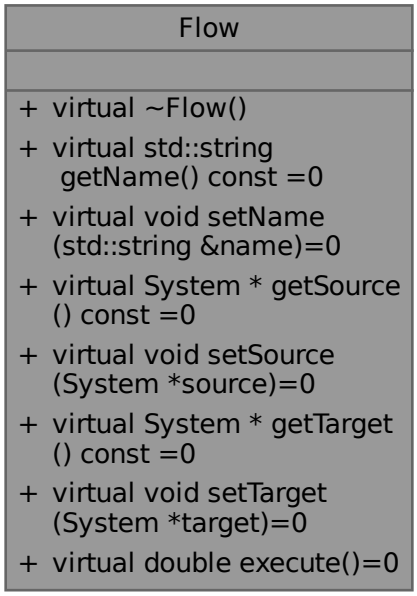
4.2 Flow Class Reference

`#include <Flow.h>`

Inheritance diagram for Flow:



Collaboration diagram for Flow:



Public Member Functions

- virtual ~Flow ()

  This destructor is a virtual destructor of the class.
- virtual std::string getName () const =0

  This method returns the name of a flow.
- virtual void setName (std::string &name)=0

  This method assigns a string to the name of a flow obj.
- virtual System * getSource () const =0

  This method returns the source system poiter.
- virtual void setSource (System *source)=0

  This method assigns a system poiter to the source of a flow obj.
- virtual System * getTarget () const =0

  This method returns the target system poiter.
- virtual void setTarget (System *target)=0

  This method assigns a system poiter to the target of a flow obj.
- virtual double execute ()=0

  Pure virtual method that will be inherited by subclasses created by the user, this one will contain an equation that will be executed in the flow by the model.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 ~Flow()

```
virtual Flow::~Flow ( )  [inline], [virtual]
```

This destructor is a virtual destructor of the class.

```
00024 {};
```

4.2.2 Member Function Documentation

4.2.2.1 execute()

```
virtual double Flow::execute ( )  [pure virtual]
```

Pure virtual method that will be inherited by subclasses created by the user, this one will contain an equation that will be executed in the flow by the model.

Returns

double

Implemented in Exponencial, Logistical, and FlowIMP.

4.2.2.2 getName()

```
virtual std::string Flow::getName ( ) const  [pure virtual]
```

This method returns the name of a flow.

Returns

a string containing the name is returned

Implemented in FlowIMP.

4.2.2.3 getSource()

```
virtual System * Flow::getSource ( ) const  [pure virtual]
```

This method returns the source system poiter.

Returns

a system poiter containing the source memory address is returned

Implemented in FlowIMP.

4.2.2.4 getTarget()

```
virtual System * Flow::getTarget ( ) const  [pure virtual]
```

This method returns the target system poiter.

Returns

a system poiter containing the target memory address is returned

Implemented in FlowIMP.

4.2.2.5 setName()

```
virtual void Flow::setName (
            std::string & name )  [pure virtual]
```

This method assigns a string to the name of a flow obj.

Parameters

name	string must be passed to the method

Implemented in FlowIMP.

4.2.2.6 setSource()

```
virtual void Flow::setSource (
            System * source )  [pure virtual]
```

This method assigns a system poiter to the source of a flow obj.

Parameters

source	system poiter must be passed to the method

Implemented in FlowIMP.

4.2.2.7 setTarget()

```
virtual void Flow::setTarget (
            System * target )  [pure virtual]
```

This method assigns a system poiter to the target of a flow obj.

Parameters

target	system poiter must be passed to the method

Implemented in FlowIMP.

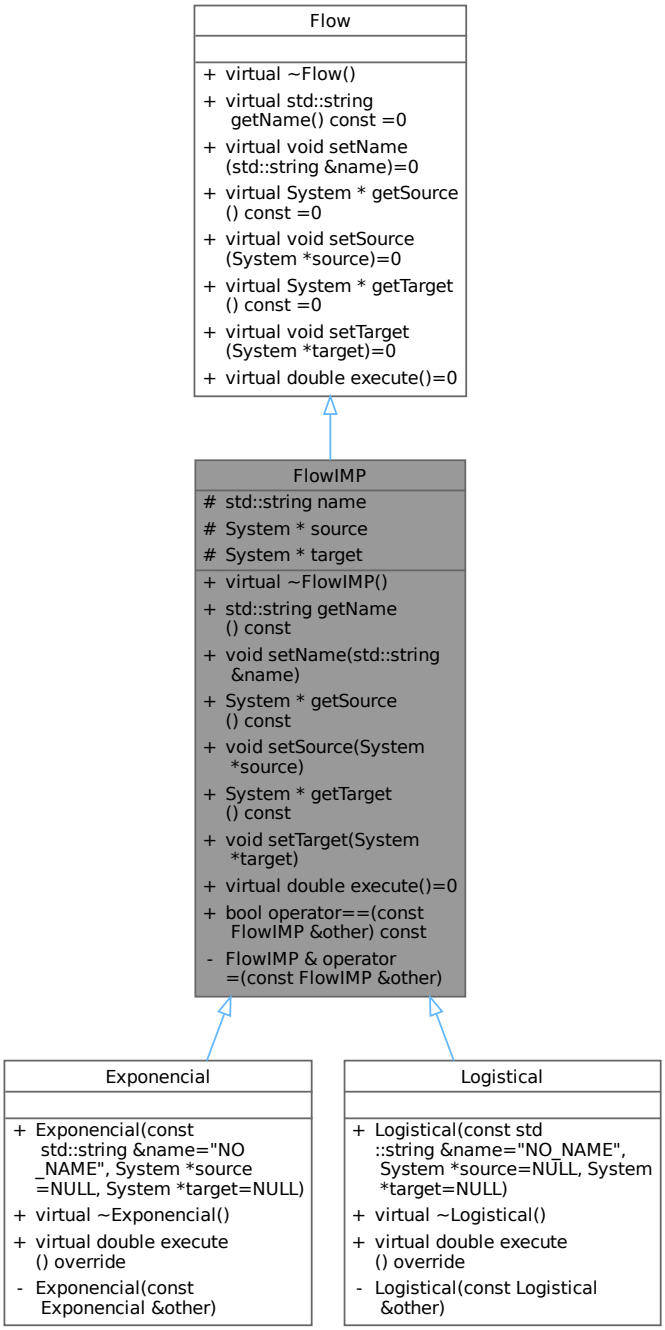
The documentation for this class was generated from the following file:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.h

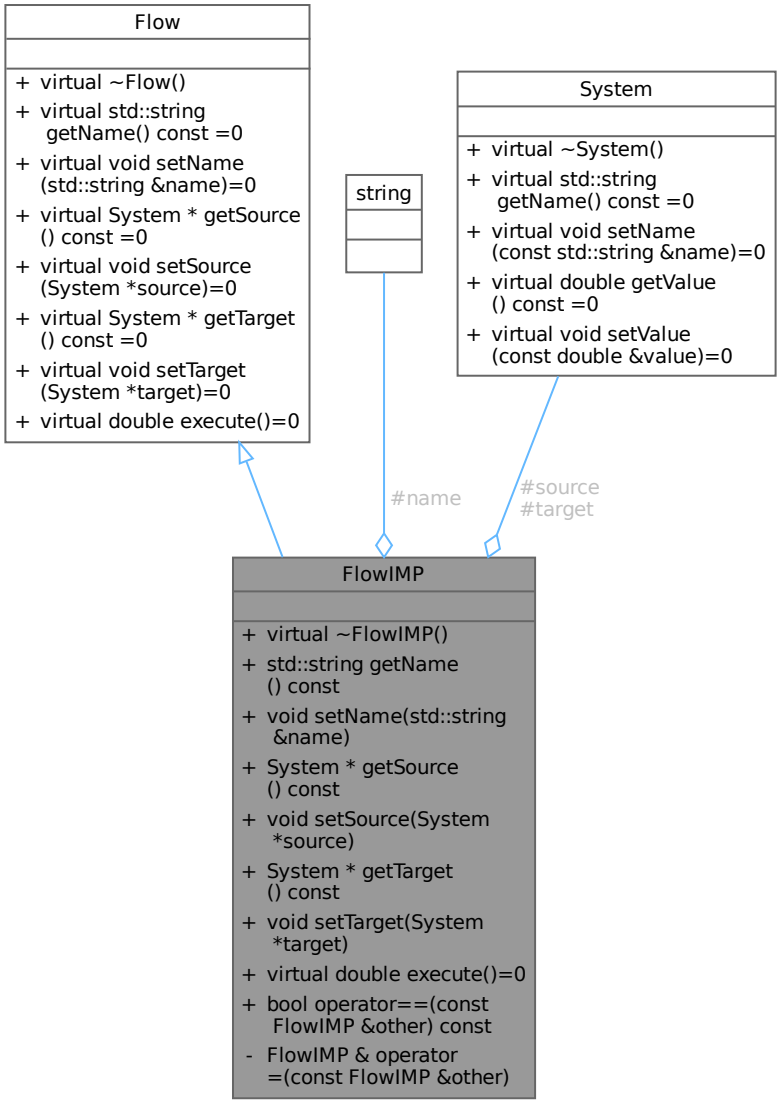
4.3 FlowIMP Class Reference

```
#include <FlowIMP.h>
```

Inheritance diagram for FlowIMP:



Collaboration diagram for FlowIMP:



Public Member Functions

- virtual ~FlowIMP ()

  This destructor is a virtual destructor of the class.
- std::string getName () const

  This method returns the name of a flow.
- void setName (std::string &name)

  This method assigns a string to the name of a flow obj.
- System * getSource () const

  This method returns the source system poiter.
- void setSource (System *source)

  This method assigns a system poiter to the source of a flow obj.

- System * getTarget () const

  This method returns the target system poiter.
- void setTarget (System *target)

  This method assigns a system poiter to the target of a flow obj.
- virtual double execute ()=0

  Pure virtual method that will be inherited by subclasses created by the user, this one will contain an equation that will be executed in the flow by the model.
- bool operator== (const FlowIMP &other) const

  This method is overloading the '==' operator, compare two flows objs.

Public Member Functions inherited from Flow

- virtual ~Flow ()

  This destructor is a virtual destructor of the class.

Protected Attributes

- std::string name
- System * source
- System * target

Private Member Functions

- FlowIMP & operator= (const FlowIMP &other)

  This method is overloading the '=' operator, "cloning" from one flow to another.

Friends

- std::ostream & operator<< (std::ostream &out, const FlowIMP &obj)

  This method is overloading the '<<' operator, print the flow obj info.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 ~FlowIMP()

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

This destructor is a virtual destructor of the class.

```
00004 {}
```

4.3.2 Member Function Documentation

4.3.2.1 execute()

```
virtual double FlowIMP::execute ( )  [pure virtual]
```

Pure virtual method that will be inherited by subclasses created by the user, this one will contain an equation that will be executed in the flow by the model.

Returns

double

Implements Flow.

Implemented in Exponencial, and Logistical.

4.3.2.2 getName()

```
std::string FlowIMP::getName ( ) const  [virtual]
```

This method returns the name of a flow.

Returns

a string containing the name is returned

Implements Flow.

```
00008 { return name; }
```

References name.

4.3.2.3 getSource()

```
System * FlowIMP::getSource ( ) const  [virtual]
```

This method returns the source system poiter.

Returns

a system poiter containing the source memory address is returned

Implements Flow.

```
00011 { return source; }
```

References source.

Referenced by Exponencial::execute().

Here is the caller graph for this function:

4.3.2.4 getTarget()

```
System * FlowIMP::getTarget ( ) const  [virtual]
```

This method returns the target system poiter.

Returns

a system poiter containing the target memory address is returned

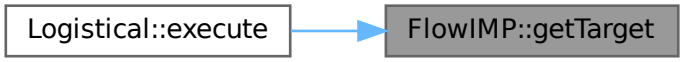
Implements Flow.

```
00014 { return target; }
```

References target.

Referenced by Logistical::execute().

Here is the caller graph for this function:



4.3.2.5 operator=()

```
FlowIMP & FlowIMP::operator= (
            const FlowIMP & other )  [private]
```

This method is overloading the '=' operator, "cloning" from one flow to another.

Parameters

other	flow obj to be cloned must be passed

Returns

A flow is returned that is a clone of what was passed to the method

```
00019                                                {
00020     if(other == *this) return *this;
00021     name = other.name;
00022     source = other.source;
00023     target = other.target;
00024     return *this;
00025 }
```

References name, source, and target.

4.3.2.6 operator==()

```
bool FlowIMP::operator== (
            const FlowIMP & other ) const
```

This method is overloading the '==' operator, compare two flows objs.

Parameters

other	flow obj to be compare must be passed

Returns

A bool is returned, true if they are equal and false if not

```
00028                                                   {
00029     return (name == other.name && source == other.source && target == other.target);
00030 }
```

References name, source, and target.

4.3.2.7 setName()

```
void FlowIMP::setName (
            std::string & name )  [virtual]
```

This method assigns a string to the name of a flow obj.

Parameters

name	string must be passed to the method

Implements Flow.

```
00009 { this->name = name; }
```

References name.

4.3.2.8 setSource()

```
void FlowIMP::setSource (
            System * source )  [virtual]
```

This method assigns a system poiter to the source of a flow obj.

Parameters

source	system poiter must be passed to the method

Implements Flow.

```
00012 { this->source = source; }
```

References source.

4.3.2.9 setTarget()

```
void FlowIMP::setTarget (
            System * target )  [virtual]
```

This method assigns a system poiter to the target of a flow obj.

Parameters

target	system poiter must be passed to the method

Implements Flow.

```
00015 { this->target = target; }
```

References target.

