My Vensin

Generated by Doxygen 1.10.0

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 Exponencial Class Reference 7

4.1.1 Constructor & Destructor Documentation 9

4.1.1.1 Exponencial() `[1/2]` 9

4.1.1.2 Exponencial() `[2/2]` 9

4.1.1.3 ∼Exponencial() 9

4.1.2 Member Function Documentation 9

4.1.2.1 execute() 9

4.2 Flow Class Reference 10

4.2.1 Member Function Documentation 11

4.2.1.1 execute() 11

4.2.1.2 getName() 11

4.2.1.3 getSource() 11

4.2.1.4 getTarget() 11

4.2.1.5 operator=() 11

4.2.1.6 operator==() 11

4.2.1.7 setName() 11

4.2.1.8 setSource() 12

4.2.1.9 setTarget() 12

4.2.2 Friends And Related Symbol Documentation 12

4.2.2.1 operator<< 12

4.2.3 Member Data Documentation 12

4.2.3.1 name 12

4.2.3.2 source 12

4.2.3.3 target 12

4.3 Logistical Class Reference 13

4.3.1 Constructor & Destructor Documentation 14

4.3.1.1 Logistical() `[1/2]` 14

4.3.1.2 Logistical() `[2/2]` 14

4.3.1.3 ∼Logistical() 14

4.3.2 Member Function Documentation 15

4.3.2.1 execute() 15

4.4 Model Class Reference 15

4.4.1 Constructor & Destructor Documentation 16

4.4.1.1 Model() 16
4.4.1.2 ∼Model() 16
4.4.2 Member Function Documentation 16
4.4.2.1 add() [1/2] 16
4.4.2.2 add() [2/2] 16
4.4.2.3 getEndtTime() 16
4.4.2.4 getName() 16
4.4.2.5 getStartTime() 16
4.4.2.6 operator==() 17
4.4.2.7 rmv() [1/2] 17
4.4.2.8 rmv() [2/2] 17
4.4.2.9 run() 17
4.4.2.10 setEndTime() 17
4.4.2.11 setName() 18
4.4.2.12 setStartTime() 18
4.4.2.13 setTime() 18
4.4.3 Friends And Related Symbol Documentation 18
4.4.3.1 operator<< 18
4.4.4 Member Data Documentation 18
4.4.4.1 endTime 18
4.4.4.2 flows 18
4.4.4.3 name 18
4.4.4.4 startTime 19
4.4.4.5 systems 19
4.5 System Class Reference 19
4.5.1 Constructor & Destructor Documentation 19
4.5.1.1 System() [1/2] 19
4.5.1.2 System() [2/2] 20
4.5.1.3 ∼System() 20
4.5.2 Member Function Documentation 20
4.5.2.1 getName() 20
4.5.2.2 getValue() 20
4.5.2.3 operator=() 20
4.5.2.4 operator==() 20
4.5.2.5 setName() 20
4.5.2.6 setValue() 21
4.5.3 Friends And Related Symbol Documentation 21
4.5.3.1 operator<< 21
4.5.4 Member Data Documentation 21
4.5.4.1 name 21
4.5.4.2 value 21

5 File Documentation **23**

5.1 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.cpp File Reference 23

5.1.1 Function Documentation 24

5.1.1.1 operator<<() 24

5.2 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.h File Reference 24

5.3 Flow.h 25

5.4 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/↩ Model.cpp File Reference 26

5.4.1 Function Documentation 26

5.4.1.1 operator<<() 26

5.5 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.h File Reference 27

5.5.1 Typedef Documentation 28

5.5.1.1 flowIterator 28

5.5.1.2 systemIterator 28

5.6 Model.h 29

5.7 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/↩ System.cpp File Reference 30

5.7.1 Function Documentation 30

5.7.1.1 operator<<() 30

5.8 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/↩ System.h File Reference 30

5.9 System.h 31

5.10 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩ _tests/src/Exponencial.cpp File Reference 32

5.11 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩ _tests/src/Exponencial.h File Reference 33

5.12 Exponencial.h 34

5.13 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩ _tests/src/Functional_tests.cpp File Reference 35

5.13.1 Function Documentation 35

5.13.1.1 Complex_test_run() 35

5.13.1.2 exponencial_test_run() 36

5.13.1.3 logistical_test_run() 36

5.14 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩ _tests/src/Functional_tests.h File Reference 37

5.14.1 Function Documentation 38

5.14.1.1 Complex_test_run() 38

5.14.1.2 exponencial_test_run() 39

5.14.1.3 logistical_test_run() 39

5.15 Functional_tests.h 40

5.16 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩ _tests/src/Logistical.cpp File Reference 40

5.17 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩
_tests/src/Logistical.h File Reference 41
5.18 Logistical.h 42
5.19 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩
_tests/src/main.cpp File Reference 43
5.19.1 Function Documentation 43
5.19.1.1 main() 43

Index **45**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Flow 10
    Exponencial 7
    Logistical 13
Model 15
System 19

Chapter 2

Class Index

2.1 Class List

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

Exponencial 7
Flow 10
Logistical 13
Model 15
System 19

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.cpp . . 23
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.h . . . 24
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.cpp . 26
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.h . . 27
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/System.cpp 30
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/System.h . 30
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/Exponencial.cpp 32
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/Exponencial.h 33
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/Functional_tests.cpp 35
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/Functional_tests.h 37
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/Logistical.cpp 40
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/Logistical.h 41
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional↩_tests/src/main.cpp 43

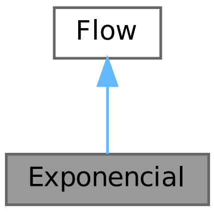
Chapter 4

Class Documentation

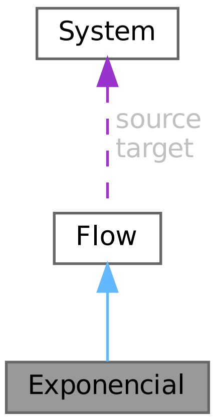
4.1 Exponencial Class Reference

```
#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)
- Exponencial (const Exponencial &other)
- virtual ∼Exponencial ()
- virtual double execute () override

Public Member Functions inherited from Flow

- std::string getName () const
- void setName (std::string &name)
- System ∗ getSource () const
- void setSource (System ∗source)
- System ∗ getTarget () const
- void setTarget (System ∗target)
- Flow & operator= (const Flow &other)
- bool operator== (const Flow &other) const

Additional Inherited Members

Protected Attributes inherited from Flow

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

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Exponencial() [1/2]

```
Exponencial::Exponencial (
            const std::string & name = "NO_NAME",
            System * source = NULL,
            System * target = NULL )
{
    this->name = name;
    this->source = source;
    this->target = target;
}
```

4.1.1.2 Exponencial() [2/2]

```
Exponencial::Exponencial (
            const Exponencial & other )
{
    this->name = other.name;
    this->source = other.source;
    this->target = other.target;
}
```

4.1.1.3 ∼Exponencial()

