MyVensin

Generated by Doxygen 1.9.3

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 ComplexFlowF Class Reference

`#include <funcional_tests.h>`

Inheritance diagram for ComplexFlowF:

4.2 ComplexFlowG Class Reference

`#include <funcional_tests.h>`

Inheritance diagram for ComplexFlowG:

Collaboration diagram for ComplexFlowG:

Public Member Functions

- ComplexFlowG (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.2.1 Constructor & Destructor Documentation

4.2.1.1 ComplexFlowG()

```
ComplexFlowG::ComplexFlowG (
            string name,
            System * source,
            System * target )  [inline]
```

4.2.2 Member Function Documentation

4.2.2.1 execute()

```
double ComplexFlowG::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.3 ComplexFlowR Class Reference

```
#include <funcional_tests.h>
```

Inheritance diagram for ComplexFlowR:

Collaboration diagram for ComplexFlowR:

Public Member Functions

- ComplexFlowR (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 ComplexFlowR()

```
ComplexFlowR::ComplexFlowR (
            string name,
            System * source,
            System * target )  [inline]
```

4.3.2 Member Function Documentation

4.3.2.1 execute()

```
double ComplexFlowR::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.4 ComplexFlowT Class Reference

```
#include <funcional_tests.h>
```

Inheritance diagram for ComplexFlowT:

Collaboration diagram for ComplexFlowT:

Public Member Functions

- ComplexFlowT (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.4.1 Constructor & Destructor Documentation

4.4.1.1 ComplexFlowT()

```
ComplexFlowT::ComplexFlowT (
            string name,
            System * source,
            System * target )  [inline]
```

4.4.2 Member Function Documentation

4.4.2.1 execute()

```
double ComplexFlowT::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.5 ComplexFlowU Class Reference

```
#include <funcional_tests.h>
```

Inheritance diagram for ComplexFlowU:

Collaboration diagram for ComplexFlowU:

Public Member Functions

- ComplexFlowU (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.5.1 Constructor & Destructor Documentation

4.5.1.1 ComplexFlowU()

```
ComplexFlowU::ComplexFlowU (
            string name,
            System * source,
            System * target )  [inline]
```

4.5.2 Member Function Documentation

4.5.2.1 execute()

```
double ComplexFlowU::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.6 ComplexFlowV Class Reference

```
#include <funcional_tests.h>
```

Inheritance diagram for ComplexFlowV:

Collaboration diagram for ComplexFlowV:

Public Member Functions

- ComplexFlowV (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.6.1 Constructor & Destructor Documentation

4.6.1.1 ComplexFlowV()

```
ComplexFlowV::ComplexFlowV (
            string name,
            System * source,
            System * target )  [inline]
```

4.6.2 Member Function Documentation

4.6.2.1 execute()

```
double ComplexFlowV::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.7 ExponencialFlow Class Reference

```
#include <funcional_tests.h>
```

Inheritance diagram for ExponencialFlow:

Collaboration diagram for ExponencialFlow:

Public Member Functions

- ExponencialFlow (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 ExponencialFlow()

```
ExponencialFlow::ExponencialFlow (
            string name,
            System * source,
            System * target )  [inline]
```

4.7.2 Member Function Documentation

4.7.2.1 execute()

```
double ExponencialFlow::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.8 Flow Class Reference

```
#include <flow.h>
```

Inheritance diagram for Flow:

Collaboration diagram for Flow:

Public Member Functions

- Flow (string name="", System *source=NULL, System *target=NULL)
- virtual ~Flow ()
- virtual double execute ()=0
- string getName () const
- System * getSource () const
- System * getTarget () const
- void setName (string flowName)
- void setSource (System *sourceSys)
- void setTarget (System *targetSys)
- void clearSource ()
- void clearTarget ()

Protected Member Functions

- Flow (const Flow &flow)
- Flow & operator= (const Flow &flow)

Protected Attributes

- string name
- System * source
- System * target

Friends

- class Model
- class UnitFlow

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Flow() [1/2]

```
Flow::Flow (
            const Flow & flow )  [protected]
```

Here is the call graph for this function:

4.8.1.2 Flow() [2/2]

```
Flow::Flow (
            string name = "",
            System * source = NULL,
            System * target = NULL )
```

4.8.1.3 ~Flow()

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

4.8.2 Member Function Documentation

4.8.2.1 clearSource()

```
void Flow::clearSource ( )
```

4.8.2.2 clearTarget()

```
void Flow::clearTarget ( )
```

4.8.2.3 execute()

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

Implemented in ExponencialFlow, LogisticFlow, ComplexFlowF, ComplexFlowT, ComplexFlowU, ComplexFlowV, ComplexFlowG, ComplexFlowR, and TestFlow.