4.3.3 Friends And Related Symbol Documentation

4.3.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const FlowIMP & obj )  [friend]
```

This method is overloading the '<<' operator, print the flow obj info.

Parameters

out	is a ostream obj
obj	is a flow obj

Returns

a ostream obj to print the obj info

```
00032                                                          {
00033     out « "(Flow) Name: " « obj.name « " - "
00034         « obj.source->getName() « " -----> " « obj.target->getName();
00035     return out;
00036 }
```

4.3.4 Member Data Documentation

4.3.4.1 name

```
std::string FlowIMP::name  [protected]
```

Name string attribute.

Referenced by Exponencial::Exponencial(), Exponencial::Exponencial(), getName(), Logistical::Logistical(), Logistical::Logistical(), operator=(), operator==(), and setName().

4.3.4.2 source

```
System* FlowIMP::source  [protected]
```

Source system pointer attribute.

Referenced by Exponencial::Exponencial(), Exponencial::Exponencial(), getSource(), Logistical::Logistical(), Logistical::Logistical(), operator=(), operator==(), and setSource().

4.3.4.3 target

```
System* FlowIMP::target  [protected]
```

Target system pointer attribute.

Referenced by Exponencial::Exponencial(), Exponencial::Exponencial(), getTarget(), Logistical::Logistical(), Logistical::Logistical(), operator=(), operator==(), and setTarget().

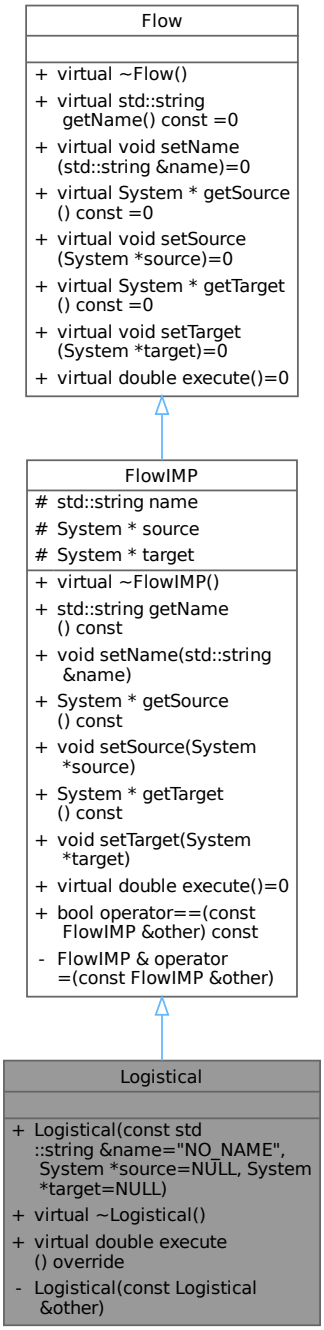
The documentation for this class was generated from the following files:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/FlowIMP.h
- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/FlowIMP.cpp

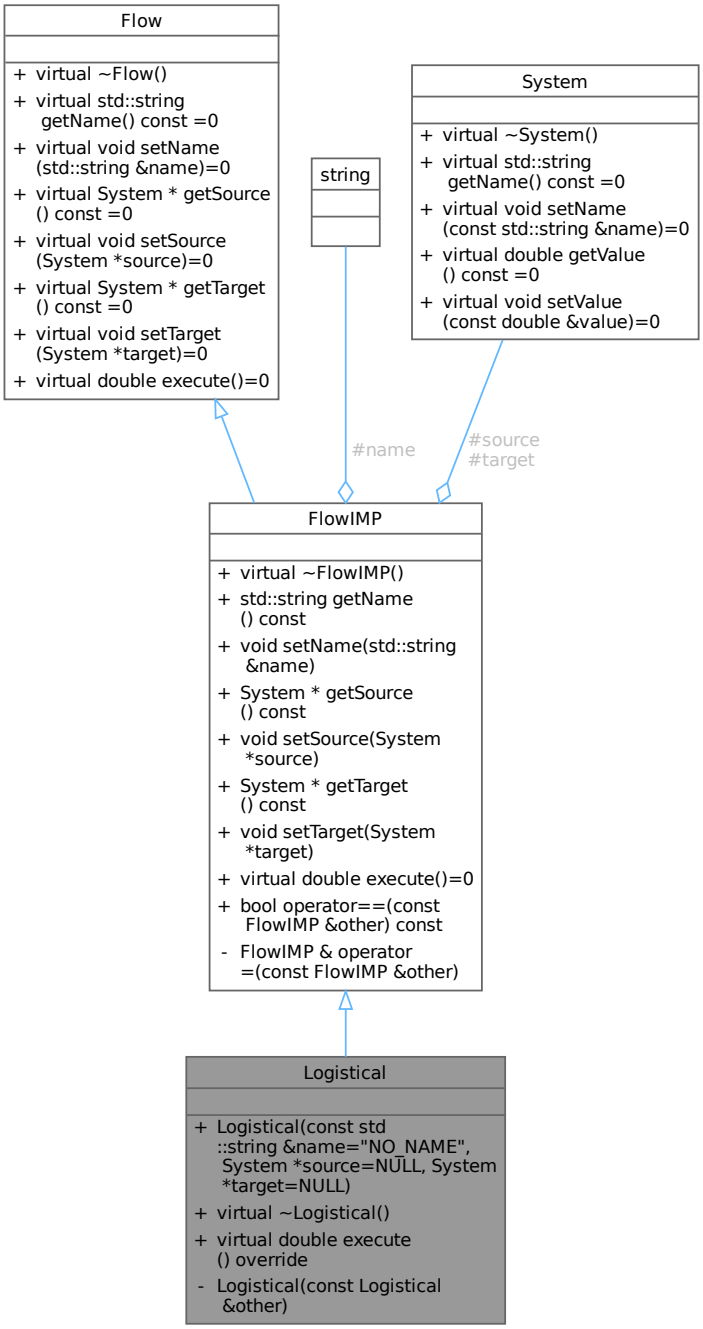
4.4 Logistical Class Reference

```
#include <Logistical.h>
```

Inheritance diagram for Logistical:



Collaboration diagram for Logistical:



Public Member Functions

- Logistical (const std::string &name="NO_NAME", System *source=NULL, System *target=NULL)

  Construct a new Logistical by name, source and target.
- virtual ~Logistical ()

  This destructor is a virtual destructor of the Class.
- virtual double execute () override

  Pure virtual method that will contain an equation that will be executed in the flow by the model.

Public Member Functions inherited from FlowIMP

- virtual ~FlowIMP ()

  This destructor is a virtual destructor of the class.
- std::string getName () const

  This method returns the name of a flow.
- void setName (std::string &name)

  This method assigns a string to the name of a flow obj.
- System * getSource () const

  This method returns the source system poiter.
- void setSource (System *source)

  This method assigns a system poiter to the source of a flow obj.
- System * getTarget () const

  This method returns the target system poiter.
- void setTarget (System *target)

  This method assigns a system poiter to the target of a flow obj.
- bool operator== (const FlowIMP &other) const

  This method is overloading the '==' operator, compare two flows objs.

Public Member Functions inherited from Flow

- virtual ~Flow ()

  This destructor is a virtual destructor of the class.

Private Member Functions

- Logistical (const Logistical &other)

  Construct a new Logistical by a obj.

Additional Inherited Members

Protected Attributes inherited from FlowIMP

- std::string name
- System * source
- System * target

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Logistical() [1/2]

```
Logistical::Logistical (
            const Logistical & other )  [private]
```

Construct a new Logistical by a obj.

Parameters

other	Logistical obj

```
00011                                                {
00012     this->name = other.name;
00013     this->source = other.source;
00014     this->target = other.target;
00015 }
```

References FlowIMP::name, FlowIMP::source, and FlowIMP::target.

4.4.1.2 Logistical() [2/2]

```
Logistical::Logistical (
              const std::string & name = "NO_NAME",
              System * source = NULL,
              System * target = NULL )
```

Construct a new Logistical by name, source and target.

Parameters

name	string with default value "NO_NAME"
source	System pointer with default value NULL
target	System pointer with default value NULL

```
00004                                                                                {
00005     this->name = name;
00006     this->source = source;
00007     this->target = target;
00008 }
```

References FlowIMP::name, FlowIMP::source, and FlowIMP::target.

4.4.1.3 ~Logistical()

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

This destructor is a virtual destructor of the Class.

```
00018 {}
```

4.4.2 Member Function Documentation

4.4.2.1 execute()

```
double Logistical::execute ( )  [override], [virtual]
```

Pure virtual method that will contain an equation that will be executed in the flow by the model.

Returns

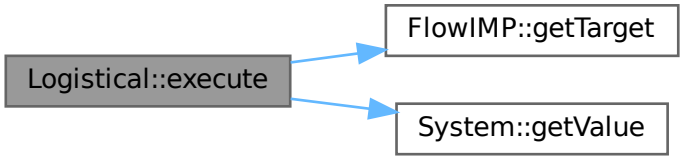
double

Implements FlowIMP.

```
00020                           {
00021     return 0.01 * getTarget()->getValue() * (1.0 - getTarget()->getValue() / 70.0);
00022 }
```

References FlowIMP::getTarget(), and System::getValue().

Here is the call graph for this function:



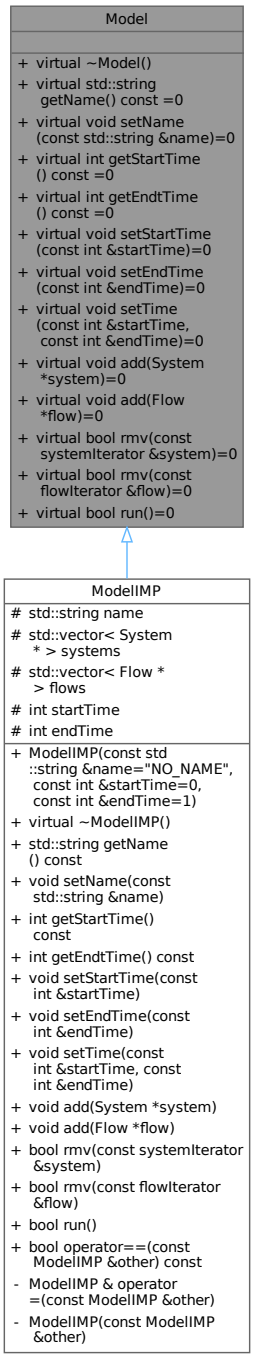
The documentation for this class was generated from the following files:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_↩ tests/src/Logistical.h
- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_↩ tests/src/Logistical.cpp

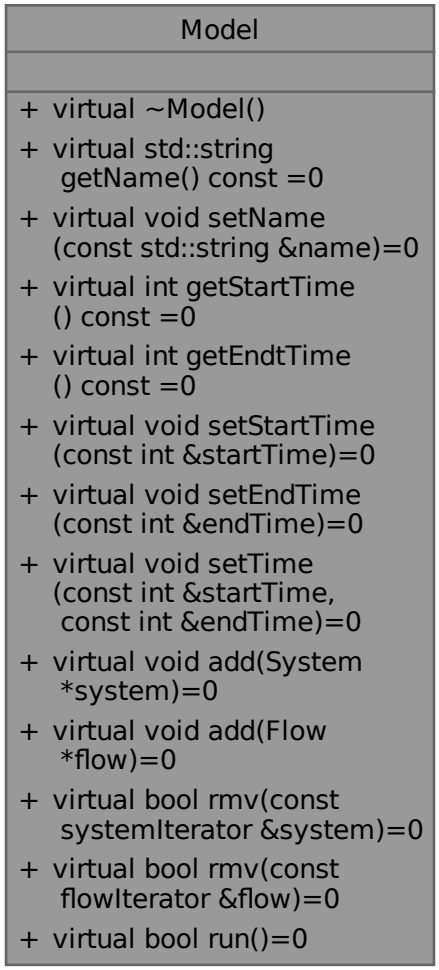
4.5 Model Class Reference

`#include <Model.h>`

Inheritance diagram for Model:



Collaboration diagram for Model:



Public Types

- typedef std::vector< System * >::iterator systemIterator

  typedef vetors iterators
- typedef std::vector< Flow * >::iterator flowIterator

Public Member Functions

- virtual ~Model ()

  This destructor is a virtual destructor of the class.
- virtual std::string getName () const =0

  This method returns the name of a Model.
- virtual void setName (const std::string &name)=0

  This method assigns a string to the name of a Model.

- virtual int getStartTime () const =0

  This method returns the startTime of a Model.
- virtual int getEndtTime () const =0

  This method returns the end of a Model.
- virtual void setStartTime (const int &startTime)=0

  This method assigns a int to the startTime of a Model.
- virtual void setEndTime (const int &endTime)=0

  This method assigns a int to the endTime of a Model.
- virtual void setTime (const int &startTime, const int &endTime)=0

  This method assigns a int to the startTime and endTime of a Model.
- virtual void add (System ∗system)=0

  This method add a System pointer to the vector of a Model.
- virtual void add (Flow ∗flow)=0

  This method add a Flow pointer to the vector of a Model.
- virtual bool rmv (const systemIterator &system)=0

  This method remove a System pointer of the vector of a Model.
- virtual bool rmv (const flowIterator &flow)=0

  This method remove a Flow pointer of the vector of a Model.
- virtual bool run ()=0

  This method run all model.

4.5.1 Member Typedef Documentation

4.5.1.1 flowIterator

```
typedef std::vector<Flow*>::iterator Model::flowIterator
```

4.5.1.2 systemIterator

```
typedef std::vector<System*>::iterator Model::systemIterator
```

typedef vetors iterators

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ∼Model()

```
virtual Model::∼Model ( )  [inline], [virtual]
```

This destructor is a virtual destructor of the class.

```
00032 {};
```

4.5.3 Member Function Documentation

4.5.3.1 add() [1/2]

```
virtual void Model::add (
            Flow * flow )  [pure virtual]
```

This method add a Flow pointer to the vector of a Model.

Parameters

flow	Flow pointer must be passed to the method

Implemented in ModelIMP.

4.5.3.2 add() [2/2]

```
virtual void Model::add (
            System * system )  [pure virtual]
```

This method add a System pointer to the vector of a Model.

Parameters

system	System pointer must be passed to the method

Implemented in ModelIMP.

4.5.3.3 getEndtTime()

```
virtual int Model::getEndtTime ( ) const  [pure virtual]
```

This method returns the end of a Model.

Returns

a int containing the end is returned

Implemented in ModelIMP.

4.5.3.4 getName()

```
virtual std::string Model::getName ( ) const  [pure virtual]
```

This method returns the name of a Model.

Returns

a string containing the name is returned

Implemented in ModelIMP.

4.5.3.5 getStartTime()

```
virtual int Model::getStartTime ( ) const  [pure virtual]
```

This method returns the startTime of a Model.

Returns

a int containing the startTime is returned

Implemented in ModelIMP.

4.5.3.6 rmv() [1/2]

```
virtual bool Model::rmv (
            const flowIterator & flow )  [pure virtual]
```

This method remove a Flow pointer of the vector of a Model.

Parameters

flow	Flow pointer iterator must be passed to the method

Returns

a bool value, true if can remove, false if not

Implemented in ModelIMP.

4.5.3.7 rmv() [2/2]

```
virtual bool Model::rmv (
            const systemIterator & system )  [pure virtual]
```

This method remove a System pointer of the vector of a Model.

Parameters

system	System pointer iterator must be passed to the method

Returns

a bool value, true if can remove, false if not

Implemented in ModelIMP.

4.5.3.8 run()

```
virtual bool Model::run ( )  [pure virtual]
```

This method run all model.

Returns

a bool value, true if can run, false if not

Implemented in ModelIMP.

4.5.3.9 setEndTime()

```
virtual void Model::setEndTime (
            const int & endTime )  [pure virtual]
```

This method assigns a int to the endTime of a Model.

Parameters

endTime	int must be passed to the method

Implemented in ModelIMP.

4.5.3.10 setName()

```
virtual void Model::setName (
            const std::string & name )  [pure virtual]
```

This method assigns a string to the name of a Model.

Parameters

name	string must be passed to the method

Implemented in ModelIMP.

4.5.3.11 setStartTime()

```
virtual void Model::setStartTime (
            const int & startTime )  [pure virtual]
```

This method assigns a int to the startTime of a Model.

Parameters

startTime	int must be passed to the method

Implemented in ModelIMP.

4.5.3.12 setTime()