```
Exponencial::∼Exponencial ( ) [virtual]
{}
```

4.1.2 Member Function Documentation

4.1.2.1 execute()

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

Implements Flow.

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

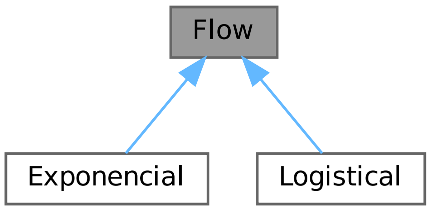
The documentation for this class was generated from the following files:

- /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

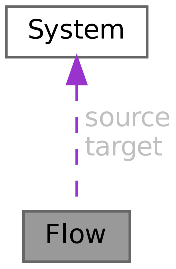
4.2 Flow Class Reference

`#include <Flow.h>`

Inheritance diagram for Flow:



Collaboration diagram for Flow:



Public Member Functions

- std::string getName () const
- void setName (std::string &name)
- System ∗ getSource () const
- void setSource (System ∗source)
- System ∗ getTarget () const
- void setTarget (System ∗target)
- virtual double execute ()=0
- Flow & operator= (const Flow &other)
- bool operator== (const Flow &other) const

Protected Attributes

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

Friends

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

4.2.1 Member Function Documentation

4.2.1.1 execute()

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

Implemented in Exponencial, and Logistical.

4.2.1.2 getName()

```
std::string Flow::getName ( ) const
{ return name; }
```

4.2.1.3 getSource()

```
System * Flow::getSource ( ) const
{ return source; }
```

4.2.1.4 getTarget()

```
System * Flow::getTarget ( ) const
{ return target; }
```

4.2.1.5 operator=()

```
Flow & Flow::operator= (
            const Flow & other )
                                            {
    if(other == *this) return *this;
    name = other.name;
    source = other.source;
    target = other.target;
    return *this;
}
```

4.2.1.6 operator==()

```
bool Flow::operator== (
            const Flow & other ) const
                                                  {
    return (name == other.name && source == other.source && target == other.target);
}
```

4.2.1.7 setName()

```
void Flow::setName (
            std::string & name )
{ this->name = name; }
```

4.2.1.8 setSource()

```
void Flow::setSource (
                System * source )
{ this->source = source; }
```

4.2.1.9 setTarget()

```
void Flow::setTarget (
                System * target )
{ this->target = target; }
```

4.2.2 Friends And Related Symbol Documentation

4.2.2.1 operator<<

```
std::ostream & operator<< (
                std::ostream & out,
                const Flow & obj )  [friend]
                                                        {
    out « "(Flow) Name: " « obj.name « " - "
        « obj.source->getName() « " -----> " « obj.target->getName();
    return out;
}
```

4.2.3 Member Data Documentation

4.2.3.1 name

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

4.2.3.2 source

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

4.2.3.3 target

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

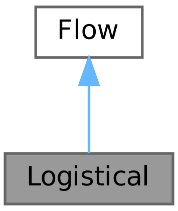
The documentation for this class was generated from the following files:

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

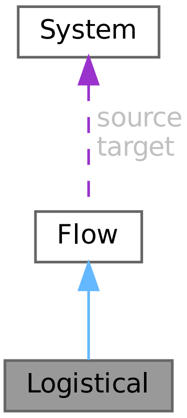
4.3 Logistical Class Reference

```
#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)
- Logistical (const Logistical &other)
- virtual ∼Logistical ()
- virtual double execute () override

Public Member Functions inherited from Flow

- std::string getName () const
- void setName (std::string &name)
- System ∗ getSource () const
- void setSource (System ∗source)
- System ∗ getTarget () const
- void setTarget (System ∗target)
- Flow & operator= (const Flow &other)
- bool operator== (const Flow &other) const

Additional Inherited Members

Protected Attributes inherited from Flow

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

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Logistical() [1/2]

```
Logistical::Logistical (
            const std::string & name = "NO_NAME",
            System * source = NULL,
            System * target = NULL )
{
    this->name = name;
    this->source = source;
    this->target = target;
}
```

4.3.1.2 Logistical() [2/2]

```
Logistical::Logistical (
            const Logistical & other )
{
    this->name = other.name;
    this->source = other.source;
    this->target = other.target;
}
```

4.3.1.3 ∼Logistical()

```
Logistical::∼Logistical ( )  [virtual]
{}
```

4.3.2 Member Function Documentation

4.3.2.1 execute()

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

Implements Flow.

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

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

4.4 Model Class Reference

```
#include <Model.h>
```

Public Member Functions

- Model (const std::string &name="NO_NAME", const int &startTime=0, const int &endTime=1)
- virtual ∼Model ()
- std::string getName () const
- void setName (const std::string &name)
- int getStartTime () const
- int getEndtTime () const
- void setStartTime (const int &startTime)
- void setEndTime (const int &endTime)
- void setTime (const int &startTime, const int &endTime)
- void add (System ∗system)
- void add (Flow ∗flow)
- bool rmv (const systemIterator &system)
- bool rmv (const flowIterator &flow)
- bool run ()
- bool operator== (const Model &other) const

Protected Attributes

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

Friends

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

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Model()

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

4.4.1.2 ∼Model()

```
Model::∼Model ( ) [virtual]
{systems.clear(); flows.clear();}
```

4.4.2 Member Function Documentation

4.4.2.1 add() [1/2]

```
void Model::add (
            Flow * flow )
{ flows.push_back(flow); }
```

4.4.2.2 add() [2/2]

```
void Model::add (
            System * system )
{ systems.push_back(system); }
```

4.4.2.3 getEndtTime()

```
int Model::getEndtTime ( ) const
{ return endTime; }
```

4.4.2.4 getName()

```
std::string Model::getName ( ) const
{ return name; }
```

4.4.2.5 getStartTime()

```
int Model::getStartTime ( ) const
{ return startTime; }
```

4.4.2.6 operator==()

```
bool Model::operator== (
            const Model & other ) const
                                    {
    return (name == other.name && systems == other.systems && flows == other.flows && startTime ==
 other.startTime && endTime == other.endTime);
}
```

4.4.2.7 rmv() [1/2]

```
bool Model::rmv (
            const flowIterator & flow )
{ return (flows.erase(flow) != flows.end()); }
```

4.4.2.8 rmv() [2/2]

```
bool Model::rmv (
            const systemIterator & system )
{ return (systems.erase(system) != systems.end()); }
```

4.4.2.9 run()

```
bool Model::run ( )
                {
    std::vector<double> flowValue;
    flowIterator f;
    std::vector<double>::iterator d;
    double calcValue;

    for(int i = startTime; i < endTime; i++){

        f = flows.begin();

        while (f != flows.end()) {
            flowValue.push_back((*f)->execute());
            f++;
        }

        f = flows.begin();
        d = flowValue.begin();

        while(f != flows.end()){
            calcValue = (*f)->getSource()->getValue() - (*d);
            (*f)->getSource()->setValue(calcValue);
            calcValue = (*f)->getTarget()->getValue() + (*d);
            (*f)->getTarget()->setValue(calcValue);
            f++;
            d++;
        }

        flowValue.clear();

    }

    return true;
}
```

4.4.2.10 setEndTime()

```
void Model::setEndTime (
            const int & endTime )
{ this->endTime = endTime; }
```

4.4.2.11 setName()

```
void Model::setName (
              const std::string & name )
{ this->name = name; }
```

4.4.2.12 setStartTime()

```
void Model::setStartTime (
              const int & startTime )
{ this->startTime = startTime; }
```

4.4.2.13 setTime()

```
void Model::setTime (
              const int & startTime,
              const int & endTime )
{ this->startTime = startTime; this->endTime = endTime; }
```