4.8.2.4 getName()

```
string Flow::getName ( ) const
```

Here is the caller graph for this function:

4.8.2.5 getSource()

```
System * Flow::getSource ( ) const
```

Here is the caller graph for this function:

4.8.2.6 getTarget()

```
System * Flow::getTarget ( ) const
```

Here is the caller graph for this function:

4.8.2.7 operator=()

```
Flow & Flow::operator= (
            const Flow & flow )  [protected]
```

Here is the call graph for this function:

4.8.2.8 setName()

```
void Flow::setName (
            string flowName )
```

4.8.2.9 setSource()

```
void Flow::setSource (
            System * sourceSys )
```

4.8.2.10 setTarget()

```
void Flow::setTarget (
            System * targetSys )
```

4.8.3 Friends And Related Function Documentation

4.8.3.1 Model

```
friend class Model  [friend]
```

4.8.3.2 UnitFlow

```
friend class UnitFlow  [friend]
```

4.8.4 Member Data Documentation

4.8.4.1 name

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

4.8.4.2 source

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

4.8.4.3 target

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

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

- src/flow.h
- src/flow.cpp

4.9 LogisticFlow Class Reference

```
#include <funcional_tests.h>
```

Inheritance diagram for LogisticFlow:

Collaboration diagram for LogisticFlow:

Public Member Functions

- LogisticFlow (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 LogisticFlow()

```
LogisticFlow::LogisticFlow (
            string name,
            System * source,
            System * target )  [inline]
```

4.9.2 Member Function Documentation

4.9.2.1 execute()

```
double LogisticFlow::execute ( )  [inline], [virtual]
```

Implements Flow.

Here is the call graph for this function:

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

- test/funcional/funcional_tests.h

4.10 Model Class Reference

```
#include <model.h>
```

Public Types

- typedef vector< System * >::iterator systemIterator
- typedef vector< Flow * >::iterator flowIterator

Public Member Functions

- systemIterator beginSystems ()
- systemIterator endSystems ()
- flowIterator beginFlows ()
- flowIterator endFlows ()
- Model (string name="", double time=0)
- virtual ~Model ()
- void execute (double start=0, double final=0, double increment=1)
- void add (System *sys)
- void remove (System *sys)
- void add (Flow *flow)
- void remove (Flow *flow)
- void setName (string modelName)
- void setTime (double currentTime)
- string getName () const
- double getTime () const
- System * getSystem (int index)
- Flow * getFlow (int index)
- void incrementTime (double increment)

Protected Attributes

- string name
- double time
- vector< System * > systems
- vector< Flow * > flows

4.10.1 Member Typedef Documentation

4.10.1.1 flowIterator

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

4.10.1.2 systemIterator

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

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Model()

```
Model::Model (
            string name = "",
            double time = 0 )
```

4.10.2.2 ~Model()

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

4.10.3 Member Function Documentation

4.10.3.1 add() [1/2]

```
void Model::add (
            Flow * flow )
```

Here is the call graph for this function:

4.10.3.2 add() [2/2]

```
void Model::add (
            System * sys )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.10.3.3 beginFlows()

```
Model::flowIterator Model::beginFlows ( )
```

Here is the caller graph for this function:

4.10.3.4 beginSystems()

```
Model::systemIterator Model::beginSystems ( )
```

Here is the caller graph for this function:

4.10.3.5 endFlows()

```
Model::flowIterator Model::endFlows ( )
```

Here is the caller graph for this function:

4.10.3.6 endSystems()

```
Model::systemIterator Model::endSystems ( )
```

Here is the caller graph for this function:

4.10.3.7 execute()