```
virtual void Model::setTime (
            const int & startTime,
            const int & endTime )  [pure virtual]
```

This method assigns a int to the startTime and endTime of a Model.

Parameters

startTime	int must be passed to the method
endTime	int must be passed to the method

Implemented in ModelIMP.

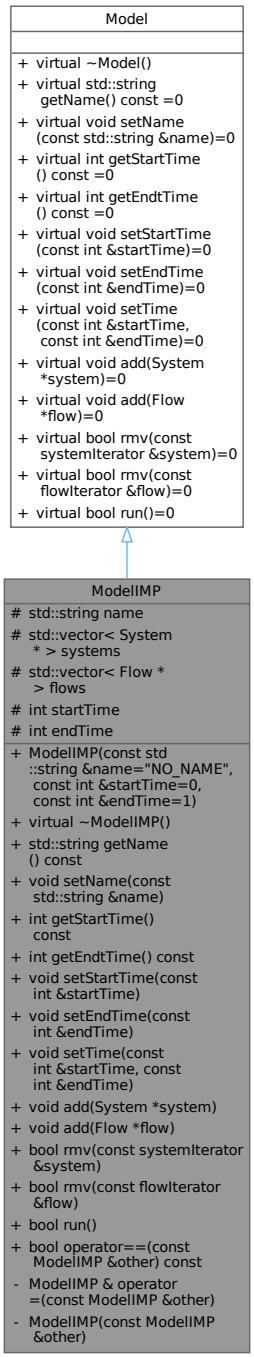
The documentation for this class was generated from the following file:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.h

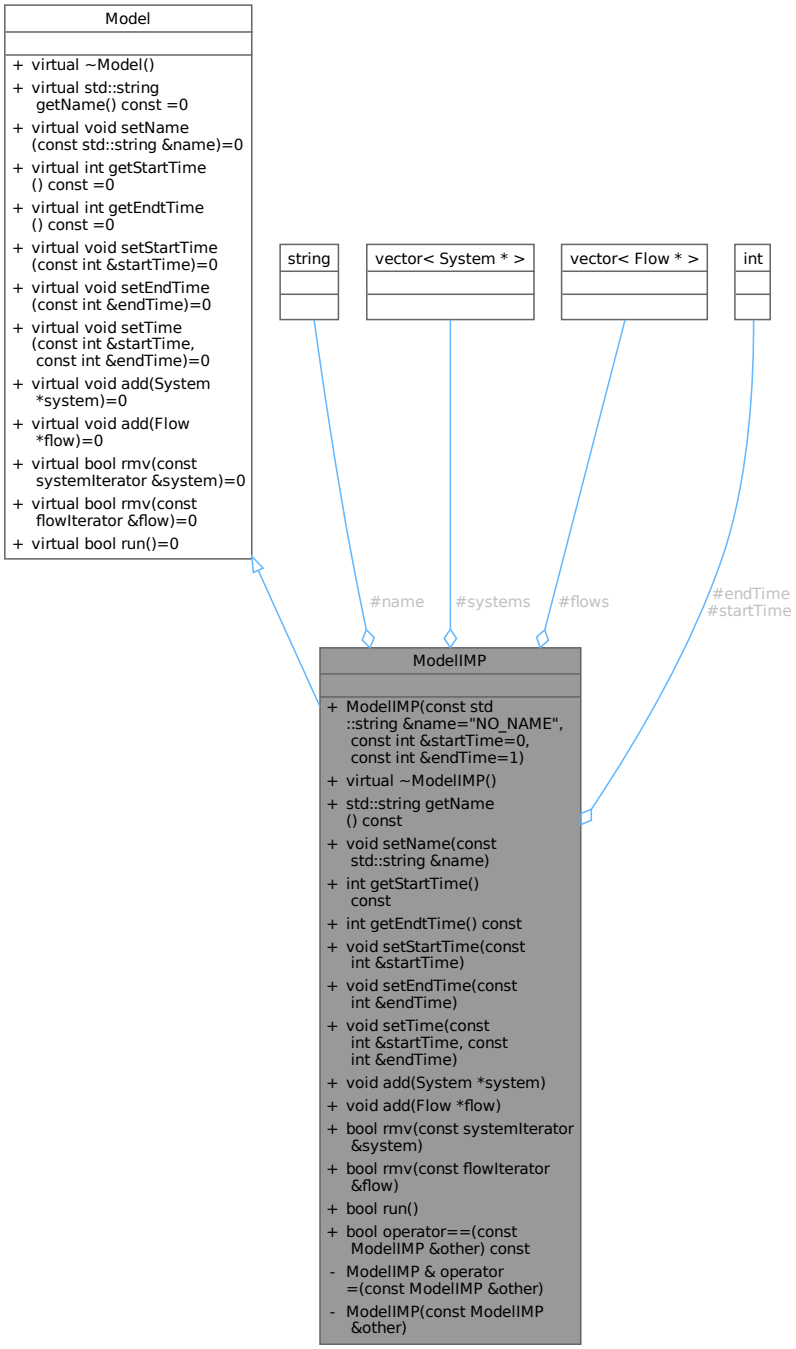
4.6 ModelIMP Class Reference

```
#include <ModelIMP.h>
```

Inheritance diagram for ModelIMP:



Collaboration diagram for ModelIMP:



Public Member Functions

- ModelIMP (const std::string &name="NO_NAME", const int &startTime=0, const int &endTime=1)

  Construct a new Model by name and sart and end time.
- virtual ~ModelIMP ()

  This destructor is a virtual destructor of the class.
- std::string getName () const

This method returns the name of a Model.

- void setName (const std::string &name)

  This method assigns a string to the name of a Model.
- int getStartTime () const

  This method returns the startTime of a Model.
- int getEndtTime () const

  This method returns the end of a Model.
- void setStartTime (const int &startTime)

  This method assigns a int to the startTime of a Model.
- void setEndTime (const int &endTime)

  This method assigns a int to the endTime of a Model.
- void setTime (const int &startTime, const int &endTime)

  This method assigns a int to the startTime and endTime of a Model.
- void add (System *system)

  This method add a System pointer to the vector of a Model.
- void add (Flow *flow)

  This method add a Flow pointer to the vector of a Model.
- bool rmv (const systemIterator &system)

  This method remove a System pointer of the vector of a Model.
- bool rmv (const flowIterator &flow)

  This method remove a Flow pointer of the vector of a Model.
- bool run ()

  This method run all model.
- bool operator== (const ModelIMP &other) const

  This method is overloading the '==' operator, compare two models objs.

Public Member Functions inherited from Model

- virtual ~Model ()

  This destructor is a virtual destructor of the class.

Protected Attributes

- std::string name
- std::vector< System * > systems
- std::vector< Flow * > flows
- int startTime
- int endTime

Private Member Functions

- ModelIMP & operator= (const ModelIMP &other)

  This method is overloading the '=' operator, "cloning" from one Model to another.
- ModelIMP (const ModelIMP &other)

  Construct a new Model by a obj.

Friends

- std::ostream & operator<< (std::ostream &out, const ModelIMP &obj)

  This method is overloading the '<<' operator, print the model obj info.

Additional Inherited Members

Public Types inherited from Model

- typedef std::vector< System * >::iterator systemIterator

  typedef vetors iterators
- typedef std::vector< Flow * >::iterator flowIterator

4.6.1 Constructor & Destructor Documentation

4.6.1.1 ModelIMP() [1/2]

```
ModelIMP::ModelIMP (
            const ModelIMP & other )  [private]
```

Construct a new Model by a obj.

Parameters

other	Model obj

```
00006                                       : name(other.name), startTime(other.startTime),
      endTime(other.endTime) {
00007     flows.clear();
00008     systems.clear();
00009     for (auto i : other.flows) flows.push_back(i);
00010     for (auto i : other.systems) systems.push_back(i);
00011 }
```

References flows, and systems.

4.6.1.2 ModelIMP() [2/2]

```
ModelIMP::ModelIMP (
            const std::string & name = "NO_NAME",
            const int & startTime = 0,
            const int & endTime = 1 )
```

Construct a new Model by name and sart and end time.

Parameters

name	string with default value "NO_NAME"
startTime	int with default value 0
endTime	int with default value 1

```
00004 : name(name), startTime(startTime), endTime(endTime) {}
```

4.6.1.3 ~ModelIMP()

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

This destructor is a virtual destructor of the class.

```
00014 {systems.clear(); flows.clear();}
```

References flows, and systems.

4.6.2 Member Function Documentation

4.6.2.1 add() [1/2]

```
void ModelIMP::add (
            Flow * flow )  [virtual]
```

This method add a Flow pointer to the vector of a Model.

Parameters

flow	Flow pointer must be passed to the method

Implements Model.

```
00030 { flows.push_back(flow); }
```

References flows.

4.6.2.2 add() [2/2]

```
void ModelIMP::add (
            System * system )  [virtual]
```

This method add a System pointer to the vector of a Model.

Parameters

system	System pointer must be passed to the method

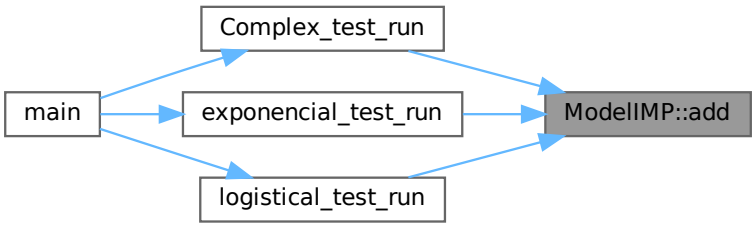
Implements Model.

```
00029 { systems.push_back(system); }
```

References systems.

Referenced by Complex_test_run(), exponencial_test_run(), and logistical_test_run().

Here is the caller graph for this function:



4.6.2.3 getEndtTime()

```
int ModelIMP::getEndtTime ( ) const  [virtual]
```

This method returns the end of a Model.

Returns

a int containing the end is returned

Implements Model.

```
00022 { return endTime; }
```

References endTime.

4.6.2.4 getName()

```
std::string ModelIMP::getName ( ) const  [virtual]
```

This method returns the name of a Model.

Returns

a string containing the name is returned

Implements Model.

```
00018 { return name; }
```

References name.

4.6.2.5 getStartTime()

```
int ModelIMP::getStartTime ( ) const  [virtual]
```

This method returns the startTime of a Model.

Returns

a int containing the startTime is returned

Implements Model.

```
00021 { return startTime; }
```

References startTime.

4.6.2.6 operator=()

```
ModelIMP & ModelIMP::operator= (
            const ModelIMP & other )  [private]
```

This method is overloading the '=' operator, "cloning" from one Model to another.

Parameters

other	Model obj to be cloned must be passed

Returns

A Model is returned that is a clone of what was passed to the method

```
00074                                                {
00075     if(other == *this) return *this;
00076     name = other.name;
00077     systems = other.systems;
00078     flows.clear();
00079     systems.clear();
00080     for (auto i : other.flows) flows.push_back(i);
00081     for (auto i : other.systems) systems.push_back(i);
00082     startTime = other.startTime;
00083     endTime = other.endTime;
00084     return *this;
00085 }
```

References endTime, flows, name, startTime, and systems.

4.6.2.7 operator==()

```
bool ModelIMP::operator== (
            const ModelIMP & other ) const
```

This method is overloading the '==' operator, compare two models objs.

Parameters

other	model obj to be compare must be passed

Returns

A bool is returned, true if they are equal and false if not

```
00087                                                 {
00088     return (name == other.name && systems == other.systems && flows == other.flows && startTime ==
      other.startTime && endTime == other.endTime);
00089 }
```

References endTime, flows, name, startTime, and systems.

4.6.2.8 rmv() [1/2]

```
bool ModelIMP::rmv (
            const flowIterator & flow )  [virtual]
```

This method remove a Flow pointer of the vector of a Model.

Parameters

flow	Flow pointer iterator must be passed to the method

Returns

a bool value, true if can remove, false if not

Implements Model.

```
00033 { return (flows.erase(flow) != flows.end()); }
```

References flows.

4.6.2.9 rmv() [2/2]

```
bool ModelIMP::rmv (
            const systemIterator & system )  [virtual]
```

This method remove a System pointer of the vector of a Model.

Parameters

system	System pointer iterator must be passed to the method

Returns

a bool value, true if can remove, false if not

Implements Model.

```
00032 { return (systems.erase(system) != systems.end()); }
```

References systems.

4.6.2.10 run()

```
bool ModelIMP::run ( )  [virtual]
```

This method run all model.

Returns

a bool value, true if can run, false if not

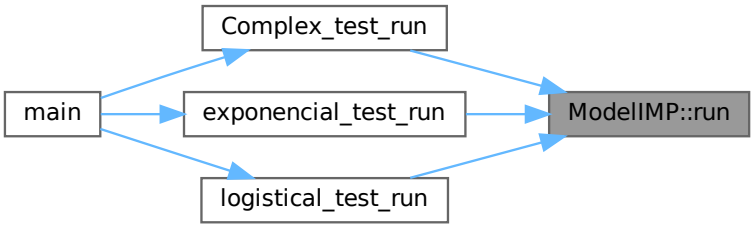
Implements Model.

```
00036                 {
00037     std::vector<double> flowValue;
00038     flowIterator f;
00039     std::vector<double>::iterator d;
00040     double calcValue;
00041
00042     for(int i = startTime; i < endTime; i++){
00043
00044         f = flows.begin();
00045
00046         while (f != flows.end()) {
00047             flowValue.push_back((*f)->execute());
00048             f++;
00049         }
00050
00051         f = flows.begin();
00052         d = flowValue.begin();
00053
00054         while(f != flows.end()){
00055             calcValue = (*f)->getSource()->getValue() - (*d);
00056             (*f)->getSource()->setValue(calcValue);
00057             calcValue = (*f)->getTarget()->getValue() + (*d);
00058             (*f)->getTarget()->setValue(calcValue);
00059             f++;
00060             d++;
00061         }
00062
00063         flowValue.clear();
00064
00065     }
00066
00067     return true;
00068 }
```

References endTime, flows, and startTime.

Referenced by Complex_test_run(), exponencial_test_run(), and logistical_test_run().

Here is the caller graph for this function:



4.6.2.11 setEndTime()

```
void ModelIMP::setEndTime (
            const int & endTime )  [virtual]
```

This method assigns a int to the endTime of a Model.

Parameters

endTime	int must be passed to the method

Implements Model.
```
00024 { this->endTime = endTime; }
```

References endTime.

4.6.2.12 setName()

```
void ModelIMP::setName (
            const std::string & name )  [virtual]
```

This method assigns a string to the name of a Model.

Parameters

name	string must be passed to the method

Implements Model.
```
00019 { this->name = name; }
```

References name.

4.6.2.13 setStartTime()

```
void ModelIMP::setStartTime (
            const int & startTime )  [virtual]
```

This method assigns a int to the startTime of a Model.

Parameters

startTime	int must be passed to the method

Implements Model.
```
00023 { this->startTime = startTime; }
```

References startTime.

4.6.2.14 setTime()

```
void ModelIMP::setTime (
            const int & startTime,
            const int & endTime )  [virtual]
```

This method assigns a int to the startTime and endTime of a Model.

Parameters

startTime	int must be passed to the method
endTime	int must be passed to the method

Implements Model.

```
00025 { this->startTime = startTime; this->endTime = endTime; }
```

References endTime, and startTime.

4.6.3 Friends And Related Symbol Documentation

4.6.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & out,
            const ModelIMP & obj )  [friend]
```

This method is overloading the '<<' operator, print the model obj info.

Parameters

out	is a ostream obj
obj	is a model obj

Returns

a ostream obj to print the obj info

```
00091                                                                {
00092     out « "Name: " « obj.name « ";\n"
00093         « "Systems:\n";
00094     for (auto item : obj.systems) out « item « "\n";
00095     out « "Flows:\n";
00096     for (auto item : obj.flows) out « item « "\n";
00097     return out;
00098 }
```

4.6.4 Member Data Documentation

4.6.4.1 endTime

```
int ModelIMP::endTime  [protected]
```

End time simulation integer attribute.

Referenced by getEndtTime(), operator=(), operator==(), run(), setEndTime(), and setTime().