4.4.3 Friends And Related Symbol Documentation

4.4.3.1 operator<<

```
std::ostream & operator<< (
              std::ostream & out,
              const Model & obj )  [friend]
                                                      {
    out « "Name: " « obj.name « ";\n"
        « "Systems:\n";
    for (auto item : obj.systems) out « item « "\n";
    out « "Flows:\n";
    for (auto item : obj.flows) out « item « "\n";
    return out;
}
```

4.4.4 Member Data Documentation

4.4.4.1 endTime

```
int Model::endTime  [protected]
```

4.4.4.2 flows

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

4.4.4.3 name

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

4.4.4.4 startTime

```
int Model::startTime  [protected]
```

4.4.4.5 systems

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

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

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

4.5 System Class Reference

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

Public Member Functions

- System (const std::string &name="NO_NAME", const double &value=0.0)
- System (const System &other)
- virtual ∼System ()
- std::string getName () const
- void setName (const std::string &name)
- double getValue () const
- void setValue (const double &value)
- System & operator= (const System &other)
- bool operator== (const System &other) const

Protected Attributes

- std::string name
- double value

Friends

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

4.5.1 Constructor & Destructor Documentation

4.5.1.1 System() [1/2]

```
System::System (
            const std::string & name = "NO_NAME",
            const double & value = 0.0 )
00004 : name(name), value(value) {}
```

4.5.1.2 System() [2/2]

```
System::System (
                const System & other )
: name(other.name), value(other.value) {}
```

4.5.1.3 ∼System()

```
System::∼System ( )  [virtual]
{};
```

4.5.2 Member Function Documentation

4.5.2.1 getName()

```
std::string System::getName ( ) const
{ return name; }
```

4.5.2.2 getValue()

```
double System::getValue ( ) const
{ return value; }
```

4.5.2.3 operator=()

```
System & System::operator= (
                const System & other )
                                             {
    if(other == *this) return *this;
    name = other.name;
    value = other.value;
    return *this;
}
```

4.5.2.4 operator==()

```
bool System::operator== (
                const System & other ) const
                                                 {
    return (name == other.name && value == other.value);
    // Compare todos os membros para verificar igualdade
}
```

4.5.2.5 setName()

```
void System::setName (
                const std::string & name )
{ this->name = name; }
```

4.5.2.6 setValue()

```
void System::setValue (
             const double & value )
{ this->value = value; }
```

4.5.3 Friends And Related Symbol Documentation

4.5.3.1 operator<<

```
std::ostream & operator<< (
             std::ostream & out,
             const System & obj )  [friend]
                                                        {
    out « "(System)(Name: " « obj.name « ", Value: " « obj.value « ")";
    return out;
}
```

4.5.4 Member Data Documentation

4.5.4.1 name

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

4.5.4.2 value

```
double System::value  [protected]
```

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

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

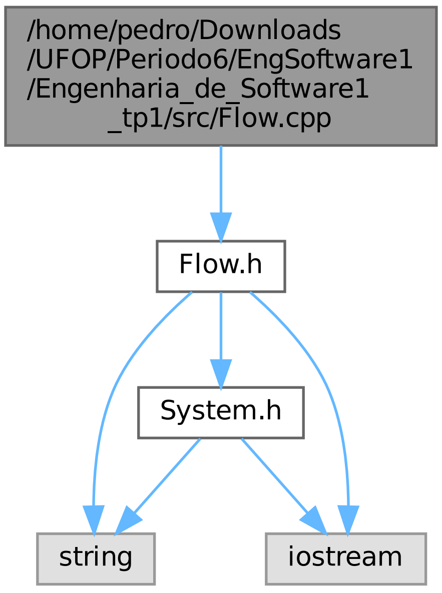
Chapter 5

File Documentation

5.1 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia↩ _de_Software1_tp1/src/Flow.cpp File Reference

```
#include "Flow.h"
```

Include dependency graph for Flow.cpp:



Functions

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

5.1.1 Function Documentation

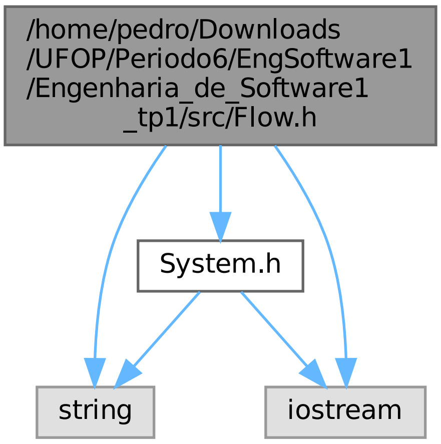
5.1.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & out,
            const Flow & obj )
                                                  {
    out « "(Flow) Name: " « obj.name « " - "
        « obj.source->getName() « " -----> " « obj.target->getName();
    return out;
}
```

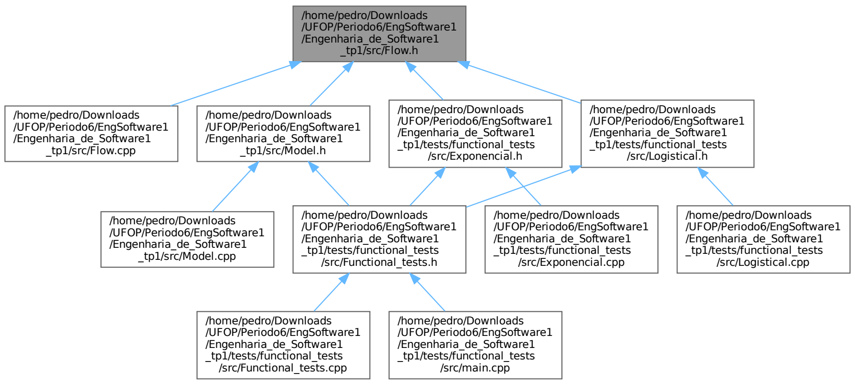
5.2 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.h File Reference

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

Include dependency graph for Flow.h:



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


/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/Flow.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/Flow.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/Model.h
/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/Logistical.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/Model.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Functional_tests.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Exponencial.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Logistical.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Functional_tests.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/main.cpp

Classes

- class Flow

5.3 Flow.h

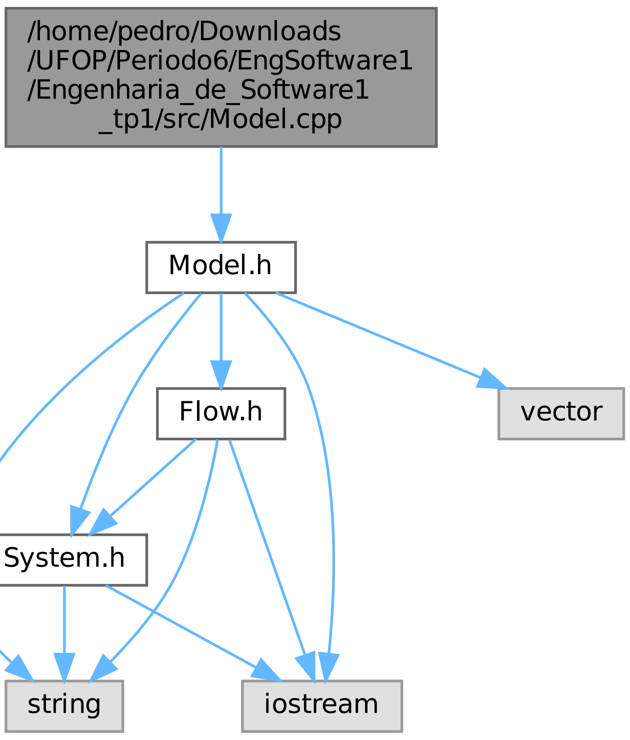
Go to the documentation of this file.