```
void Model::execute (
            double start = 0,
            double final = 0,
            double increment = 1 )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.10.3.8 getFlow()

```
Flow * Model::getFlow (
            int index )
```

Here is the caller graph for this function:

4.10.3.9 getName()

```
string Model::getName ( ) const
```

Here is the caller graph for this function:

4.10.3.10 getSystem()

```
System * Model::getSystem (
            int index )
```

Here is the caller graph for this function:

4.10.3.11 getTime()

```
double Model::getTime ( ) const
```

Here is the caller graph for this function:

4.10.3.12 incrementTime()

```
void Model::incrementTime (
            double increment )
```

4.10.3.13 remove() [1/2]

```
void Model::remove (
            Flow * flow )
```

Here is the call graph for this function:

4.10.3.14 remove() [2/2]

```
void Model::remove (
            System * sys )
```

Here is the call graph for this function:

4.10.3.15 setName()

```
void Model::setName (
            string modelName )
```

4.10.3.16 setTime()

```
void Model::setTime (
            double currentTime )
```

4.10.4 Member Data Documentation

4.10.4.1 flows

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

4.10.4.2 name

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

4.10.4.3 systems

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

4.10.4.4 time

```
double Model::time  [protected]
```

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

- src/model.h
- src/model.cpp

4.11 System Class Reference

```
#include <system.h>
```

Public Member Functions

- System (const System &sys)
- System (string name="", double value=0)
- virtual ~System ()
- void setName (string sysName)
- void setValue (double sysValue)
- string getName () const
- double getValue () const
- double operator+ (const System &sys)
- double operator+ (const double &valueSys)
- double operator- (const System &sys)
- double operator- (const double &valueSys)
- double operator* (const System &sys)
- double operator* (const double &valueSys)
- double operator/ (const System &sys)
- double operator/ (const double &valueSys)

Protected Member Functions

- System & operator= (const System &sys)

Protected Attributes

- string name
- double value

Friends

- class Flow
- class Model
- class UnitSystem

4.11.1 Constructor & Destructor Documentation

4.11.1.1 System() [1/2]

```
System::System (
            const System & sys )
```

Here is the call graph for this function:

4.11.1.2 System() [2/2]

```
System::System (
            string name = "",
            double value = 0 )
```

4.11.1.3 ∼System()

```
System::∼System ( )  [virtual]
```

4.11.2 Member Function Documentation

4.11.2.1 getName()

```
string System::getName ( ) const
```

Here is the caller graph for this function:

4.11.2.2 getValue()

```
double System::getValue ( ) const
```

Here is the caller graph for this function:

4.11.2.3 operator∗() [1/2]

```
double System::operator* (
            const double & valueSys )
```

4.11.2.4 operator*() [2/2]

```
double System::operator* (
            const System & sys )
```

Here is the call graph for this function:

4.11.2.5 operator+() [1/2]

```
double System::operator+ (
            const double & valueSys )
```

4.11.2.6 operator+() [2/2]

```
double System::operator+ (
            const System & sys )
```

Here is the call graph for this function:

4.11.2.7 operator-() [1/2]

```
double System::operator- (
            const double & valueSys )
```

4.11.2.8 operator-() [2/2]

```
double System::operator- (
            const System & sys )
```

Here is the call graph for this function:

4.11.2.9 operator/() [1/2]

```
double System::operator/ (
            const double & valueSys )
```

4.11.2.10 operator/() [2/2]

```
double System::operator/ (
            const System & sys )
```

Here is the call graph for this function:

4.11.2.11 operator=()

```
System & System::operator= (
            const System & sys )  [protected]
```

Here is the call graph for this function:

4.11.2.12 setName()

```
void System::setName (
            string sysName )
```

Here is the caller graph for this function:

4.11.2.13 setValue()

```
void System::setValue (
            double sysValue )
```

Here is the caller graph for this function:

4.11.3 Friends And Related Function Documentation

4.11.3.1 Flow

```
friend class Flow  [friend]
```

4.11.3.2 Model

```
friend class Model  [friend]
```

4.11.3.3 UnitSystem

```
friend class UnitSystem  [friend]
```

4.11.4 Member Data Documentation

4.11.4.1 name

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

4.11.4.2 value

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

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

- src/system.h
- src/system.cpp

4.12 TestFlow Class Reference

```
#include <unit_tests.h>
```

Inheritance diagram for TestFlow:

Collaboration diagram for TestFlow:

Public Member Functions

- TestFlow (string name, System *source, System *target)
- double execute ()

Additional Inherited Members

4.12.1 Constructor & Destructor Documentation

4.12.1.1 TestFlow()

```
TestFlow::TestFlow (
            string name,
            System * source,
            System * target )  [inline]
```

4.12.2 Member Function Documentation

4.12.2.1 execute()

```
double TestFlow::execute ( )  [inline], [virtual]
```

Implements Flow.