4.6.4.2 flows

```
std::vector<Flow*> ModelIMP::flows  [protected]
```

Flow pointers vector.

Referenced by add(), ModelIMP(), operator=(), operator==(), rmv(), run(), and ~ModelIMP().

4.6.4.3 name

```
std::string ModelIMP::name  [protected]
```

Name string attribute.

Referenced by getName(), operator=(), operator==(), and setName().

4.6.4.4 startTime

```
int ModelIMP::startTime  [protected]
```

Start time simulation integer attribute.

Referenced by getStartTime(), operator=(), operator==(), run(), setStartTime(), and setTime().

4.6.4.5 systems

```
std::vector<System*> ModelIMP::systems  [protected]
```

System pointers vector.

Referenced by add(), ModelIMP(), operator=(), operator==(), rmv(), and ~ModelIMP().

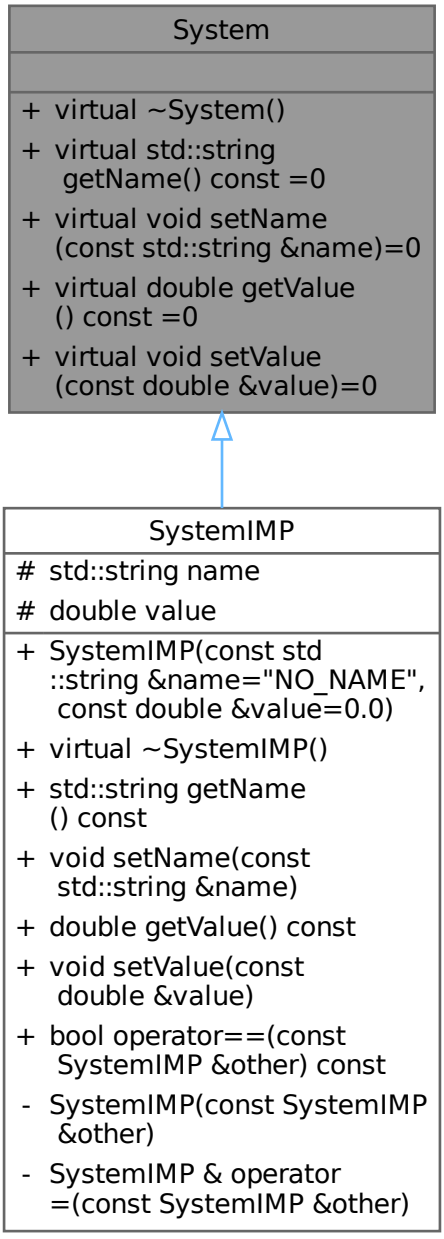
The documentation for this class was generated from the following files:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/ModelIMP.h
- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/ModelIMP.cpp

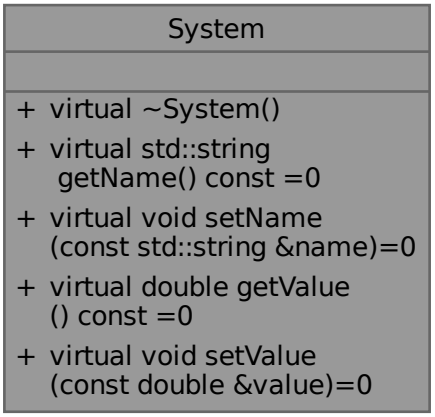
4.7 System Class Reference

```
#include <System.h>
```

Inheritance diagram for System:



Collaboration diagram for System:



Public Member Functions

- virtual ~System ()
  This destructor is a virtual destructor of the Class.
- virtual std::string getName () const =0
  This method returns the name of a system.
- virtual void setName (const std::string &name)=0
  This method assigns a string to the name of a system.
- virtual double getValue () const =0
  This method returns the value of a system.
- virtual void setValue (const double &value)=0
  This method assigns a double to the value of a system.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 ~System()

```
virtual System::~System ( )  [inline], [virtual]
```

This destructor is a virtual destructor of the Class.

```
00024 {};
```

4.7.2 Member Function Documentation

4.7.2.1 getName()

```
virtual std::string System::getName ( ) const  [pure virtual]
```

This method returns the name of a system.

Returns

a string containing the name is returned

Implemented in SystemIMP.

4.7.2.2 getValue()

```
virtual double System::getValue ( ) const  [pure virtual]
```

This method returns the value of a system.

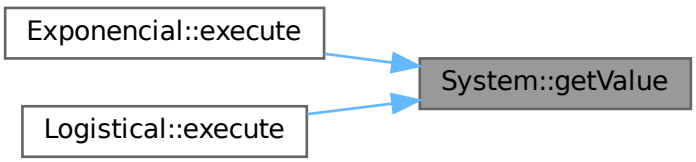
Returns

a double containing the value is returned

Implemented in SystemIMP.

Referenced by Exponencial::execute(), and Logistical::execute().

Here is the caller graph for this function:



4.7.2.3 setName()

```
virtual void System::setName (
            const std::string & name )  [pure virtual]
```

This method assigns a string to the name of a system.

Parameters

name	string must be passed to the method

Implemented in SystemIMP.

4.7.2.4 setValue()

```
virtual void System::setValue (
            const double & value )  [pure virtual]
```

This method assigns a double to the value of a system.

Parameters

value	double must be passed to the method

Implemented in SystemIMP.

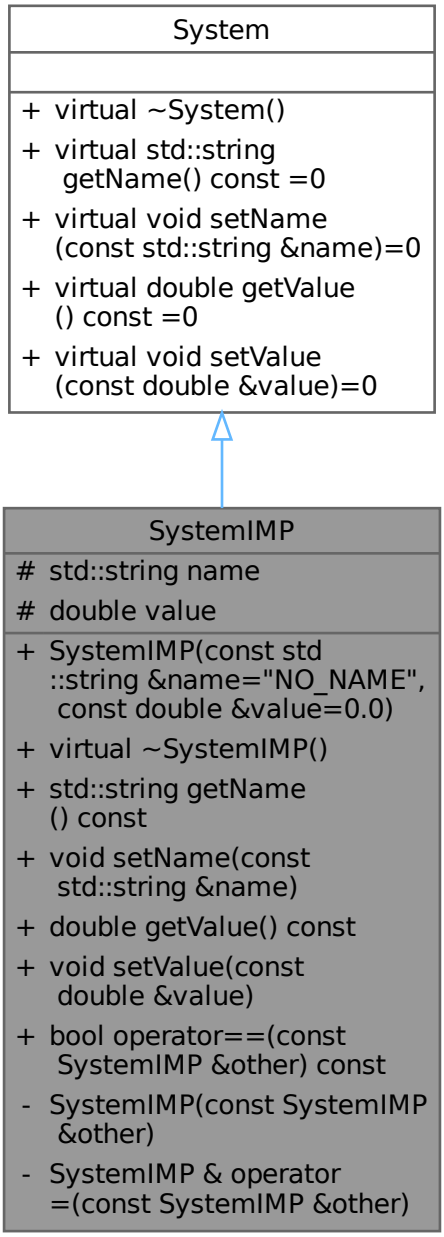
The documentation for this class was generated from the following file:

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/System.h

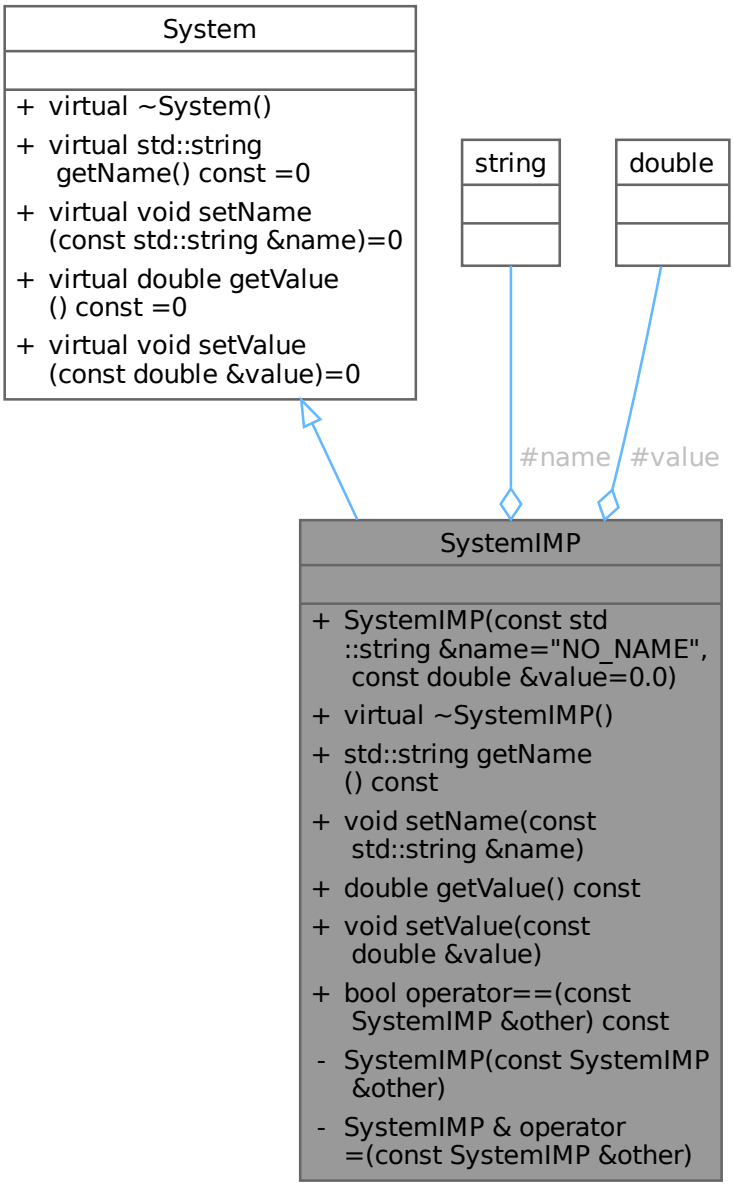
4.8 SystemIMP Class Reference

```
#include <SystemIMP.h>
```

Inheritance diagram for SystemIMP:



Collaboration diagram for SystemIMP:



Public Member Functions

- SystemIMP (const std::string &name="NO_NAME", const double &value=0.0)

  Construct a new System by name and value.
- virtual ~SystemIMP ()

  This destructor is a virtual destructor of the Class.
- std::string getName () const

  This method returns the name of a system.
- void setName (const std::string &name)

This method assigns a string to the name of a system.

- double getValue () const

  This method returns the value of a system.

- void setValue (const double &value)

  This method assigns a double to the value of a system.

- bool operator== (const SystemIMP &other) const

  This method is overloading the '==' operator, compare two systems objs.

Public Member Functions inherited from System

- virtual ~System ()

  This destructor is a virtual destructor of the Class.

Protected Attributes

- std::string name
- double value

Private Member Functions

- SystemIMP (const SystemIMP &other)

  Construct a new System by a obj.

- SystemIMP & operator= (const SystemIMP &other)

  This method is overloading the '=' operator, "cloning" from one system to another.

Friends

- std::ostream & operator<< (std::ostream &out, const SystemIMP &obj)

  This method is overloading the '<<' operator, print the system obj info.

4.8.1 Constructor & Destructor Documentation

4.8.1.1 SystemIMP() [1/2]

```
SystemIMP::SystemIMP (
            const SystemIMP & other )  [private]
```

Construct a new System by a obj.

Parameters

other	System obj

```
00006 : name(other.name), value(other.value) {}
```

4.8.1.2 SystemIMP() [2/2]

```
SystemIMP::SystemIMP (
            const std::string & name = "NO_NAME",
            const double & value = 0.0 )
```

Construct a new System by name and value.

Parameters

name	string with default value "NO_NAME"
value	double with default value 0.0

```
00004 : name(name), value(value) {}
```

4.8.1.3 ~SystemIMP()

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

This destructor is a virtual destructor of the Class.

```
00009 {};
```

4.8.2 Member Function Documentation

4.8.2.1 getName()

```
std::string SystemIMP::getName ( ) const  [virtual]
```

This method returns the name of a system.

Returns

a string containing the name is returned

Implements System.

```
00013 { return name; }
```

References name.

4.8.2.2 getValue()

```
double SystemIMP::getValue ( ) const  [virtual]
```

This method returns the value of a system.

Returns

a double containing the value is returned

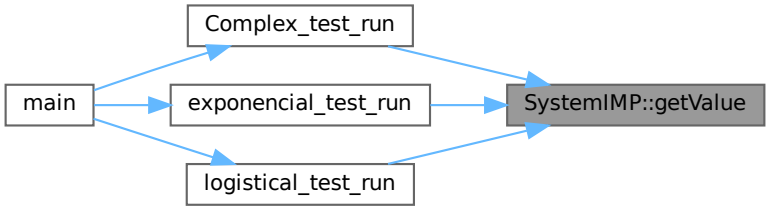
Implements System.

```
00016 { return value; }
```

References value.

Referenced by Complex_test_run(), exponencial_test_run(), and logistical_test_run().

Here is the caller graph for this function:



4.8.2.3 operator=()

```
SystemIMP & SystemIMP::operator= (
            const SystemIMP & other )  [private]
```

This method is overloading the '=' operator, "cloning" from one system to another.

Parameters

other	system obj to be cloned must be passed

Returns

A system is returned that is a clone of what was passed to the method

```
00021                                                   {
00022     if(other == *this) return *this;
00023     name = other.name;
00024     value = other.value;
00025     return *this;
00026 }
```

References name, and value.

4.8.2.4 operator==()

```
bool SystemIMP::operator== (
            const SystemIMP & other ) const
```

This method is overloading the '==' operator, compare two systems objs.

Parameters

other	system obj to be compare must be passed

Returns

A bool is returned, true if they are equal and false if not

```
00028                                                   {
00029     return (name == other.name && value == other.value);
00030     // Compare todos os membros para verificar igualdade
00031 }
```

References name, and value.

4.8.2.5 setName()

```
void SystemIMP::setName (
            const std::string & name )  [virtual]
```

This method assigns a string to the name of a system.

Parameters

name	string must be passed to the method

Implements System.

```
00014 { this->name = name; }
```

References name.

4.8.2.6 setValue()

```
void SystemIMP::setValue (
            const double & value )  [virtual]
```

This method assigns a double to the value of a system.

Parameters

value	double must be passed to the method

Implements System.

```
00017 { this->value = value; }
```

References value.

4.8.3 Friends And Related Symbol Documentation

4.8.3.1 operator<<

```
std::ostream & operator<< (
```

```
            std::ostream & out,
            const SystemIMP & obj )  [friend]
```

This method is overloading the '<<' operator, print the system obj info.

Parameters

out	is a ostream obj
obj	is a system obj

Returns

a ostream obj to print the obj info

```
00033                                                                 {
00034     out « "(System)(Name: " « obj.name « ", Value: " « obj.value « ")";
00035     return out;
00036 }
```

4.8.4 Member Data Documentation

4.8.4.1 name

```
std::string SystemIMP::name  [protected]
```

Name string attribute.

Referenced by getName(), operator=(), operator==(), and setName().

4.8.4.2 value

```
double SystemIMP::value  [protected]
```

Value double attribute.

Referenced by getValue(), operator=(), operator==(), and setValue().