```
#ifndef FLOW_H
#define FLOW_H

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


class Flow{
    protected:
        std::string name;
        System* source;
        System* target;

    public:
        //Geters e seters
        //Name
        std::string getName() const;
        void setName(std::string& name);
        //Source
        System* getSource() const;
        void setSource(System* source);
        //Target
        System* getTarget() const;
        void setTarget(System* target);

        //Metodos
        virtual double execute() = 0;

        //Sobrecarga de operadores
        Flow& operator=(const Flow& other); // Operador de atribuição
        bool operator==(const Flow& other) const; // Operador de igualdade
        friend std::ostream& operator«(std::ostream& out, const Flow& obj); //Operador de saida
};


#endif
```

5.4 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia←_de_Software1_tp1/src/Model.cpp File Reference

`#include "Model.h"`

Include dependency graph for Model.cpp:



Functions

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

5.4.1 Function Documentation

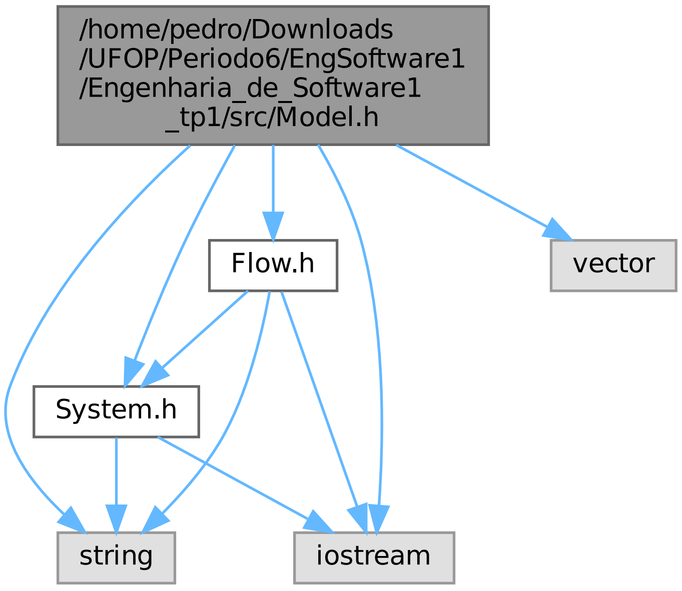
5.4.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & out,
            const Model & obj )
{
    out « "Name: " « obj.name « ";\n"
        « "Systems:\n";
    for (auto item : obj.systems) out « item « "\n";
    out « "Flows:\n";
    for (auto item : obj.flows) out « item « "\n";
    return out;
}
```

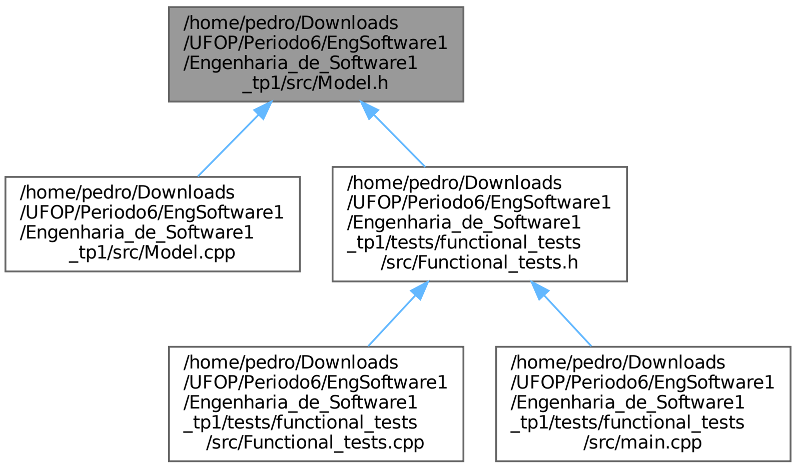
5.5 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia↩ _de_Software1_tp1/src/Model.h File Reference

```
#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:


/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.h
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.cpp
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Functional_tests.h
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Functional_tests.cpp
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/main.cpp

Classes

- class Model

Typedefs

- typedef std::vector< System ∗ >::iterator systemIterator
- typedef std::vector< Flow ∗ >::iterator flowIterator

5.5.1 Typedef Documentation

5.5.1.1 flowIterator

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

5.5.1.2 systemIterator

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

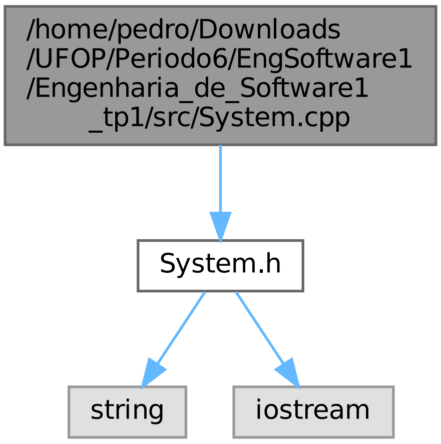
5.6 Model.h

Go to the documentation of this file.

```
#ifndef MODEL_H
#define MODEL_H

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

typedef std::vector<System*>::iterator systemIterator;
typedef std::vector<Flow*>::iterator flowIterator;

class Model{
    protected:
        std::string name;
        std::vector<System*> systems;
        std::vector<Flow*> flows;
        int startTime;
        int endTime;

    private:
        Model& operator=(const Model& other); // Operador de atribuição
        Model(const Model& other); //Copia outro flow

    public:
        //Contructors
        Model(const std::string& name = "NO_NAME", const int& startTime = 0, const int& endTime = 1);

        //Destrutor
        virtual ~Model();

        //Geters e seters
        //Name
        std::string getName() const;
        void setName(const std::string& name);
        //Time
        int getStartTime() const;
        int getEndtTime() const;
        void setStartTime(const int& startTime);
        void setEndTime(const int& endTime);
        void setTime(const int& startTime, const int& endTime);

        //Metodos
        //add
        void add(System* system);
        void add(Flow* flow);
        //remove
        bool rmv(const systemIterator& system);
        bool rmv(const flowIterator& flow);
        //Others
        bool run();

        bool operator==(const Model& other) const; // Operador de igualdade
        friend std::ostream& operator<<(std::ostream& out, const Model& obj); //Operador de saida
};

#endif
```

5.7 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia↩ _de_Software1_tp1/src/System.cpp File Reference

```
#include "System.h"
```
Include dependency graph for System.cpp:


/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/System.cpp
System.h
string
iostream

Functions

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

5.7.1 Function Documentation

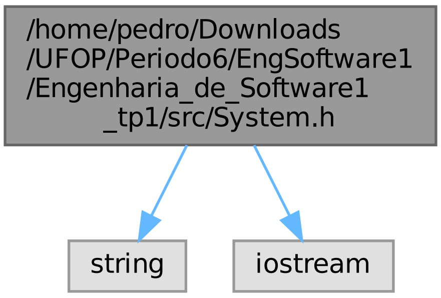
5.7.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & out,
            const System & obj )
                                                          {
    out « "(System)(Name: " « obj.name « ", Value: " « obj.value « ")";
    return out;
}
```