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

- test/unit/unit_tests.h

Chapter 5

File Documentation

5.1 src/flow.cpp File Reference

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

5.2 src/flow.h File Reference

```
#include "system.h"
#include <string>
```
Include dependency graph for flow.h: This graph shows which files directly or indirectly include this file:

Classes

- class Flow

5.3 flow.h

Go to the documentation of this file.

```
1 #ifndef FLOW_H
2 #define FLOW_H
3
4 #include "system.h"
5
6 #include <string>
7
8 class Flow {
9     protected:
10         string name;
11         System *source;
12         System *target;
13
14         Flow (const Flow &flow);
15
16         Flow& operator= (const Flow &flow);
17
18     public:;
19         friend class Model;
20         friend class UnitFlow;
21
22         Flow (string name = "", System *source = NULL, System *target = NULL);
```

```
23
24          virtual ~Flow ();
25
26          virtual double execute () = 0;
27
28          string getName () const;
29          System* getSource () const;
30          System* getTarget () const;
31
32          void setName (string flowName);
33          void setSource (System *sourceSys);
34          void setTarget (System *targetSys);
35
36          void clearSource ();
37          void clearTarget ();
38
39 };
40
41
42 #endif
```

5.4 src/model.cpp File Reference

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

5.5 src/model.h File Reference

```
#include "flow.h"
#include <vector>
```
Include dependency graph for model.h: This graph shows which files directly or indirectly include this file:

Classes

- class Model

5.6 model.h

Go to the documentation of this file.

```
1 #ifndef MODEL_H
2 #define MODEL_H
3
4 #include "flow.h"
5
6 #include <vector>
7
8 class Model {
9     protected:
10          string name;
11          double time;
12
13          vector <System*> systems;
14          vector <Flow*> flows;
15
16      private:
17          Model (const Model &model);
18
19          void operator= (const Model &model);
20
21      public:
22          typedef vector<System*> :: iterator systemIterator;
23          typedef vector<Flow*> :: iterator flowIterator;
24
25          systemIterator beginSystems();
```

```
26          systemIterator endSystems();
27          flowIterator beginFlows();
28          flowIterator endFlows();
29
30          Model (string name = "", double time = 0);
31
32          virtual ~Model();
33
34
35          void execute (double start = 0, double final = 0, double increment = 1);
36
37          void add (System *sys);
38          void remove (System *sys);
39
40          void add (Flow *flow);
41          void remove (Flow *flow);
42
43
44          void setName(string modelName);
45          void setTime(double currentTime);
46
47          string getName() const;
48          double getTime() const;
49
50          System* getSystem(int index);
51          Flow* getFlow(int index);
52
53          void incrementTime(double increment);
54
55
56 };
57
58 #endif
```

5.7 src/system.cpp File Reference

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

Functions

- double operator+ (const double &valueSys, const System &sys)
- double operator- (const double &valueSys, const System &sys)
- double operator∗ (const double &valueSys, const System &sys)
- double operator/ (const double &valueSys, const System &sys)

5.7.1 Function Documentation

5.7.1.1 operator∗()

```
double operator* (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.7.1.2 operator+()

```
double operator+ (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.7.1.3 operator-()

```
double operator- (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.7.1.4 operator/()

```
double operator/ (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.8 src/system.h File Reference

```
#include <iostream>
#include <string>
#include <ios>
#include <fstream>
```

Include dependency graph for system.h: This graph shows which files directly or indirectly include this file:

Classes

- class System

Functions

- double operator+ (const double &valueSys, const System &sys)
- double operator- (const double &valueSys, const System &sys)
- double operator∗ (const double &valueSys, const System &sys)
- double operator/ (const double &valueSys, const System &sys)

5.8.1 Function Documentation

5.8.1.1 operator∗()

```
double operator* (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.8.1.2 operator+()

```
double operator+ (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.8.1.3 operator-()

```
double operator- (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.8.1.4 operator/()

```
double operator/ (
            const double & valueSys,
            const System & sys )
```

Here is the call graph for this function:

5.9 system.h

Go to the documentation of this file.