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

- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/SystemIMP.h
- /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/SystemIMP.cpp

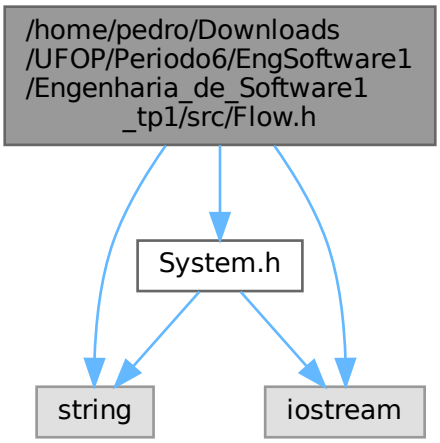
Chapter 5

File Documentation

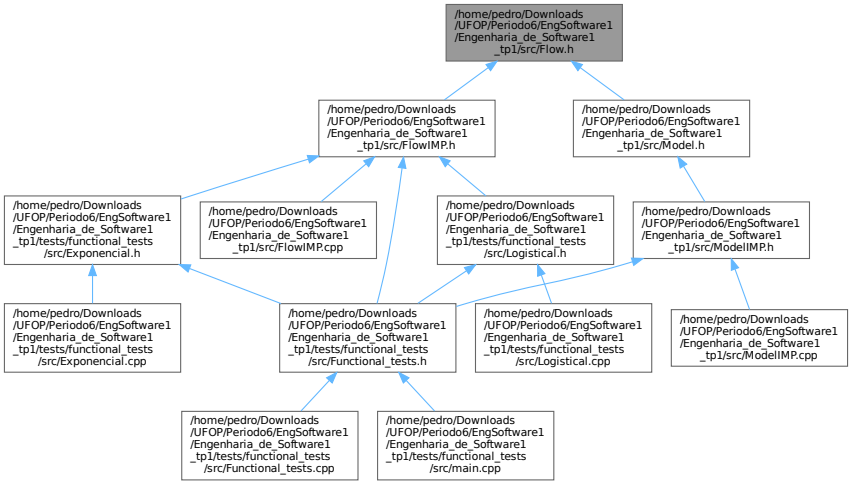
5.1 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.h File Reference

```
#include "System.h"
#include <string>
#include <iostream>
```

Include dependency graph for Flow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Flow

5.2 Flow.h

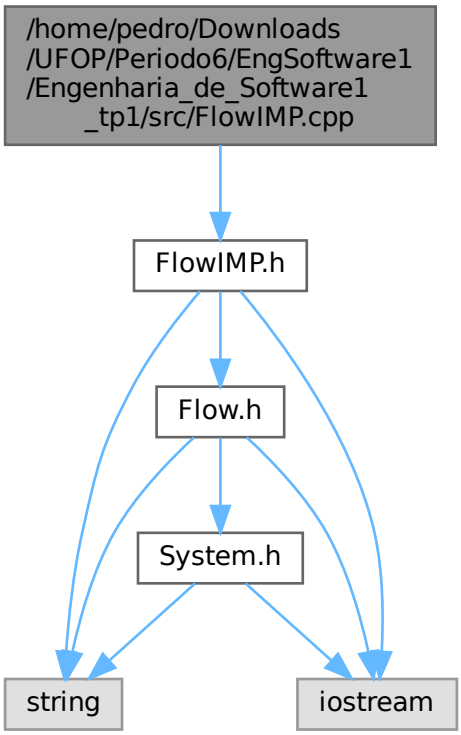
Go to the documentation of this file.

```
00001 /***********************************************
00002  * @file Flow.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the flow Interface
00005 ***********************************************/
00006
00007 #ifndef FLOW_H
00008 #define FLOW_H
00009
00010 #include "System.h"
00011 #include <string>
00012 #include <iostream>
00013
00014 /*****************************************************************************************
00015  *@brief The Flow Interface is the Interface that defines the methods to be implemented
00016 *****************************************************************************************/
00017
00018 class Flow{
00019     public:
00020         //Destructor
00024         virtual ~Flow() {};
00025
00026         //Geters e seters
00027         //Name
00032         virtual std::string getName() const = 0;
00037         virtual void setName(std::string& name) = 0;
00038         //Source
00043         virtual System* getSource() const = 0;
00048         virtual void setSource(System* source) = 0;
00049         //Target
00054         virtual System* getTarget() const = 0;
00059         virtual void setTarget(System* target) = 0;
00060
00061         //Metodos
00066         virtual double execute() = 0;
00067 };
00068
00069
00070 #endif
```

5.3 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/FlowIMP.cpp File Reference

```
#include "FlowIMP.h"
```

Include dependency graph for FlowIMP.cpp:



Functions

- std::ostream & operator<< (std::ostream &out, const FlowIMP &obj)

5.3.1 Function Documentation

5.3.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & out,
            const FlowIMP & obj )
```

Parameters

out	is a ostream obj
obj	is a flow obj

Returns

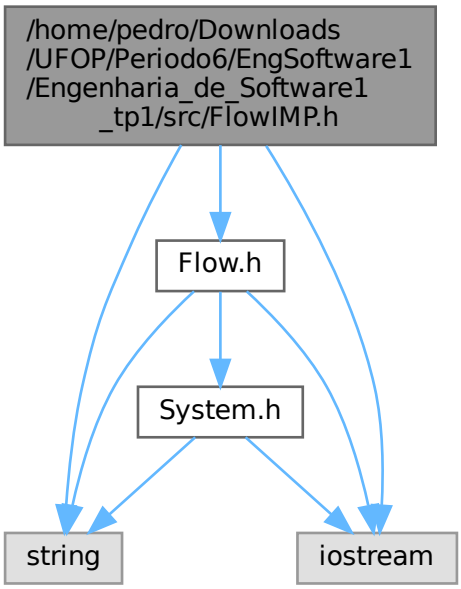
a ostream obj to print the obj info

```
00032                                                                   {
00033     out « "(Flow) Name: " « obj.name « " - "
00034         « obj.source->getName() « " -----> " « obj.target->getName();
00035     return out;
00036 }
```

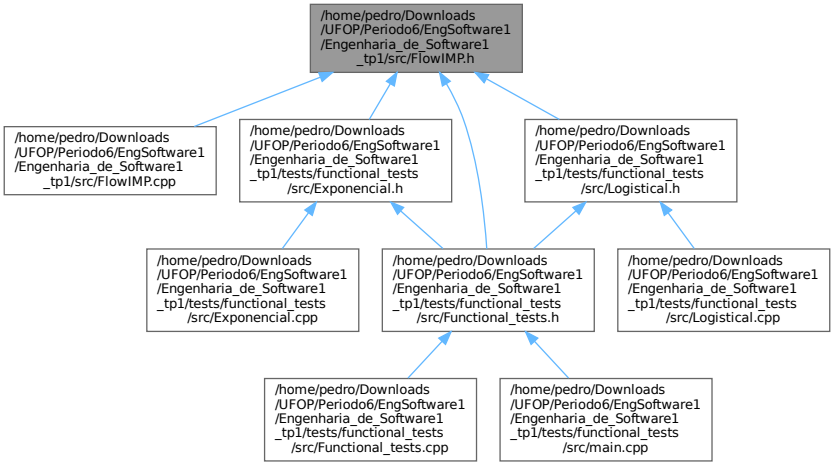
5.4 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/FlowIMP.h File Reference

```
#include "Flow.h"
#include <string>
#include <iostream>
```

Include dependency graph for FlowIMP.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FlowIMP

5.5 FlowIMP.h

Go to the documentation of this file.

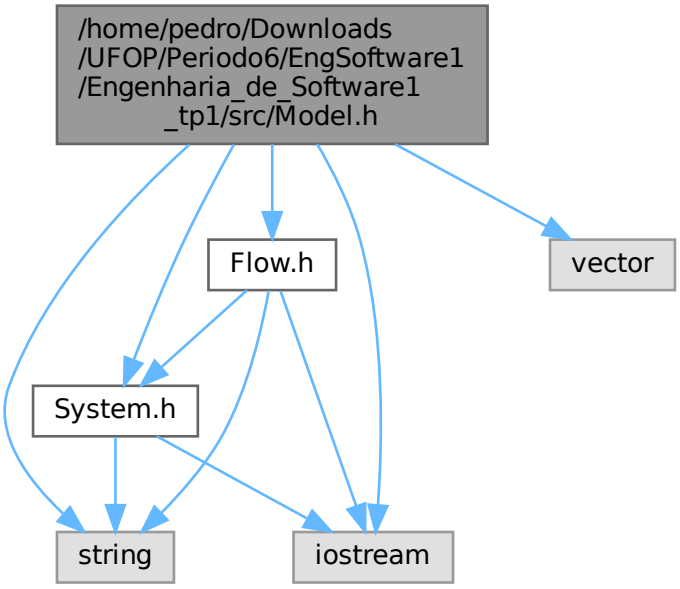
```
00001 /***********************************************
00002  * @file FlowIMP.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the flow implementation
00005 ***********************************************/
00006
00007 #ifndef FLOWIMP_H
00008 #define FLOWIMP_H
00009
00010 #include "Flow.h"
00011 #include <string>
00012 #include <iostream>
00013
00014 /*****************************************************************************
00015  *@brief The Flow implementation defines the attributes and implements the methods
00016 *****************************************************************************/
00017
00018 class FlowIMP : public Flow{
00019     private:
00025         FlowIMP& operator=(const FlowIMP& other); // Operador de atribuição
00026
00027     protected:
00028         std::string name;
00029         System* source;
00030         System* target;
00032     public:
00033         //Destructor
00037         virtual ~FlowIMP();
00038
00039         //Geters e seters
00040         //Name
00045         std::string getName() const;
00050         void setName(std::string& name);
00051         //Source
00056         System* getSource() const;
00061         void setSource(System* source);
00062         //Target
00067         System* getTarget() const;
00072         void setTarget(System* target);
```

```
00073
00074         //Metodos
00079         virtual double execute() = 0;
00080
00081         //Sobrecarga de operadores
00087         bool operator==(const FlowIMP& other) const; // Operador de igualdade
00094         friend std::ostream& operator<<(std::ostream& out, const FlowIMP& obj); //Operador de saida
00095 };
00096
00097
00098 #endif
```

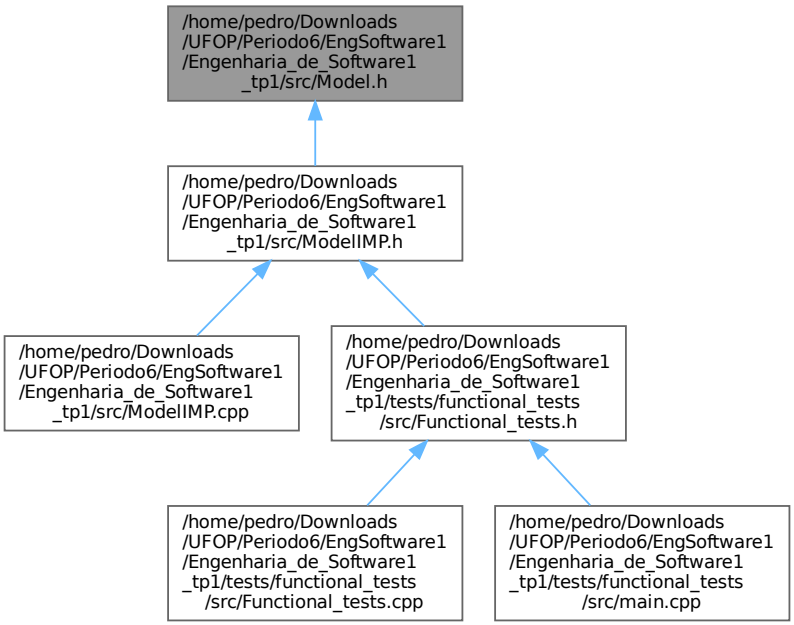
5.6 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.h File Reference

```
#include "System.h"
#include "Flow.h"
#include <string>
#include <iostream>
#include <vector>
```

Include dependency graph for Model.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Model

5.7 Model.h

Go to the documentation of this file.

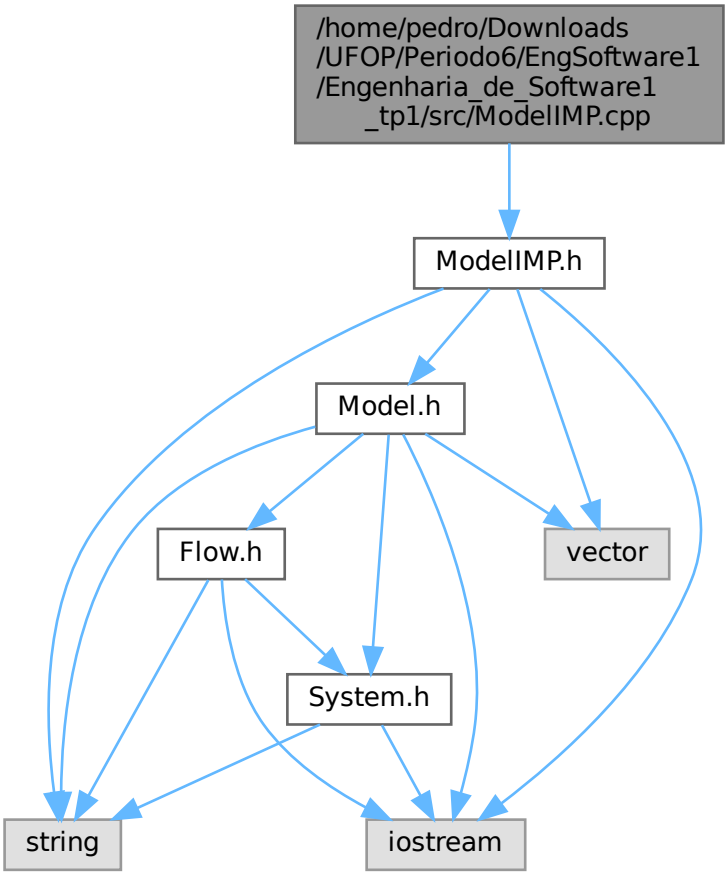
```
00001 /*************************************************
00002  * @file Model.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the simulation model
00005 *************************************************/
00006
00007 #ifndef MODEL_H
00008 #define MODEL_H
00009
00010 #include "System.h"
00011 #include "Flow.h"
00012 #include <string>
00013 #include <iostream>
00014 #include <vector>
00015
00016
      /*****************************************************************************************************************
00017  *@brief This class represents the general simulation model, it contains figures for simulation and
      its execution.
00018
      *****************************************************************************************************************/
00019 class Model{
00020     public:
00021         //Iteradores
00025         typedef std::vector<System*>::iterator systemIterator;
00026         typedef std::vector<Flow*>::iterator flowIterator;
00027
00028         //Destrutor
00032         virtual ~Model() {};
00033
```

```
00034         //Geters e seters
00035         //Name
00040         virtual std::string getName() const = 0;
00045         virtual void setName(const std::string& name) = 0;
00046         //Time
00051         virtual int getStartTime() const = 0;
00056         virtual int getEndtTime() const = 0;
00061         virtual void setStartTime(const int& startTime) = 0;
00066         virtual void setEndTime(const int& endTime) = 0;
00072         virtual void setTime(const int& startTime, const int& endTime) = 0;
00073
00074         //Metodos
00075         //add
00080         virtual void add(System* system) = 0;
00085         virtual void add(Flow* flow) = 0;
00086         //remove
00092         virtual bool rmv(const systemIterator& system) = 0;
00098         virtual bool rmv(const flowIterator& flow) = 0;
00099         //Others
00104         virtual bool run() = 0;
00105 };
00106
00107 #endif
```

5.8 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/ModelIMP.cpp File Reference