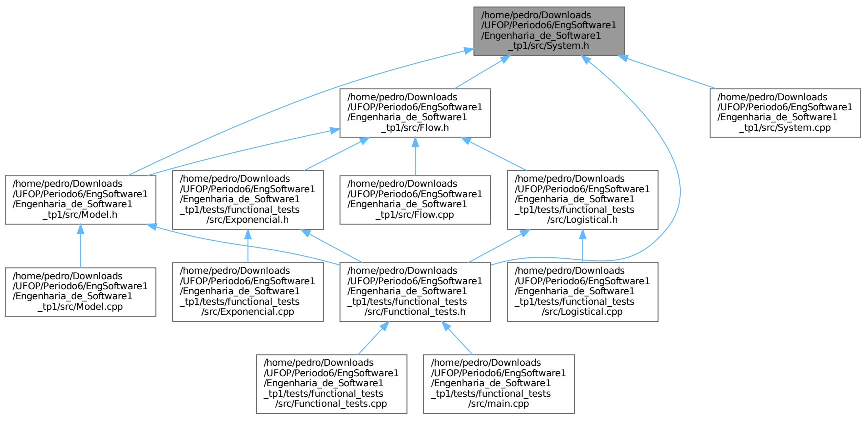
5.8 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia↩ _de_Software1_tp1/src/System.h File Reference

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

Include dependency graph for System.h:



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


/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/System.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/Flow.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/System.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/src/Model.h
/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/src/Flow.cpp
/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/src/Model.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Exponencial.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Functional_tests.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Logistical.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Functional_tests.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/main.cpp

Classes

- class System

5.9 System.h

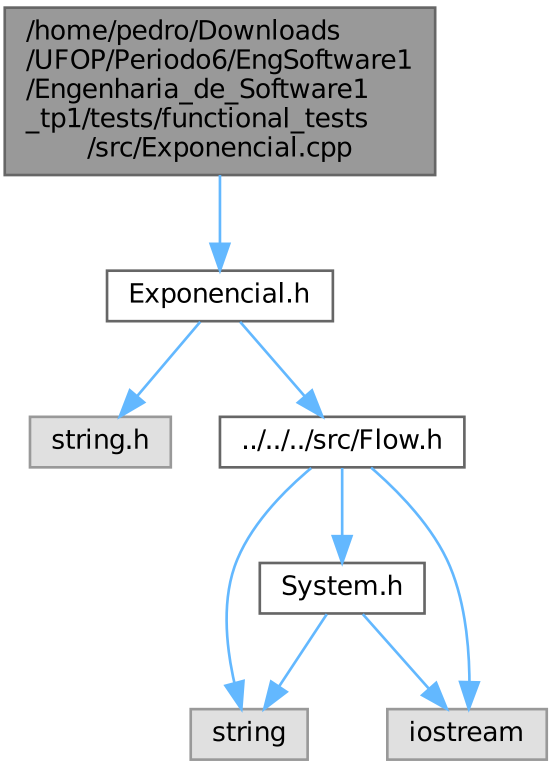
Go to the documentation of this file.

```
#ifndef SYSTEM_H
#define SYSTEM_H

//Bibliotecas
#include <string>
#include <iostream>

class System{
    protected:
        std::string name;
        double value;

    public:
        //Contructors
        System(const std::string& name = "NO_NAME", const double& value = 0.0);
        System(const System& other); //Copia outro system

        //Destructors
        virtual ~System();

        //Geters e seters
        //Nome
        std::string getName() const;
        void setName(const std::string& name);
        //Value
        double getValue() const;
        void setValue(const double& value);

        //Sobrecarga de operadores
        System& operator=(const System& other);  // Operador de atribuição
        bool operator==(const System& other) const;  // Operador de igualdade
        friend std::ostream& operator«(std::ostream& out, const System& obj); //Operador de saida
};

#endif
```

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

`#include "Exponencial.h"`

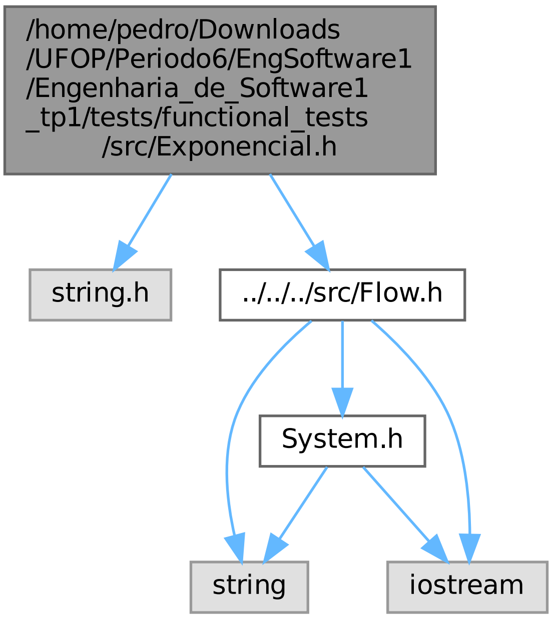
Include dependency graph for Exponencial.cpp:



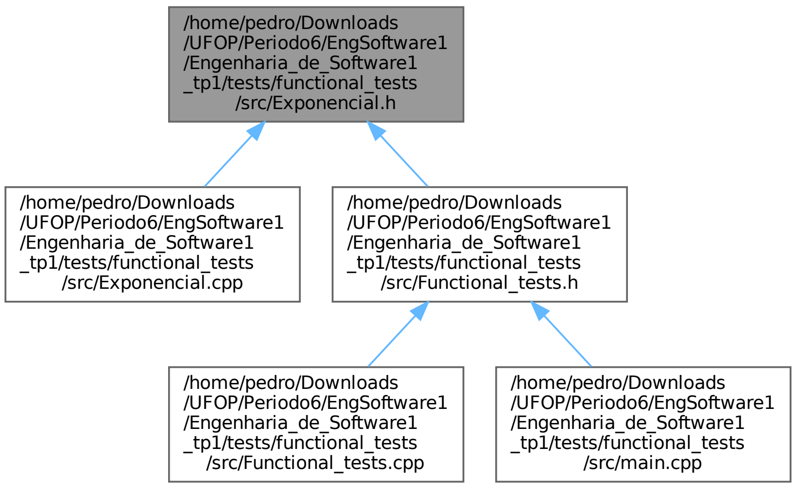
5.11 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Exponencial.h File Reference

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

Include dependency graph for Exponencial.h:



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



Classes

- class Exponencial

5.12 Exponencial.h

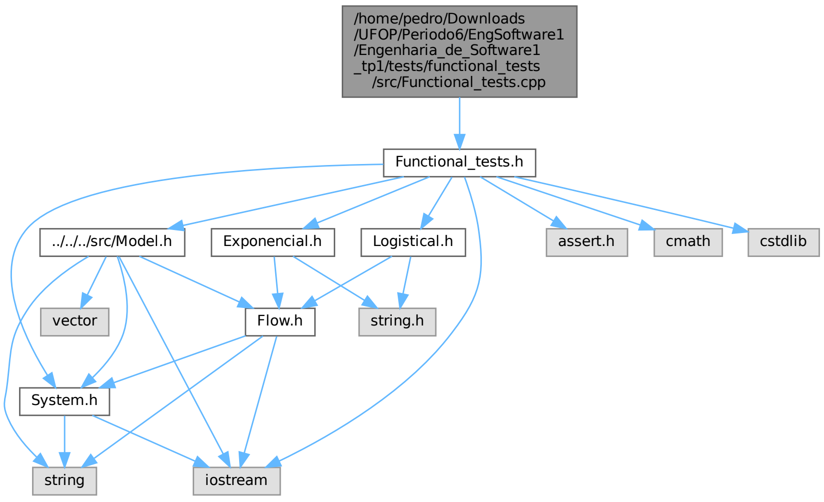
Go to the documentation of this file.

```
#ifndef EXPONENCIAL_DEF
#define EXPONENCIAL_DEF

#include <string.h>
#include "../../../src/Flow.h"
class Exponencial : public Flow{
    public:
        //Contructor
        Exponencial(const std::string& name = "NO_NAME", System* source = NULL, System* target =
      NULL);
        Exponencial(const Exponencial& other);
        //Destructor
        virtual ~Exponencial();

        //Metodos
        virtual double execute() override;
};

#endif
```

5.13 /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 ()
- void logistical_test_run ()
- void Complex_test_run ()

5.13.1 Function Documentation

5.13.1.1 Complex_test_run()

```
void Complex_test_run ( )
                    {
    std::cout « "Complex funcional test" « std::endl;

    Model* model = new Model("Model", 0, 100);
    System* q1 = new System("q1", 100.0);
    System* q2 = new System("q2", 0.0);
    System* q3 = new System("q3", 100.0);
    System* q4 = new System("q4", 0.0);
    System* q5 = new System("q5", 0.0);
    Exponencial* f = new Exponencial("f", q1, q2);
    Exponencial* t = new Exponencial("t", q2, q3);
    Exponencial* u = new Exponencial("u", q3, q4);
    Exponencial* v = new Exponencial("v", q4, q1);
    Exponencial* g = new Exponencial("g", q1, q3);
    Exponencial* r = new Exponencial("r", q2, q5);

    model->add(q1);
    model->add(q2);
    model->add(q3);
    model->add(q4);
    model->add(q5);
    model->add(f);
    model->add(t);
    model->add(u);
    model->add(v);
    model->add(g);
    model->add(r);

    model->run();

    assert(fabs((round((q1->getValue() * 10000)) - 10000 * 31.8513)) < 0.0001);
    assert(fabs((round((q2->getValue() * 10000)) - 10000 * 18.4003)) < 0.0001);
    assert(fabs((round((q3->getValue() * 10000)) - 10000 * 77.1143)) < 0.0001);
    assert(fabs((round((q4->getValue() * 10000)) - 10000 * 56.1728)) < 0.0001);
    assert(fabs((round((q5->getValue() * 10000)) - 10000 * 16.4612)) < 0.0001);

    delete model;
    delete q1;
    delete q2;
    delete q3;
    delete q4;
    delete q5;
    delete f;
    delete t;
    delete u;
    delete v;
    delete g;
    delete r;

    std::cout « "Passed Complex funcional test" « std::endl;
}
```

5.13.1.2 exponencial_test_run()

void exponencial_test_run ( )

```
                             {
    std::cout « "Exponencial funcional test" « std::endl;

    System* pop1 = new System("pop1", 100.0);
    System* pop2 = new System("pop2", 0.0);
    Exponencial* exp = new Exponencial("exp", pop1, pop2);
    Model* exponencial = new Model("Exponencial", 0, 100);

    //Add os systems e flows ao modelo
    exponencial->add(pop1);
    exponencial->add(pop2);
    exponencial->add(exp);

    //Roda o modelo
    exponencial->run();

    assert(fabs((round(pop1->getValue() * 10000) - 10000 * 36.6032)) < 0.0001);
    assert(fabs((round(pop2->getValue() * 10000) - 10000 * 63.3968)) < 0.0001);

    delete(exponencial);
    delete(exp);
    delete(pop1);
    delete(pop2);

    std::cout « "Passed exponencial funcional test" « std::endl;
}
```

5.13.1.3 logistical_test_run()

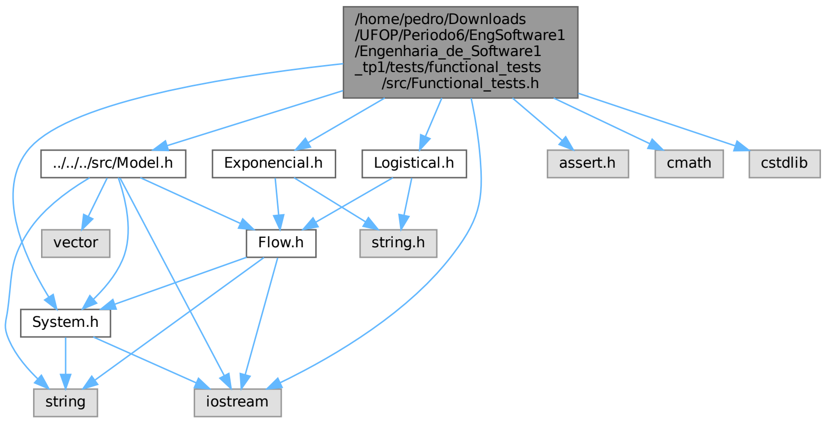
void logistical_test_run ( )

```
                            {
    std::cout « "Logistical funcional test" « std::endl;

    System* p1 = new System("p1", 100.0);
    System* p2 = new System("p2", 10.0);
    Logistical* log = new Logistical("log", p1, p2);
    Model* logistical = new Model("Logistical", 0, 100);

    //Add os systems e flows ao modelo
    logistical->add(p1);
    logistical->add(p2);
    logistical->add(log);

    //Roda o modelo
    logistical->run();

    assert(fabs(round(p1->getValue() * 10000) - 10000 * 88.2167) < 0.0001);
    assert(fabs(round(p2->getValue() * 10000) - 10000 * 21.7833) < 0.0001);

    delete(logistical);
    delete(log);
    delete(p1);
    delete(p2);
}
```

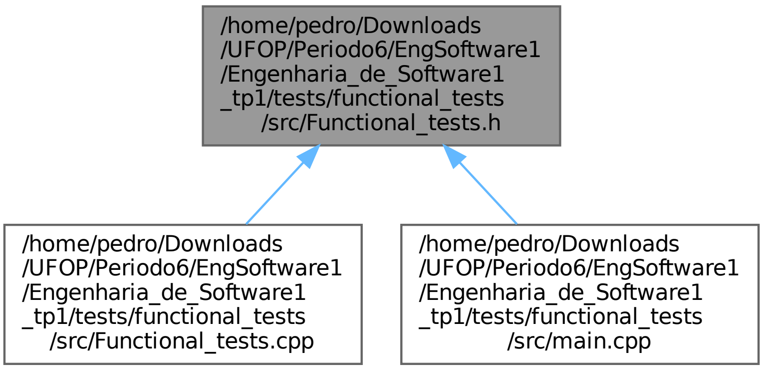
5.14 /home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/↩ Engenharia_de_Software1_tp1/tests/functional_tests/src/↩ Functional_tests.h File Reference

```
#include "../../../src/Model.h"
#include "../../../src/System.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:


/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Functional_tests.h
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Functional_tests.cpp
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/main.cpp

Functions

- void exponencial_test_run ()
- void logistical_test_run ()
- void Complex_test_run ()

5.14.1 Function Documentation

5.14.1.1 Complex_test_run()