```
1 #ifndef SYSTEM_H
2 #define SYSTEM_H
3
4 #include <iostream>
5 #include <string>
6 #include <ios>
7 #include <fstream>
8
9 using namespace std;
10
11 class System {
12     protected:
13         string name;
14         double value;
15
16         System& operator=(const System& sys);
17
18     public:
19
20         friend class Flow;
21         friend class Model;
22         friend class UnitSystem;
23
```

```
24          // Construtor e Destrutor
25          System (const System& sys);
26
27          System (string name = "", double value = 0);
28
29          virtual ~System();
30
31          // Setters
32
33          void setName(string sysName);
34
35          void setValue(double sysValue);
36
37          // Getters
38
39          string getName() const;
40
41          double getValue() const;
42
43          // Sobrecarga de Operadores
44          // Operador +
45          double operator+(const System& sys);
46
47          double operator+(const double& valueSys);
48
49          //Operador -
50          double operator-(const System& sys);
51
52          double operator-(const double& valueSys);
53
54          // Operador *
55          double operator*(const System& sys);
56
57          double operator*(const double& valueSys);
58
59          // Operador /
60          double operator/(const System& sys);
61
62          double operator/(const double& valueSys);
63
64
65
66 };
67
68 double operator+(const double& valueSys, const System& sys);
69 double operator-(const double& valueSys, const System& sys);
70 double operator*(const double& valueSys, const System& sys);
71 double operator/(const double& valueSys, const System& sys);
72
73 #endif
```

5.10 test/funcional/funcional_tests.cpp File Reference

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

Functions

- void exponentialFuncionalTest ()
- void logisticalFuncionalTest ()
- void complexFuncionalTest ()

5.10.1 Function Documentation

5.10.1.1 complexFuncionalTest()

```
void complexFuncionalTest ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.10.1.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.10.1.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.11 test/funcional/funcional_tests.h File Reference

```
#include "../../src/flow.h"
#include "../../src/system.h"
#include "../../src/model.h"
#include <assert.h>
```

Include dependency graph for funcional_tests.h: This graph shows which files directly or indirectly include this file:

Classes

- class ExponencialFlow
- class LogisticFlow
- class ComplexFlowF
- class ComplexFlowT
- class ComplexFlowU
- class ComplexFlowV
- class ComplexFlowG
- class ComplexFlowR

Functions

- void exponentialFuncionalTest ()
- void logisticalFuncionalTest ()
- void complexFuncionalTest ()

5.11.1 Function Documentation

5.11.1.1 complexFuncionalTest()

```
void complexFuncionalTest ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.11.1.2 exponentialFuncionalTest()

```
void exponentialFuncionalTest ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.11.1.3 logisticalFuncionalTest()

```
void logisticalFuncionalTest ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.12 funcional_tests.h

Go to the documentation of this file.

```
1 #ifndef FUNCTIONAL_TEST
2 #define FUNCTIONAL_TEST
3
4 #include "../../src/flow.h"
5 #include "../../src/system.h"
6 #include "../../src/model.h"
7
8 #include <assert.h>
9
10 class ExponencialFlow : public Flow {
11     public:
12         ExponencialFlow (string name, System *source, System *target):
13             Flow (name, source, target) {}
14
15         double execute () {
16             if (getSource() != NULL){
17                 return (0.01 * getSource()->getValue());
18             }
19             else
20                 return 0;
21         }
22
23 };
24
25 class LogisticFlow : public Flow {
26     public:
27         LogisticFlow (string name, System *source, System *target):
28             Flow (name, source, target) {}
29
30         double execute(){
31             if (getTarget() != NULL)
32                 return 0.01 * getTarget()->getValue() * (1 - getTarget()->getValue() / 70);
33             else
34                 return 0;
35         }
36 };
37
38 class ComplexFlowF : public Flow{
39     public:
40         ComplexFlowF(string name, System *source, System *target):
41             Flow (name, source, target) {}
42
43         double execute() {
44             if (getSource() != NULL)
45                 return 0.01 * getSource()->getValue();
46             else
```