```
#include "ModelIMP.h"
```

Include dependency graph for ModelIMP.cpp:



Functions

- std::ostream & operator<< (std::ostream &out, const ModelIMP &obj)

5.8.1 Function Documentation

5.8.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & out,
            const ModelIMP & obj )
```

Parameters

out	is a ostream obj
obj	is a model obj

Returns

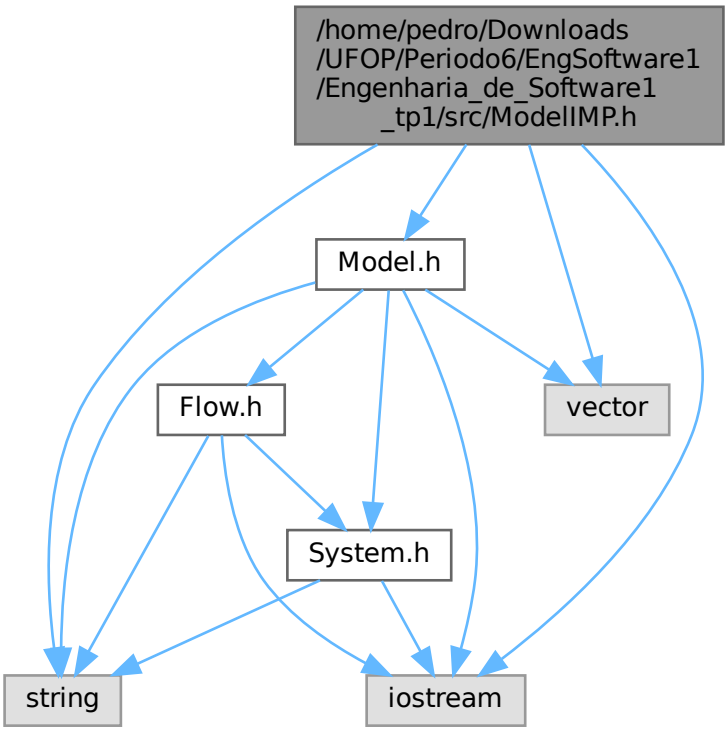
a ostream obj to print the obj info

```
00091                                                                    {
00092     out « "Name: " « obj.name « ";\n"
00093         « "Systems:\n";
00094     for (auto item : obj.systems) out « item « "\n";
00095     out « "Flows:\n";
00096     for (auto item : obj.flows) out « item « "\n";
00097     return out;
00098 }
```

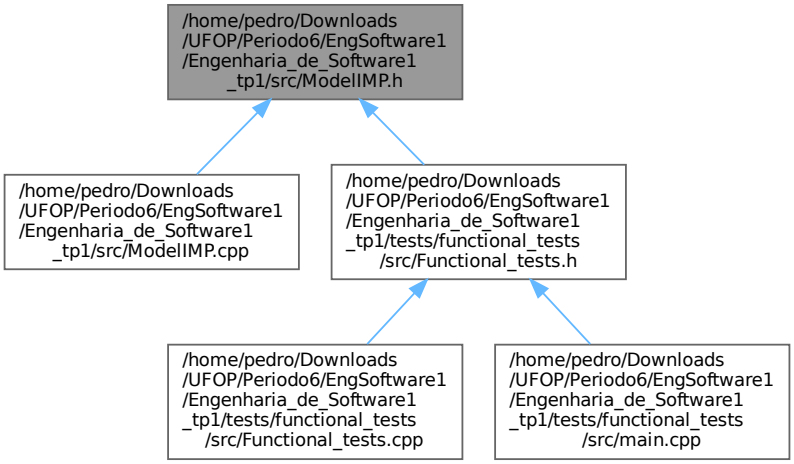
5.9 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/ModelIMP.h File Reference

```
#include "Model.h"
#include <string>
#include <iostream>
#include <vector>
```

Include dependency graph for ModelIMP.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ModelIMP

5.10 ModelIMP.h

Go to the documentation of this file.

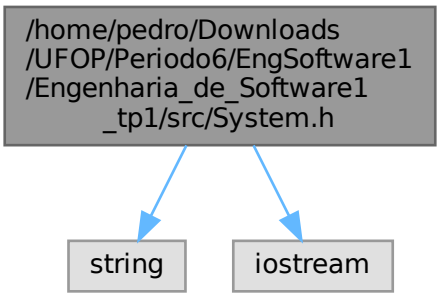
```
00001 /*****************************************************
00002  * @file ModelIMP.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the model implementation
00005 *****************************************************/
00006
00007 #ifndef MODELIMP_H
00008 #define MODELIMP_H
00009
00010 #include "Model.h"
00011 #include <string>
00012 #include <iostream>
00013 #include <vector>
00014
00015 /*******************************************************************************
00016  *@brief This class implementation defines the attributes and implements the methods
00017 *******************************************************************************/
00018 class ModelIMP : public Model{
00019     private:
00025         ModelIMP& operator=(const ModelIMP& other); // Operador de atribuição
00030         ModelIMP(const ModelIMP& other); //Copia outro flow
00031
00032     protected:
00033         std::string name;
00034         std::vector<System*> systems;
00035         std::vector<Flow*> flows;
00036         int startTime;
00037         int endTime;
00039     public:
00040         //Contructors
00047         ModelIMP(const std::string& name = "NO_NAME", const int& startTime = 0, const int& endTime =
      1);
00048
00049         //Destrutor
00053         virtual ~ModelIMP();
00054
00055         //Geters e seters
```

```
00056         //Name
00061         std::string getName() const;
00066         void setName(const std::string& name);
00067         //Time
00072         int getStartTime() const;
00077         int getEndtTime() const;
00082         void setStartTime(const int& startTime);
00087         void setEndTime(const int& endTime);
00093         void setTime(const int& startTime, const int& endTime);
00094
00095         //Metodos
00096         //add
00101         void add(System* system);
00106         void add(Flow* flow);
00107         //remove
00113         bool rmv(const systemIterator& system);
00119         bool rmv(const flowIterator& flow);
00120         //Others
00125         bool run();
00126
00127         //Sobrecarga de operadores
00133         bool operator==(const ModelIMP& other) const; // Operador de igualdade
00140         friend std::ostream& operator<<(std::ostream& out, const ModelIMP& obj); //Operador de saida
00141 };
00142
00143 #endif
```

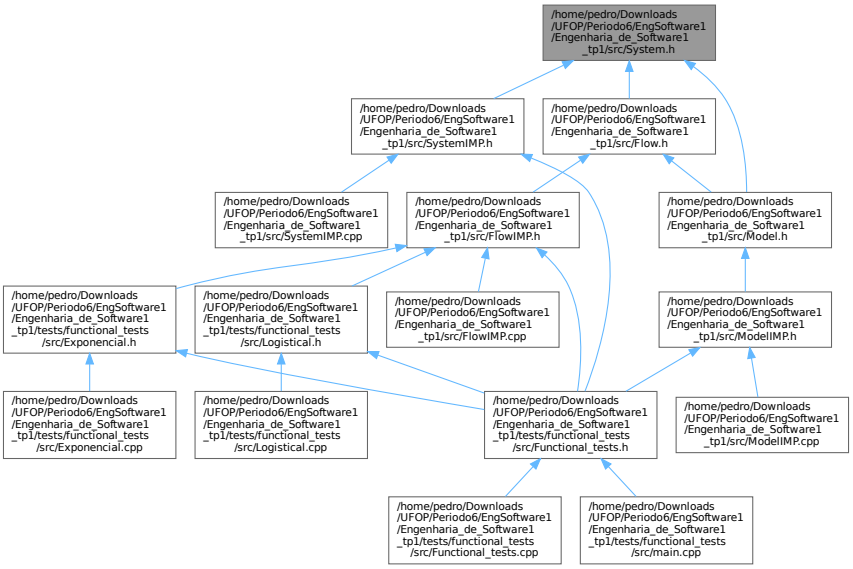
5.11 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/src/System.h File Reference

```
#include <string>
#include <iostream>
```

Include dependency graph for System.h:



This graph shows which files directly or indirectly include this file:



Classes

- class System

5.12 System.h

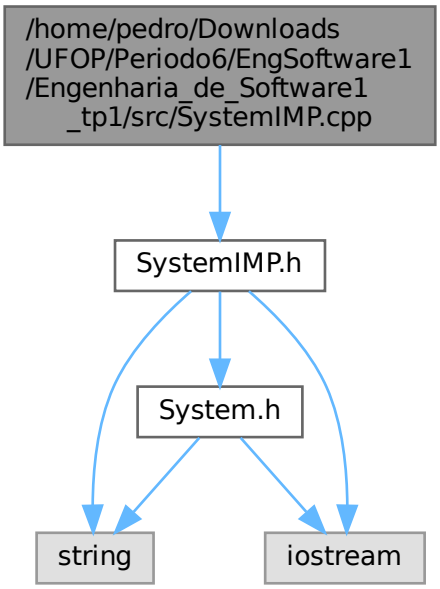
Go to the documentation of this file.

```
00001 /*************************************************
00002  * @file System.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the System interface
00005 *************************************************/
00006
00007 #ifndef SYSTEM_H
00008 #define SYSTEM_H
00009
00010 //Bibliotecas
00011 #include <string>
00012 #include <iostream>
00013
00014 /*****************************************************************************************
00015  *@brief The System Interface is the Interface that defines the methods to be implemented
00016 *****************************************************************************************/
00017
00018 class System{
00019     public:
00020         //Destructors
00024         virtual ~System() {};
00025
00026         //Geters e seters
00027         //Nome
00032         virtual std::string getName() const = 0;
00037         virtual void setName(const std::string& name) = 0;
00038         //Value
00043         virtual double getValue() const = 0;
00048         virtual void setValue(const double& value) = 0;
00049 };
00050
00051 #endif
```

5.13 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/src/SystemIMP.cpp File Reference

```
#include "SystemIMP.h"
```

Include dependency graph for SystemIMP.cpp:



Functions

- std::ostream & operator<< (std::ostream &out, const SystemIMP &obj)

5.13.1 Function Documentation

5.13.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & out,
            const SystemIMP & obj )
```

Parameters

out	is a ostream obj
obj	is a system obj

Returns

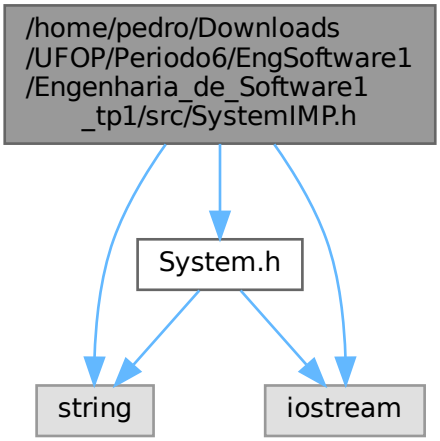
a ostream obj to print the obj info

```
00033                                                       {
00034     out « "(System)(Name: " « obj.name « ", Value: " « obj.value « ")";
00035     return out;
00036 }
```

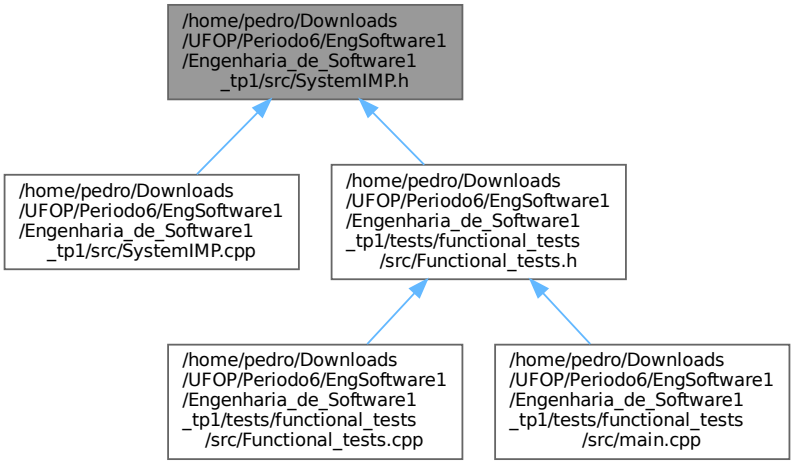
5.14 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/src/SystemIMP.h File Reference

```
#include "System.h"
#include <string>
#include <iostream>
```

Include dependency graph for SystemIMP.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SystemIMP

5.15 SystemIMP.h

Go to the documentation of this file.

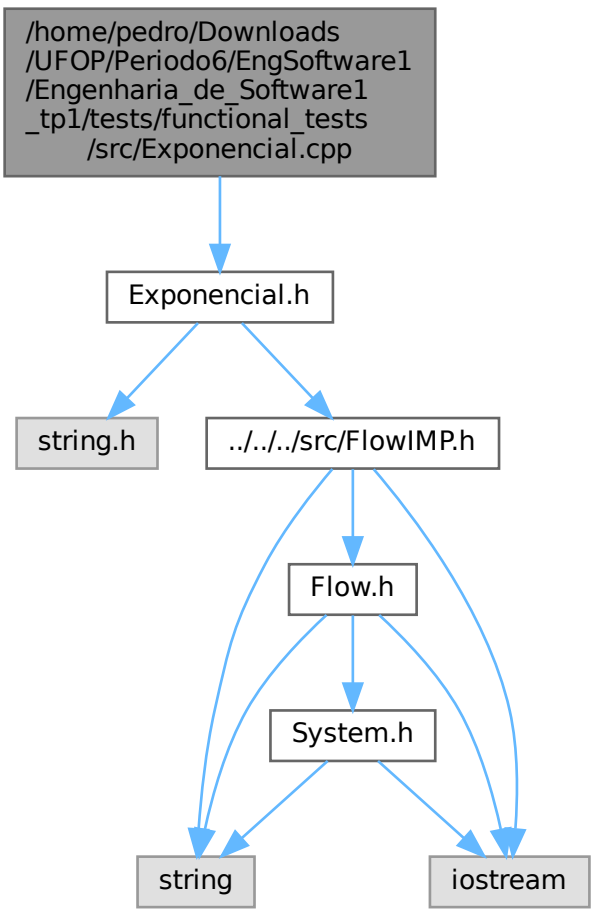
```
00001 /*****************************************************
00002  * @file SystemIMP.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the system implementation
00005 *****************************************************/
00006
00007 #ifndef SYSTEMIMP_H
00008 #define SYSTEMIMP_H
00009
00010 //Bibliotecas
00011 #include "System.h"
00012 #include <string>
00013 #include <iostream>
00014
00015 /*********************************************************************************************
00016  *@brief The System implementation defines the attributes and implements the methods
00017 *********************************************************************************************/
00018
00019 class SystemIMP : public System{
00020     private:
00025         SystemIMP(const SystemIMP& other); //Copia outro system
00031         SystemIMP& operator=(const SystemIMP& other);  // Operador de atribuição
00032
00033     protected:
00034         std::string name;
00035         double value;
00037     public:
00038         //Contructors
00044         SystemIMP(const std::string& name = "NO_NAME", const double& value = 0.0);
00045
00049         //Destructors
00050         virtual ~SystemIMP();
00051
00052         //Geters e seters
00053         //Nome
00058         std::string getName() const;
00063         void setName(const std::string& name);
```