```
void Complex_test_run ( )
                          {
    std::cout « "Complex funcional test" « std::endl;

    Model* model = new Model("Model", 0, 100);
    System* q1 = new System("q1", 100.0);
    System* q2 = new System("q2", 0.0);
    System* q3 = new System("q3", 100.0);
    System* q4 = new System("q4", 0.0);
    System* q5 = new System("q5", 0.0);
    Exponencial* f = new Exponencial("f", q1, q2);
    Exponencial* t = new Exponencial("t", q2, q3);
    Exponencial* u = new Exponencial("u", q3, q4);
    Exponencial* v = new Exponencial("v", q4, q1);
    Exponencial* g = new Exponencial("g", q1, q3);
    Exponencial* r = new Exponencial("r", q2, q5);

    model->add(q1);
    model->add(q2);
    model->add(q3);
    model->add(q4);
    model->add(q5);
    model->add(f);
    model->add(t);
    model->add(u);
    model->add(v);
    model->add(g);
    model->add(r);

    model->run();

    assert(fabs((round((q1->getValue() * 10000)) - 10000 * 31.8513)) < 0.0001);
    assert(fabs((round((q2->getValue() * 10000)) - 10000 * 18.4003)) < 0.0001);
    assert(fabs((round((q3->getValue() * 10000)) - 10000 * 77.1143)) < 0.0001);
    assert(fabs((round((q4->getValue() * 10000)) - 10000 * 56.1728)) < 0.0001);
    assert(fabs((round((q5->getValue() * 10000)) - 10000 * 16.4612)) < 0.0001);

    delete model;
    delete q1;
    delete q2;
    delete q3;
    delete q4;
    delete q5;
    delete f;
    delete t;
    delete u;
    delete v;
    delete g;
    delete r;

    std::cout « "Passed Complex funcional test" « std::endl;
}
```

5.14.1.2 exponencial_test_run()

void exponencial_test_run ( )

```
                            {
    std::cout « "Exponencial funcional test" « std::endl;

    System* pop1 = new System("pop1", 100.0);
    System* pop2 = new System("pop2", 0.0);
    Exponencial* exp = new Exponencial("exp", pop1, pop2);
    Model* exponencial = new Model("Exponencial", 0, 100);

    //Add os systems e flows ao modelo
    exponencial->add(pop1);
    exponencial->add(pop2);
    exponencial->add(exp);

    //Roda o modelo
    exponencial->run();

    assert(fabs((round(pop1->getValue() * 10000) - 10000 * 36.6032)) < 0.0001);
    assert(fabs((round(pop2->getValue() * 10000) - 10000 * 63.3968)) < 0.0001);

    delete(exponencial);
    delete(exp);
    delete(pop1);
    delete(pop2);

    std::cout « "Passed exponencial funcional test" « std::endl;
}
```

5.14.1.3 logistical_test_run()

void logistical_test_run ( )

```
                           {
    std::cout « "Logistical funcional test" « std::endl;

    System* p1 = new System("p1", 100.0);
    System* p2 = new System("p2", 10.0);
    Logistical* log = new Logistical("log", p1, p2);
    Model* logistical = new Model("Logistical", 0, 100);

    //Add os systems e flows ao modelo
    logistical->add(p1);
    logistical->add(p2);
    logistical->add(log);

    //Roda o modelo
    logistical->run();

    assert(fabs(round(p1->getValue() * 10000) - 10000 * 88.2167) < 0.0001);
    assert(fabs(round(p2->getValue() * 10000) - 10000 * 21.7833) < 0.0001);

    delete(logistical);
    delete(log);
    delete(p1);
    delete(p2);
}
```

5.15 Functional_tests.h

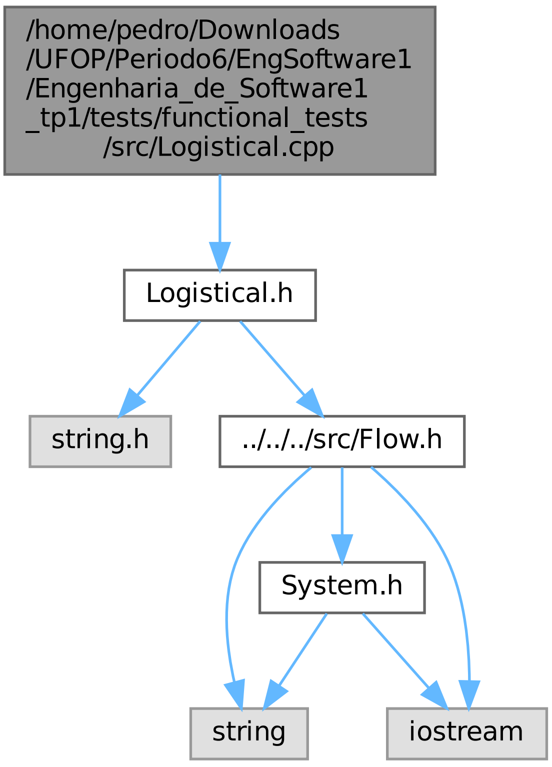
Go to the documentation of this file.

```
#ifndef FUNCTIONAL_TESTS_H
#define FUNCTIONAL_TESTS_H

#include "../../../src/Model.h"
#include "../../../src/System.h"
#include "Exponencial.h"
#include "Logistical.h"
#include <assert.h>
#include <cmath>
#include <iostream>
#include <cstdlib>

void exponencial_test_run();
void logistical_test_run();
void Complex_test_run();

#endif
```

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

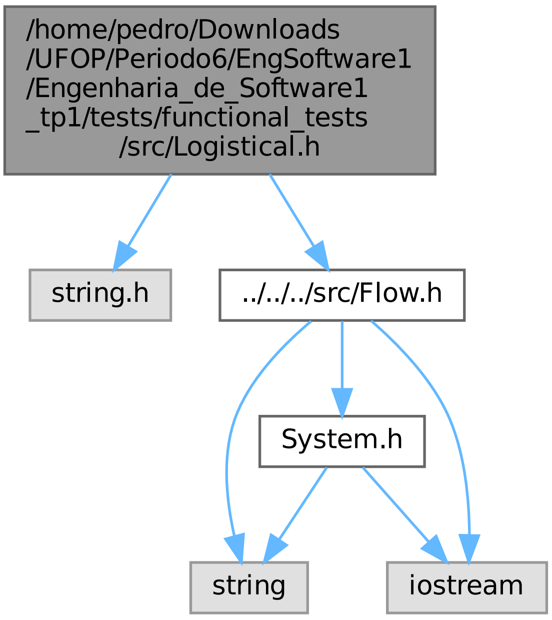
```
#include "Logistical.h"
```

Include dependency graph for Logistical.cpp:

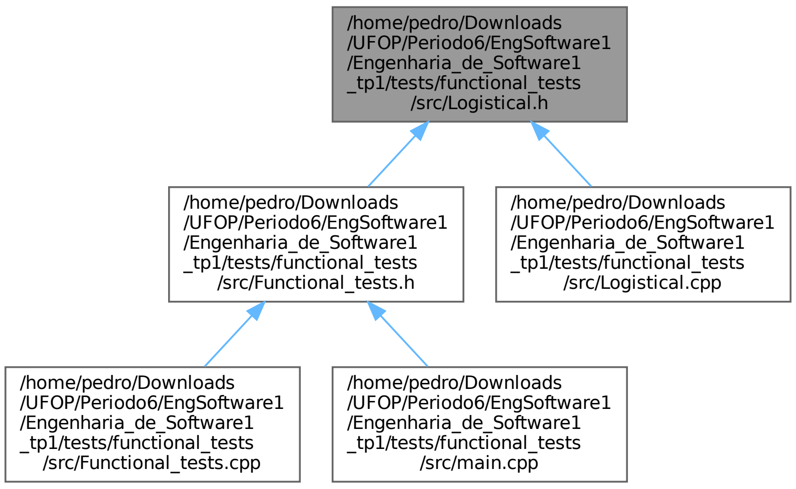

/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Logistical.cpp
Logistical.h
string.h
../../../src/Flow.h
System.h
string
iostream

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

```
#include <string.h>
#include "../../../src/Flow.h"
```
Include dependency graph for Logistical.h:



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


/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/Functional_tests.h
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Logistical.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/Functional_tests.cpp
/home/pedro/Downloads /UFOP/Periodo6/EngSoftware1 /Engenharia_de_Software1 _tp1/tests/functional_tests /src/main.cpp

Classes

- class Logistical

5.18 Logistical.h

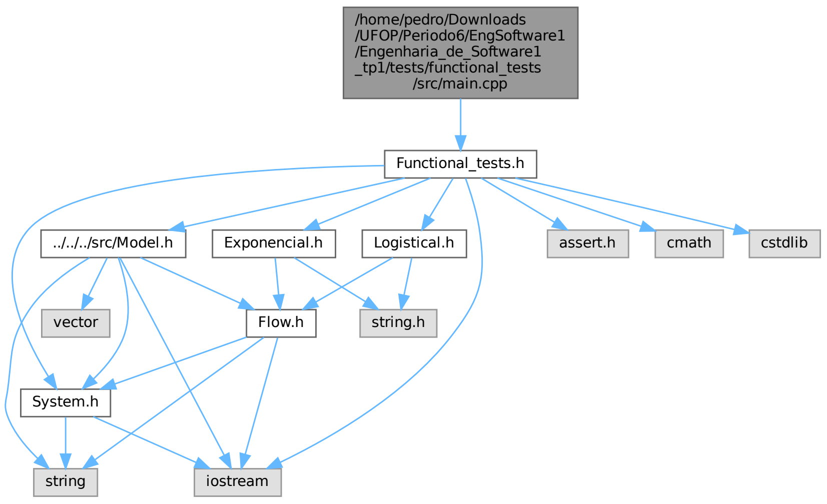
Go to the documentation of this file.

```
#ifndef LOGISTICAL_DEF
#define LOGISTICAL_DEF

#include <string.h>
#include "../../../src/Flow.h"

class Logistical : public Flow{
    public:
        //Contructor
        Logistical(const std::string& name = "NO_NAME", System* source = NULL, System* target = NULL);
        Logistical(const Logistical& other);

        //Destructor
        virtual ~Logistical();

        //Metodos
        virtual double execute() override;
};

#endif
```

5.19 /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 ()

5.19.1 Function Documentation

5.19.1.1 main()

```
int main ( )
        {
    exponencial_test_run();
    logistical_test_run();
    Complex_test_run();
    return 0;
}
```

Index

/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.cpp, 23
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Flow.h, 24, 25
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.cpp, 26
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/Model.h, 27, 29
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/System.cpp, 30
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/src/System.h, 30, 31
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Exponencial.cpp, 32
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Exponencial.h, 33, 34
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Functional_tests.cp
35
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Functional_tests.h, 37, 40
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Logistical.cpp, 40
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/Logistical.h, 41, 42
/home/pedro/Downloads/UFOP/Periodo6/EngSoftware1/Engenharia_de_Software1_tp1/tests/functional_tests/src/main.cpp, 43

∼Exponencial
- Exponencial, 9

∼Logistical
- Logistical, 14

∼Model
- Model, 16

∼System
- System, 20

add
- Model, 16

Complex_test_run
- Functional_tests.cpp, 35
- Functional_tests.h, 38

endTime
- Model, 18

execute
- Exponencial, 9
- Flow, 11
- Logistical, 15

Exponencial, 7
- ∼Exponencial, 9
- execute, 9
- Exponencial, 9

exponencial_test_run
- Functional_tests.cpp, 36
- Functional_tests.h, 39

Flow, 10
- execute, 11
- getName, 11
- getSource, 11
- getTarget, 11
- name, 12
- operator<<, 12
- operator=, 11
- operator=, 11
- setName, 11
- setSource, 11
- setTarget, 12
- source, 12
- target, 12

Flow.cpp
- operator<<, 24

flowIterator
- Model.h, 28

flows
- Model, 18

Functional_tests.cpp
- Complex_test_run, 35
- exponencial_test_run, 36
- logistical_test_run, 36

Functional_tests.h
- Complex_test_run, 38
- exponencial_test_run, 39
- logistical_test_run, 39

getEndtTime
- Model, 16

getName
- Flow, 11
- Model, 16
- System, 20

getSource
- Flow, 11

getStartTime
- Model, 16

getTarget
- Flow, 11

getValue
- System, 20

Logistical, 13
 ∼Logistical, 14
 execute, 15
 Logistical, 14
logistical_test_run
 Functional_tests.cpp, 36
 Functional_tests.h, 39

main
 main.cpp, 43
main.cpp
 main, 43
Model, 15
 ∼Model, 16
 add, 16
 endTime, 18
 flows, 18
 getEndtTime, 16
 getName, 16
 getStartTime, 16
 Model, 16
 name, 18
 operator<<, 18
 operator==, 16
 rmv, 17
 run, 17
 setEndTime, 17
 setName, 17
 setStartTime, 18
 setTime, 18
 startTime, 18
 systems, 19
Model.cpp
 operator<<, 26
Model.h
 flowIterator, 28
 systemIterator, 28

name
 Flow, 12
 Model, 18
 System, 21

operator<<
 Flow, 12
 Flow.cpp, 24
 Model, 18
 Model.cpp, 26
 System, 21
 System.cpp, 30
operator=
 Flow, 11
 System, 20
operator==
 Flow, 11
 Model, 16
 System, 20

rmv
 Model, 17
run
 Model, 17

setEndTime
 Model, 17
setName
 Flow, 11
 Model, 17
 System, 20
setSource
 Flow, 11
setStartTime
 Model, 18
setTarget
 Flow, 12
setTime
 Model, 18
setValue
 System, 20
source
 Flow, 12
startTime
 Model, 18
System, 19
 ∼System, 20
 getName, 20
 getValue, 20
 name, 21
 operator<<, 21
 operator=, 20
 operator==, 20
 setName, 20
 setValue, 20
 System, 19
 value, 21
System.cpp
 operator<<, 30
systemIterator
 Model.h, 28
systems
 Model, 19

target
 Flow, 12

value
 System, 21