```
47                 return 0;
48         }
49 };
50
51
52 class ComplexFlowT : public Flow{
53     public:
54         ComplexFlowT(string name, System *source, System *target):
55             Flow (name, source, target) {}
56
57         double execute() {
58             if (getSource() != NULL)
59                 return 0.01 * getSource()->getValue();
60             else
61                 return 0;
62         }
63 };
64
65
66 class ComplexFlowU : public Flow {
67     public:
68         ComplexFlowU(string name, System *source, System *target):
69             Flow (name, source, target) {}
70
71         double execute(){
72             if (getSource() != NULL)
73                 return 0.01 * getSource()->getValue();
74             else
75                 return 0;
76         }
77 };
78
79
80 class ComplexFlowV : public Flow {
81     public:
82         ComplexFlowV(string name, System *source, System *target):
83             Flow (name, source, target) {}
84
85         double execute(){
86             if (getSource() != NULL)
87                 return 0.01 * getSource()->getValue();
88             else
89                 return 0;
90         }
91 };
92
93
94
95 class ComplexFlowG : public Flow {
96     public:
97         ComplexFlowG(string name, System *source, System *target):
98             Flow (name, source, target) {}
99
100         double execute(){
101             if (getSource() != NULL)
102                 return 0.01 * getSource()->getValue();
103             else
104                 return 0;
105         }
106 };
107
108
109
110 class ComplexFlowR: public Flow {
111     public:
112         ComplexFlowR(string name, System *source, System *target):
113             Flow (name, source, target) {}
114
115         double execute(){
116             if (getSource() != NULL)
117                 return 0.01 * getSource()->getValue();
118             else
119                 return 0;
120         }
121 };
122
123
124 void exponentialFuncionalTest ();
125 void logisticalFuncionalTest ();
126 void complexFuncionalTest ();
127
128 #endif
```

5.13 test/funcional/main.cpp File Reference

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

Functions

- int main ()

5.13.1 Function Documentation

5.13.1.1 main()

```
int main ( )
```

Here is the call graph for this function:

5.14 test/unit/main.cpp File Reference

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

Functions

- int main ()

5.14.1 Function Documentation

5.14.1.1 main()

```
int main ( )
```

Here is the call graph for this function:

5.15 test/unit/unit_tests.cpp File Reference

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

Functions

- void testFlow ()
- void testModel ()
- void testSystem ()

5.15.1 Function Documentation

5.15.1.1 testFlow()

```
void testFlow ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.15.1.2 testModel()

```
void testModel ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.15.1.3 testSystem()

```
void testSystem ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.16 test/unit/unit_tests.h File Reference

```
#include "../../src/flow.h"
#include "../../src/system.h"
#include "../../src/model.h"
#include <assert.h>
#include <stdlib.h>
```

Include dependency graph for unit_tests.h: This graph shows which files directly or indirectly include this file:

Classes

- class TestFlow

Functions

- void testFlow ()
- void testModel ()
- void testSystem ()

5.16.1 Function Documentation

5.16.1.1 testFlow()

```
void testFlow ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.16.1.2 testModel()

```
void testModel ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.16.1.3 testSystem()

```
void testSystem ( )
```

Here is the call graph for this function: Here is the caller graph for this function:

5.17 unit_tests.h

Go to the documentation of this file.

```
1 #ifndef UNIT_TEST
2 #define UNIT_TEST
3
4 #include "../../src/flow.h"
5 #include "../../src/system.h"
6 #include "../../src/model.h"
7
8 #include <assert.h>
9 #include <stdlib.h>
10
11 class TestFlow : public Flow {
12     public:
13         TestFlow (string name, System *source, System *target):
14             Flow (name, source, target) {}
15
16         double execute () {
17             cout « "Testing a Flow" « endl;
18             return 0;
19         }
20
21
22 };
23
24 void testFlow();
25 void testModel();
26 void testSystem();
27
28
29
30 #endif
```

Index