```
00064         //Value
00069         double getValue() const;
00074         void setValue(const double& value);
00075
00076         //Sobrecarga de operadores
00082         bool operator==(const SystemIMP& other) const;  // Operador de igualdade
00089         friend std::ostream& operator«(std::ostream& out, const SystemIMP& obj); //Operador de saida
00090 };
00091
00092 #endif
```

5.16 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Exponencial.cpp File Reference

`#include "Exponencial.h"`

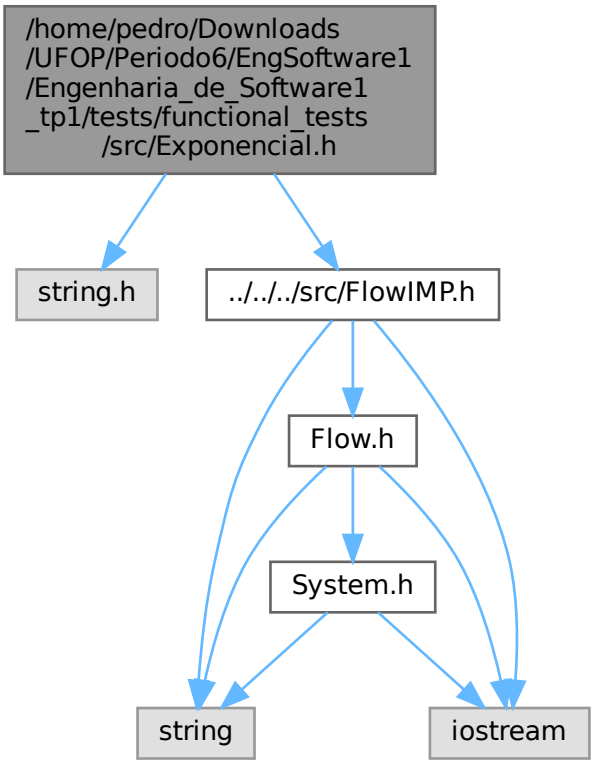
Include dependency graph for Exponencial.cpp:



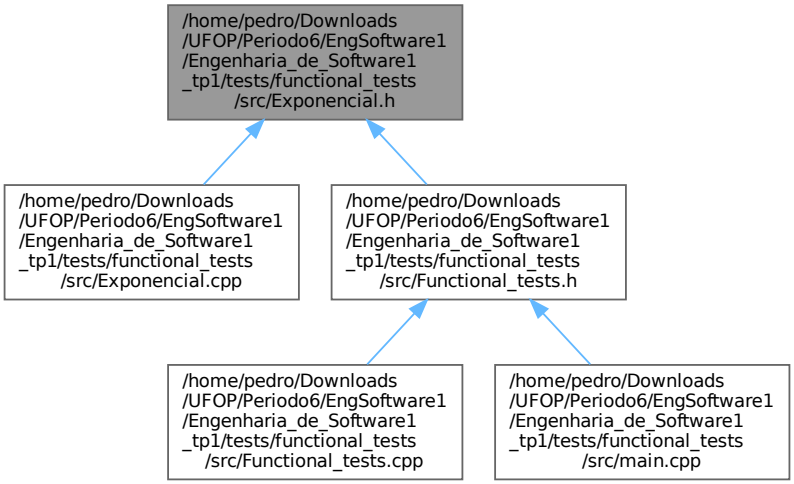
5.17 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Exponencial.h File Reference

```
#include <string.h>
#include "../../../src/FlowIMP.h"
```

Include dependency graph for Exponencial.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Exponencial

5.18 Exponencial.h

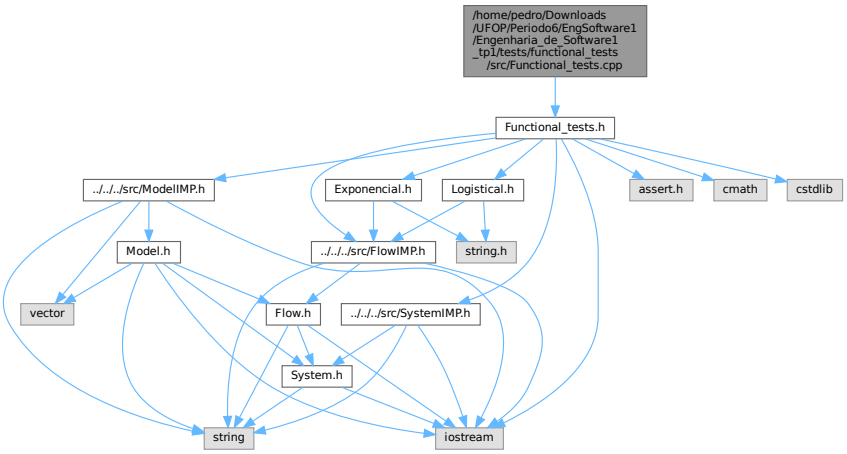
Go to the documentation of this file.

```
00001 /*************************************************************
00002  * @file Exponencial.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the exponential simulation flow
00005 *************************************************************/
00006
00007 #ifndef EXPONENCIAL_DEF
00008 #define EXPONENCIAL_DEF
00009
00010 #include <string.h>
00011 #include "../../../src/FlowIMP.h"
00012
00013
      /*****************************************************************************************************************
00014  * @brief This Flow class connects two systems and through the entered equation transfers values from
      one system to another
00015
      *****************************************************************************************************************
00016 class Exponencial : public FlowIMP{
00017     private:
00022         Exponencial(const Exponencial& other);
00023
00024     public:
00025         //Contructor
00032         Exponencial(const std::string& name = "NO_NAME", System* source = NULL, System* target =
      NULL);
00033
00034         //Destructor
00038         virtual ~Exponencial();
00039
00040         //Metodos
00045         virtual double execute() override;
00046 };
00047
00048 #endif
```

5.19 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Functional_tests.cpp File Reference

```
#include "Functional_tests.h"
```

Include dependency graph for Functional_tests.cpp:



Functions

- void exponencial_test_run ()

  This function performs the exponential functional test.
- void logistical_test_run ()

  This function performs the logistic test.
- void Complex_test_run ()

  This function runs the "complex" test, which has multiple systems and flows.

5.19.1 Function Documentation

5.19.1.1 Complex_test_run()

```
void Complex_test_run ( )
```

This function runs the "complex" test, which has multiple systems and flows.

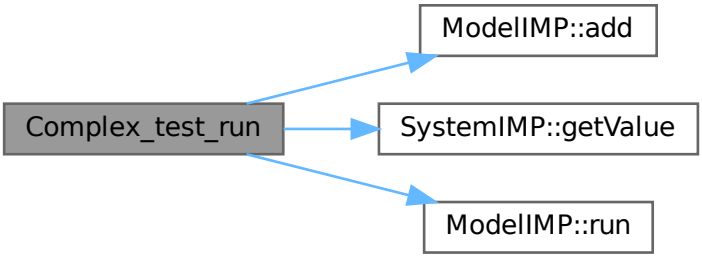
```
00055                       {
00056     std::cout « "Complex funcional test" « std::endl;
00057
00058     ModelIMP* model = new ModelIMP("Model", 0, 100);
00059     SystemIMP* q1 = new SystemIMP("q1", 100.0);
00060     SystemIMP* q2 = new SystemIMP("q2", 0.0);
00061     SystemIMP* q3 = new SystemIMP("q3", 100.0);
00062     SystemIMP* q4 = new SystemIMP("q4", 0.0);
00063     SystemIMP* q5 = new SystemIMP("q5", 0.0);
00064     Exponencial* f = new Exponencial("f", q1, q2);
00065     Exponencial* t = new Exponencial("t", q2, q3);
00066     Exponencial* u = new Exponencial("u", q3, q4);
00067     Exponencial* v = new Exponencial("v", q4, q1);
00068     Exponencial* g = new Exponencial("g", q1, q3);
00069     Exponencial* r = new Exponencial("r", q2, q5);
```

```
00070
00071     model->add(q1);
00072     model->add(q2);
00073     model->add(q3);
00074     model->add(q4);
00075     model->add(q5);
00076     model->add(f);
00077     model->add(t);
00078     model->add(u);
00079     model->add(v);
00080     model->add(g);
00081     model->add(r);
00082
00083     model->run();
00084
00085     assert(fabs((round((q1->getValue() * 10000)) - 10000 * 31.8513)) < 0.0001);
00086     assert(fabs((round((q2->getValue() * 10000)) - 10000 * 18.4003)) < 0.0001);
00087     assert(fabs((round((q3->getValue() * 10000)) - 10000 * 77.1143)) < 0.0001);
00088     assert(fabs((round((q4->getValue() * 10000)) - 10000 * 56.1728)) < 0.0001);
00089     assert(fabs((round((q5->getValue() * 10000)) - 10000 * 16.4612)) < 0.0001);
00090
00091     delete model;
00092     delete q1;
00093     delete q2;
00094     delete q3;
00095     delete q4;
00096     delete q5;
00097     delete f;
00098     delete t;
00099     delete u;
00100     delete v;
00101     delete g;
00102     delete r;
00103
00104     std::cout « "Passed Complex funcional test" « std::endl;
00105 }
```

References ModelIMP::add(), SystemIMP::getValue(), and ModelIMP::run().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

5.19.1.2 exponencial_test_run()

```
void exponencial_test_run ( )
```

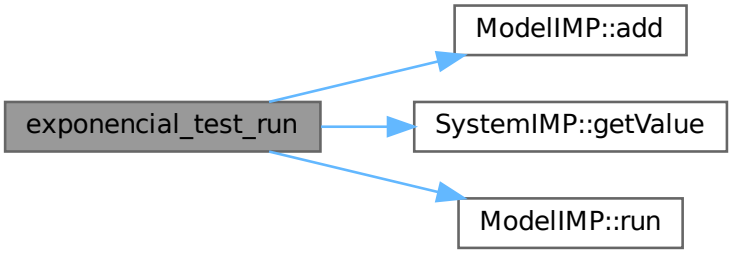
This function performs the exponential functional test.

```
00003                           {
00004     std::cout « "Exponencial funcional test" « std::endl;
00005
00006     SystemIMP* pop1 = new SystemIMP("pop1", 100.0);
00007     SystemIMP* pop2 = new SystemIMP("pop2", 0.0);
00008     Exponencial* exp = new Exponencial("exp", pop1, pop2);
00009     ModelIMP* exponencial = new ModelIMP("Exponencial", 0, 100);
00010
00011     //Add os systems e flows ao modelo
00012     exponencial->add(pop1);
00013     exponencial->add(pop2);
00014     exponencial->add(exp);
00015
00016     //Roda o modelo
00017     exponencial->run();
00018
00019     assert(fabs((round(pop1->getValue() * 10000) - 10000 * 36.6032)) < 0.0001);
00020     assert(fabs((round(pop2->getValue() * 10000) - 10000 * 63.3968)) < 0.0001);
00021
00022     delete(exponencial);
00023     delete(exp);
00024     delete(pop1);
00025     delete(pop2);
00026
00027     std::cout « "Passed exponencial funcional test" « std::endl;
00028 }
```

References ModelIMP::add(), SystemIMP::getValue(), and ModelIMP::run().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

5.19.1.3 logistical_test_run()

```
void logistical_test_run ( )
```

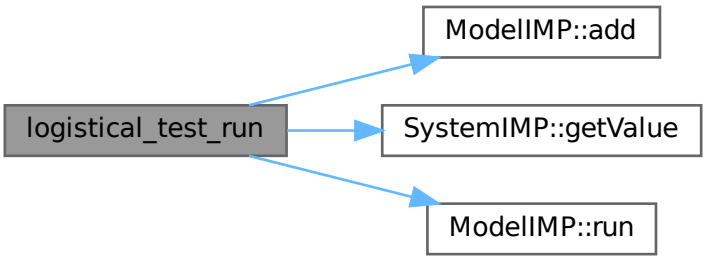
This function performs the logistic test.

```
00030                           {
00031     std::cout « "Logistical funcional test" « std::endl;
00032
00033     SystemIMP* p1 = new SystemIMP("p1", 100.0);
00034     SystemIMP* p2 = new SystemIMP("p2", 10.0);
00035     Logistical* log = new Logistical("log", p1, p2);
00036     ModelIMP* logistical = new ModelIMP("Logistical", 0, 100);
00037
00038     //Add os systems e flows ao modelo
00039     logistical->add(p1);
00040     logistical->add(p2);
00041     logistical->add(log);
00042
00043     //Roda o modelo
00044     logistical->run();
00045
00046     assert(fabs(round(p1->getValue() * 10000) - 10000 * 88.2167) < 0.0001);
00047     assert(fabs(round(p2->getValue() * 10000) - 10000 * 21.7833) < 0.0001);
00048
00049     delete(logistical);
00050     delete(log);
00051     delete(p1);
00052     delete(p2);
00053 }
```

References ModelIMP::add(), SystemIMP::getValue(), and ModelIMP::run().

Referenced by main().

Here is the call graph for this function:

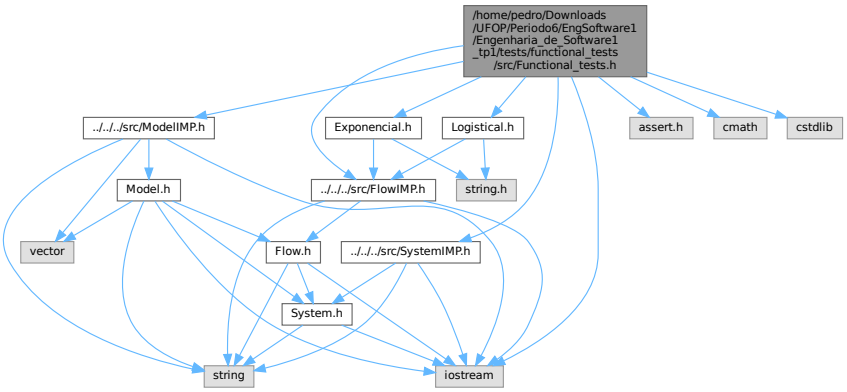


Here is the caller graph for this function:

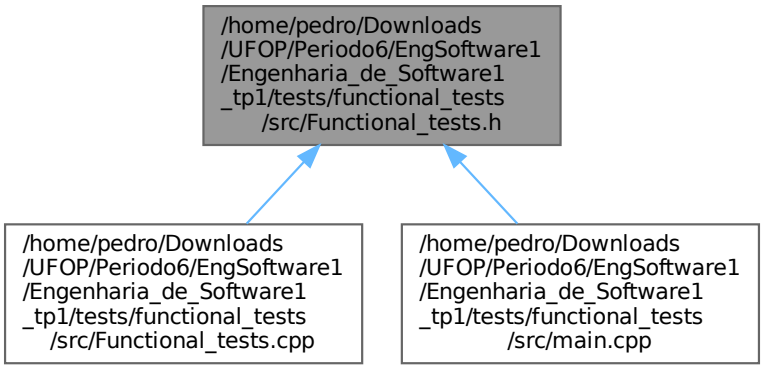
5.20 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Functional_tests.h File Reference

```
#include "../../../src/ModelIMP.h"
#include "../../../src/SystemIMP.h"
#include "../../../src/FlowIMP.h"
#include "Exponencial.h"
#include "Logistical.h"
#include <assert.h>
#include <cmath>
#include <iostream>
#include <cstdlib>
```

Include dependency graph for Functional_tests.h:



This graph shows which files directly or indirectly include this file:



Functions

- void exponencial_test_run ()

This function performs the exponential functional test.

- void logistical_test_run ()

  This function performs the logistic test.

- void Complex_test_run ()

  This function runs the "complex" test, which has multiple systems and flows.

5.20.1 Function Documentation

5.20.1.1 Complex_test_run()

```
void Complex_test_run ( )
```

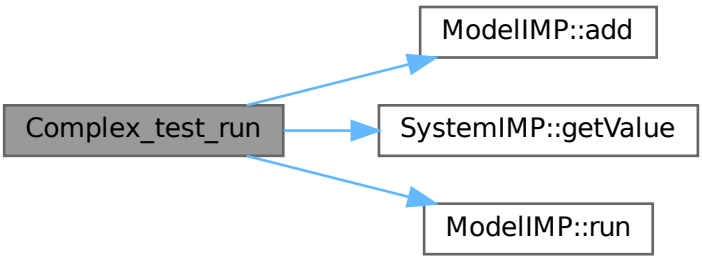
This function runs the "complex" test, which has multiple systems and flows.

```
00055                           {
00056     std::cout « "Complex funcional test" « std::endl;
00057
00058     ModelIMP* model = new ModelIMP("Model", 0, 100);
00059     SystemIMP* q1 = new SystemIMP("q1", 100.0);
00060     SystemIMP* q2 = new SystemIMP("q2", 0.0);
00061     SystemIMP* q3 = new SystemIMP("q3", 100.0);
00062     SystemIMP* q4 = new SystemIMP("q4", 0.0);
00063     SystemIMP* q5 = new SystemIMP("q5", 0.0);
00064     Exponencial* f = new Exponencial("f", q1, q2);
00065     Exponencial* t = new Exponencial("t", q2, q3);
00066     Exponencial* u = new Exponencial("u", q3, q4);
00067     Exponencial* v = new Exponencial("v", q4, q1);
00068     Exponencial* g = new Exponencial("g", q1, q3);
00069     Exponencial* r = new Exponencial("r", q2, q5);
00070
00071     model->add(q1);
00072     model->add(q2);
00073     model->add(q3);
00074     model->add(q4);
00075     model->add(q5);
00076     model->add(f);
00077     model->add(t);
00078     model->add(u);
00079     model->add(v);
00080     model->add(g);
00081     model->add(r);
00082
00083     model->run();
00084
00085     assert(fabs((round((q1->getValue() * 10000)) - 10000 * 31.8513)) < 0.0001);
00086     assert(fabs((round((q2->getValue() * 10000)) - 10000 * 18.4003)) < 0.0001);
00087     assert(fabs((round((q3->getValue() * 10000)) - 10000 * 77.1143)) < 0.0001);
00088     assert(fabs((round((q4->getValue() * 10000)) - 10000 * 56.1728)) < 0.0001);
00089     assert(fabs((round((q5->getValue() * 10000)) - 10000 * 16.4612)) < 0.0001);
00090
00091     delete model;
00092     delete q1;
00093     delete q2;
00094     delete q3;
00095     delete q4;
00096     delete q5;
00097     delete f;
00098     delete t;
00099     delete u;
00100     delete v;
00101     delete g;
00102     delete r;
00103
00104     std::cout « "Passed Complex funcional test" « std::endl;
00105 }
```

References ModelIMP::add(), SystemIMP::getValue(), and ModelIMP::run().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

5.20.1.2 exponencial_test_run()

```
void exponencial_test_run ( )
```

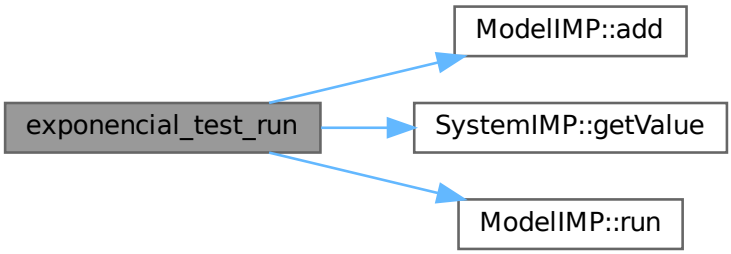
This function performs the exponential functional test.

```
00003                           {
00004     std::cout « "Exponencial funcional test" « std::endl;
00005
00006     SystemIMP* pop1 = new SystemIMP("pop1", 100.0);
00007     SystemIMP* pop2 = new SystemIMP("pop2", 0.0);
00008     Exponencial* exp = new Exponencial("exp", pop1, pop2);
00009     ModelIMP* exponencial = new ModelIMP("Exponencial", 0, 100);
00010
00011     //Add os systems e flows ao modelo
00012     exponencial->add(pop1);
00013     exponencial->add(pop2);
00014     exponencial->add(exp);
00015
00016     //Roda o modelo
00017     exponencial->run();
00018
00019     assert(fabs((round(pop1->getValue() * 10000) - 10000 * 36.6032)) < 0.0001);
00020     assert(fabs((round(pop2->getValue() * 10000) - 10000 * 63.3968)) < 0.0001);
00021
00022     delete(exponencial);
00023     delete(exp);
00024     delete(pop1);
00025     delete(pop2);
00026
00027     std::cout « "Passed exponencial funcional test" « std::endl;
00028 }
```

References ModelIMP::add(), SystemIMP::getValue(), and ModelIMP::run().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

5.20.1.3 logistical_test_run()

```
void logistical_test_run ( )
```

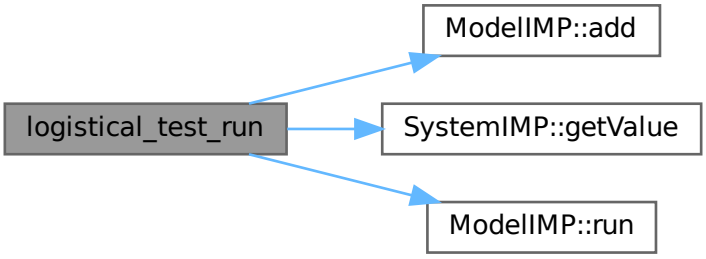
This function performs the logistic test.

```
00030                           {
00031     std::cout « "Logistical funcional test" « std::endl;
00032
00033     SystemIMP* p1 = new SystemIMP("p1", 100.0);
00034     SystemIMP* p2 = new SystemIMP("p2", 10.0);
00035     Logistical* log = new Logistical("log", p1, p2);
00036     ModelIMP* logistical = new ModelIMP("Logistical", 0, 100);
00037
00038     //Add os systems e flows ao modelo
00039     logistical->add(p1);
00040     logistical->add(p2);
00041     logistical->add(log);
00042
00043     //Roda o modelo
00044     logistical->run();
00045
00046     assert(fabs(round(p1->getValue() * 10000) - 10000 * 88.2167) < 0.0001);
00047     assert(fabs(round(p2->getValue() * 10000) - 10000 * 21.7833) < 0.0001);
00048
00049     delete(logistical);
00050     delete(log);
00051     delete(p1);
00052     delete(p2);
00053 }
```

References ModelIMP::add(), SystemIMP::getValue(), and ModelIMP::run().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

5.21 Functional_tests.h

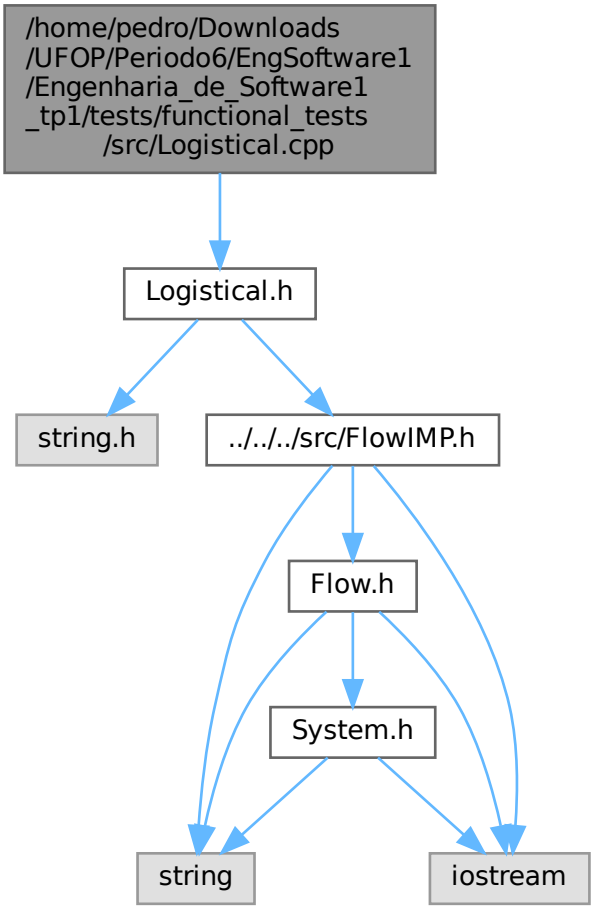
Go to the documentation of this file.

```
00001 /*************************************************************
00002  * @file Exponencial.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the logistical simulation flow
00005 *************************************************************/
00006
00007 #ifndef FUNCTIONAL_TESTS_H
00008 #define FUNCTIONAL_TESTS_H
00009
00010 #include "../../../src/ModelIMP.h"
00011 #include "../../../src/SystemIMP.h"
00012 #include "../../../src/FlowIMP.h"
00013 #include "Exponencial.h"
00014 #include "Logistical.h"
00015 #include <assert.h>
00016 #include <cmath>
00017 #include <iostream>
00018 #include <cstdlib>
00019
00020 /************************************
00021  * @brief execution of functional tests
00022 ************************************/
00023
00027 void exponencial_test_run();
00028
00032 void logistical_test_run();
00033
00037 void Complex_test_run();
00038
00039 #endif
```

5.22 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Logistical.cpp File Reference

```
#include "Logistical.h"
```

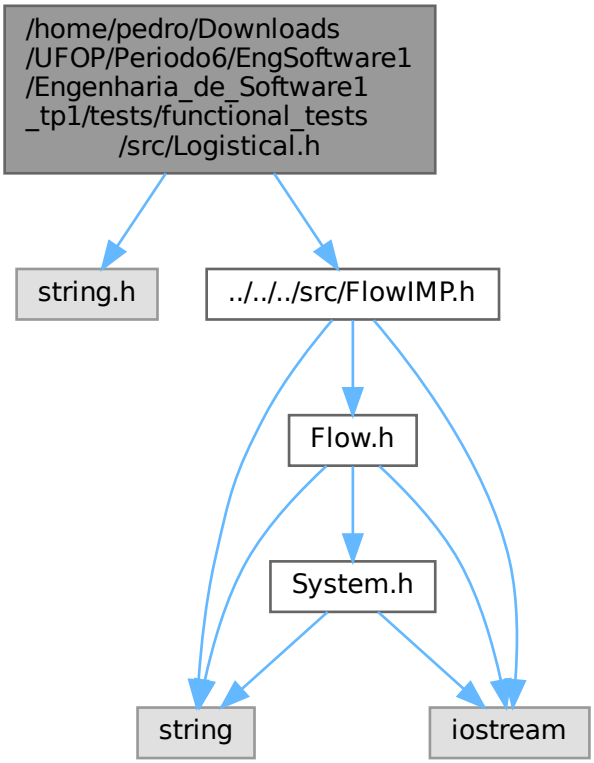
Include dependency graph for Logistical.cpp:



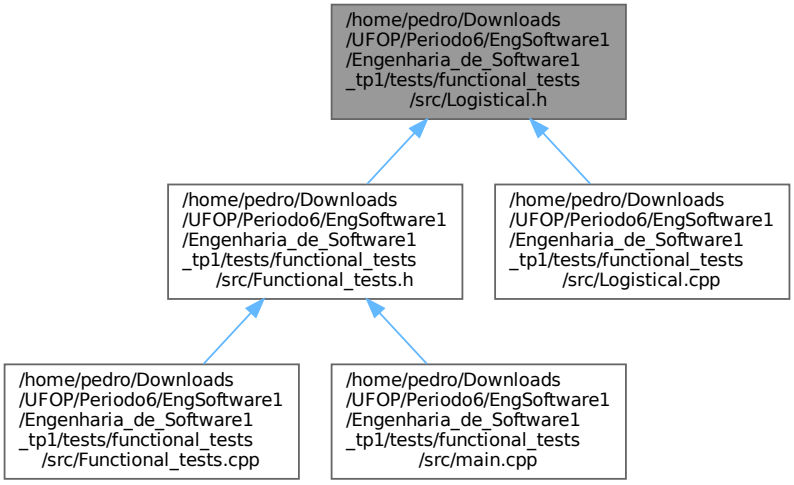
5.23 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Logistical.h File Reference

```
#include <string.h>
#include "../../../src/FlowIMP.h"
```

Include dependency graph for Logistical.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Logistical

5.24 Logistical.h

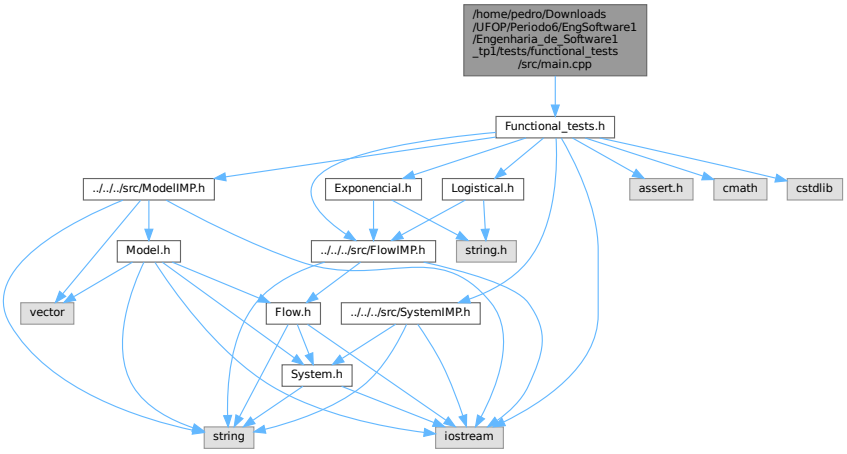
Go to the documentation of this file.

```
00001 /************************************************************
00002  * @file Logistical.h
00003  * @author Pedro Augusto Sousa Gonçalves
00004  * @brief This file represents the logistical simulation flow
00005 ************************************************************/
00006
00007 #ifndef LOGISTICAL_DEF
00008 #define LOGISTICAL_DEF
00009
00010 #include <string.h>
00011 #include "../../../src/FlowIMP.h"
00012
00013 class Logistical : public FlowIMP{
00014     private:
00019         Logistical(const Logistical& other);
00020
00021     public:
00022         //Contructor
00029         Logistical(const std::string& name = "NO_NAME", System* source = NULL, System* target = NULL);
00030
00031         //Destructor
00035         virtual ~Logistical();
00036
00037         //Metodos
00042         virtual double execute() override;
00043 };
00044
00045 #endif
```

5.25 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/main.cpp File Reference

```
#include "Functional_tests.h"
```

Include dependency graph for main.cpp:



Functions

- int main ()

5.25.1 Function Documentation

5.25.1.1 main()

```
int main ( )
00003           {
00004     exponencial_test_run();
00005     logistical_test_run();
00006     Complex_test_run();
00007     return 0;
00008 }
```

References Complex_test_run(), exponencial_test_run(), and logistical_test_run().

Here is the call graph for this function: