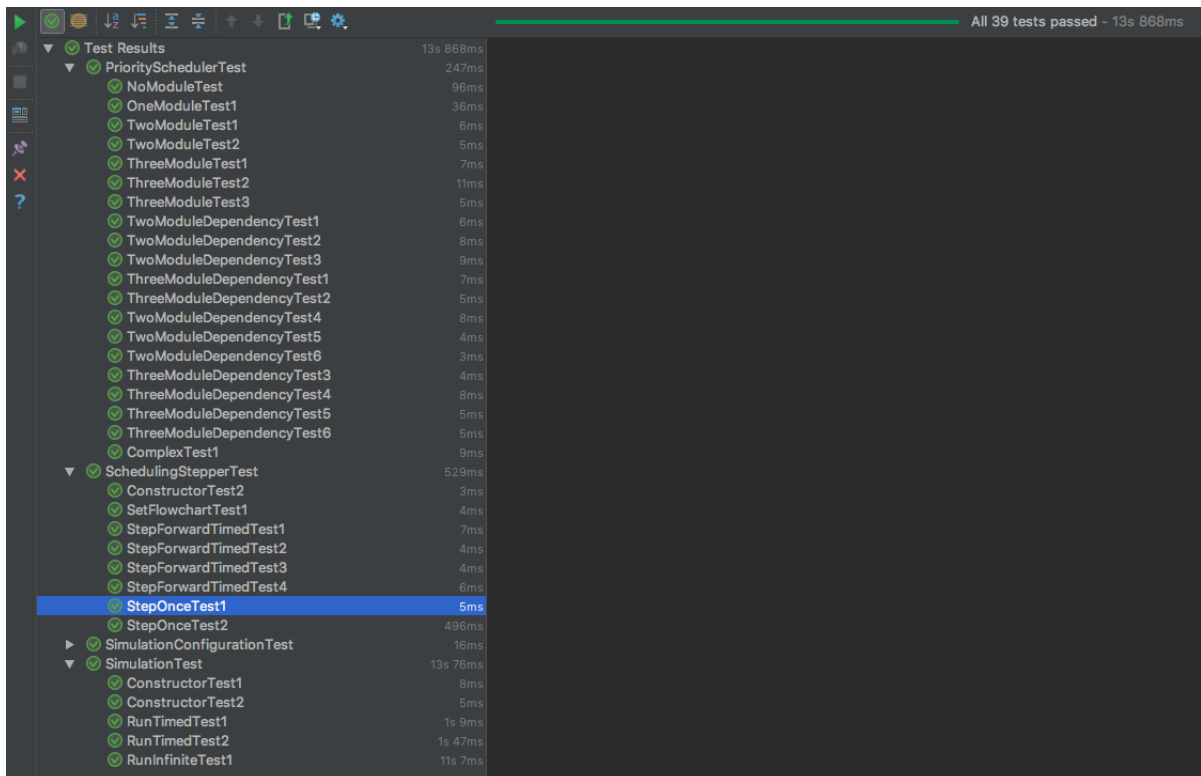


Testing Flowchart-Module for OCTANE

Tobias Bleymehl Tobias Bodmer Diana Burkart
Matthias Lüthy Sebastian Weber
Jonathan Schenkenberger



Test Results		13s 868ms
▼	PrioritySchedulerTest	247ms
✓	NoModuleTest	96ms
✓	OneModuleTest1	36ms
✓	TwoModuleTest1	6ms
✓	TwoModuleTest2	5ms
✓	ThreeModuleTest1	7ms
✓	ThreeModuleTest2	11ms
✓	ThreeModuleTest3	5ms
✓	TwoModuleDependencyTest1	6ms
✓	TwoModuleDependencyTest2	8ms
✓	TwoModuleDependencyTest3	9ms
✓	ThreeModuleDependencyTest1	7ms
✓	ThreeModuleDependencyTest2	5ms
✓	TwoModuleDependencyTest4	8ms
✓	TwoModuleDependencyTest5	4ms
✓	TwoModuleDependencyTest6	3ms
✓	ThreeModuleDependencyTest3	4ms
✓	ThreeModuleDependencyTest4	8ms
✓	ThreeModuleDependencyTest5	5ms
✓	ThreeModuleDependencyTest6	5ms
✓	ComplexTest1	9ms
▼	SchedulingStepperTest	529ms
✓	ConstructorTest2	3ms
✓	SetFlowchartTest1	4ms
✓	StepForwardTimedTest1	7ms
✓	StepForwardTimedTest2	4ms
✓	StepForwardTimedTest3	4ms
✓	StepForwardTimedTest4	6ms
✓	StepOnceTest1	5ms
✓	StepOnceTest2	496ms
▶	SimulationConfigurationTest	16ms
▼	SimulationTest	13s 76ms
✓	ConstructorTest1	8ms
✓	ConstructorTest2	5ms
✓	RunTimedTest1	1s 9ms
✓	RunTimedTest2	1s 47ms
✓	RunInfiniteTest1	11s 7ms

October 21, 2020

Contents

1	Introduction	2
2	Unit tests - code coverage	2
2.1	Model	2
2.1.1	FlowChart	2
2.1.2	Graphical Representation	2
2.1.3	Simulation	3
2.1.4	Store and Load	3
2.1.5	Data	3
2.2	View	3
2.3	Controller	3
3	Found issues	4
3.1	View	4
3.2	Model	7
3.3	Controller	8
4	Unresolved known issues	9
4.1	View	9
4.2	General	9
4.3	Modules	10
5	Local test cases from the requirements specification	11
5.1	Modules	11
5.1.1	Add a module to a flowchart (T10.1)	11
5.1.2	Add a module to a flowchart (T10.2)	11
5.1.3	Select a module in a flowchart (T11)	11
5.1.4	Deselect a module in a flowchart (T12)	12
5.1.5	Deselect one module and select another one (T13)	12
5.1.6	Delete a module (T14)	12
5.1.7	Shift a module (T15)	12
5.1.8	Turn a module (T16)	12
5.1.9	Scale a module (T17)	13
5.1.10	Change the number of FunctionModule inputs (T18)	13
5.1.11	Calculating with the FunctionModule (T19)	13
5.1.12	Connecting two module instances (T20)	13
5.1.13	Connecting two modules that cannot be connected (T21)	14
5.1.14	Modules stay connected under rotation (T22)	14
5.1.15	Show data types on the connections (T23)	14
5.1.16	Adding a rigid connection(T25)	14
5.1.17	Adding a StdArrow(T26)	14
5.1.18	Splitting an Arrow (T27)	15
5.1.19	Splitting an Arrow incorrectly (T28)	15
5.2	Simulation	15
5.2.1	Starting the Simulation (T30)	15
5.2.2	Running one simulation cycle (T31)	15

5.2.3	Pausing the simulation (T32)	15
5.2.4	Resetting the simulation (T33)	16
5.3	Saving and loading of a flowchart	16
5.3.1	Saving (T40.1)	16
5.3.2	Saving (T40.2)	16
5.3.3	Loading (T41.1)	17
5.3.4	Loading (T41.2)	17
5.3.5	Add modules (T42)	17
5.3.6	Reload module list (T43)	17
5.4	Others	18
5.4.1	Module specific settings (T50 changed)	18
5.4.2	Changing the name of a module (T51)	18
5.4.3	Hide module list sidebar (T52 changed)	18
6	Global test cases from the requirements specification	19
6.0.1	Create a flowchart (T100)	19
6.0.2	Error while creating a flowchart (T101)	19
6.0.3	Trying to open a corrupted flow file, thereafter opening a correct file and starting the simulation (T102 changed)	19
7	Known Windows/VS issues while developing	20
8	Conclusion	20

1 Introduction

This document depicts the test phase of the Flowchart-Module for the open source simulation environment OCTANE. It first shows the unit tests of all components of the project and all data related to that e.g. code coverage. Then we shortly present some issues we had and still have. In the following two chapters an overview of all test cases of the requirement specification is given. Finally we give a short conclusion of the test phase.

2 Unit tests - code coverage

The section provides a short overview over all unit tests and the code coverage of each part of the project. We explain which parts of the project were tested and how we tested them. Additionally we give approximations of how much of the actual code of each part is covered with the corresponding tests. Since these are only estimates please oversee minor deviation.

2.1 Model

The following subsections show each part of the Model and how it has been tested.

2.1.1 FlowChart

The FlowChart part of the project is tested by unit tests as well as by tests carried out by hand. The unit tests consist of a FlowChart test (also testing the smaller parts of the flowchart such as modules, pins and connections), a module test (testing more special pin properties of the module) and a graphical representation test in order to test the correct functioning of the graphical representation. The tests carried out by hand are explained later in this document. We can ensure an estimate code coverage around 55%.

2.1.2 Graphical Representation

The graphical representation is tested by unit tests for each flowchart element. The calculation of the connections has been improved and the arrows at the end of a standard arrow are now set correctly even if the line is moved.

2.1.3 Simulation

The Simulation part of the project is tested quite well. Since the simulation is a major part of the functionality of the project we did not spare any expenses to fully test the simulation including the Scheduler and Stepper. We can ensure an estimate code coverage around 60%.

2.1.4 Store and Load

Store and Load was implemented with the intention of using dynamic loading, but we did not integrate this. Therefore DynamicModuleLoader, ModuleParser and ModuleList contain methods and data intended to manage dynamic loading of classes. This code cannot be tested as the whole concept is not working yet. FlowchartParser, DOM and StoreLoad are tested quite well, so the overall coverage is about 60%. Tested features are saving and loading a flowchart and loading the module list (in linux the module list is loaded and filled statically, in windows the list is loaded dynamically but filled statically). Saving a flowchart is only working reliably on Windows. Loading is also possible on Linux.

2.1.5 Data

The data part of the project is tested quite well by 20 unit test cases. These tests test all data wrappers and data signatures. We can ensure an estimate code coverage around 60%.

2.2 View

The View is quite hard to test since many tests are user interactions. Therefore we did many of the tests by hand. Since there are infinite ways to e.g. move a module it is hard for us to measure the code coverage of this part. All we want to say here is that we tested every functionality of the View by hand and we can say that we also reach around 60% code coverage here if not more.

2.3 Controller

Since the Controllers job is only to pass commands from the View to the Model there was not much to test here. All commands are tested with mocks so that they take the data given to them and pass them on correctly. In terms of code coverage this should be around 90% minimum since there are no branches whatsoever and each command has been tested.

3 Found issues

This section of the document focuses on the issues we came across while testing the software. It describes which issues and bugs we found and how we fixed them.

3.1 View

Bug 01: The modules do not update their view-icon if the name is changed in the sidebar

symptom: If you change the name of a module in the module tabs-Sidebar, the icon was not updated. It had still the same name.

reason: There was an error in the Observer-pattern. The corresponding ModuleGR was not notified that it needs to update the icon. Besides that the externalConfig did not know anything about the module.

repair: The external configuration of each module now contains a reference to the module. If the config is changed, it automatically calls the setInfoText-method of the corresponding ModuleGR.

Bug 02: Arrows sometimes look strange if the module is moved horizontally

symptom: If a standard arrow was moved horizontally, some strange things happened to the arrows (wrong layout).

reason: There was a mistake with the use of the different points.

repair: The algorithm now uses the correct points for the calculation.

Bug 03: Activating and deactivating the sidebars was not possible

symptom: It was only possible to hide the ModuleListSidebar, but not to reactivate it. The ModuleTabsSidebar could not be hidden.

reason: ModuleListSidebar: feature was not implemented yet.

ModuleTabsSidebar: there was a mistake in the use of wxAui so that the sidebar was not hidden if no module is activated.

repair: ModuleListSidebar: implemented the menu items;
ModuleTabsSidebar now calls another method which definitely hides/ shows the sidebar

Bug 04: Raster on/off is not working

symptom: It was not possible to switch the raster on and off, it was always displayed.

reason: The feature was not implemented yet.

repair: Implemented the corresponding menu item.

Bug 05: Not accessible menu items are not deactivated

symptom: Different menu items which are not accessible (because the feature is not implemented or an action is not possible at the moment, e.g. start the simulation if the simulation is already started) were activated in the menu bar.

reason: During the implementation it was not definitely clear which features will work at the end. The simulation items were not respected.

repair: Deactivated the appropriate menu items. Added a condition for the simulation items to be activated or deactivated.

Bug 06: All pins are in front of the modules

symptom: All module pins were shown in front of all modules.

reason: The view first drew all modules, then all pins and at the end all connections.

repair: The module pins are now stored in each ModuleView and they are drawn together with their module so that never modules and pins are displayed in front of older ones.

Bug 07: Remove connections is not working

symptom: It was not possible to remove connections from the flowchart.

reason: There have been different double ownerships which led to a segmentation fault.

repair: Use of unique pointers for the connections

Bug 08: Selected pins and connections are not marked graphically

symptom: If a pin or a connection was selected, you were not able to see whether it was really selected or not.

reason: The feature was not implemented yet.

repair: Implemented the relevant methods.

Bug 09: Selected modules are not deactivated if a connection gets activated

symptom: If a module was activated and afterwards a connection was activated without clicking into the raster in between, the module was not deactivated.

reason: The case was forgotten in the implementation.

repair:

Bug 10: The icon of the module can not be changed in the ModuleTabsSidebar

symptom: If you clicked on "select icon" in the ModuleTabsSidebar, nothing happened.

reason: The feature was not implemented yet.

repair: Implemented the feature.

Bug 11: Connections were not deleted when deleting a module

symptom: If a module was deleted, the connected connections were not deleted.

reason: The case of connected connections was not considered in the implementation.

repair: Added the case to the remove-module method.

Bug 12: Modules and connections can be added and removed while simulation is running

symptom: It was possible to add and remove modules and connections while the simulation is running.

reason: There was no condition that modules and connections can be added and removed only if the simulation is not running.

repair: Added the condition.

Bug 13: Raster is not correct in scrolled areas

symptom: The raster was not drawn correctly when the window was scrolled.

reason: Some of the methods called when drawing the raster lead to this issue.

repair: Removed those methods. Obviously they are not necessary.

Bug 14: Selected connection is not highlighted

symptom: The currently selected connection was not highlighted in a different colour

reason: The activateElement-method has directly set the colour which should be used. This colour-change was lost with the time.

repair: The activateElement-method now only sets a bool-value, everything else is done by the draw-method.

Bug 15: No reminder to save the flowchart on exit

symptom: If the program was closed, there was no warning that the current flowchart is not saved.

reason: The feature was not implemented yet.

repair: Implemented the feature.

Bug 16: FunctionModule: Inputs are marked

symptom: The two input pins of the function module were always marked as selected, even if they were not selected.

reason: The function module is the only existing module with two pins. The normal pin size was too big, so it has to be recalculated according to the module height. The corresponding ModulePinGRs have to notify their ModulePinViews that the icons have changed. In the ModulePinView the icons were not updated in the update()-method.

repair: Added the icons to the update()-method in ModulePinView.

3.2 Model

Bug 20: Not-checking data::BasicCheckable signatures did not copy correctly

symptom: A partly not-checking BasicCheckable signature was not copied correctly and became completely checking again.

reason: data::detail::BasicSignatureComponentSignature's getCopy-method did not copy data-id.

repair: Deactivating checking on copy if original BasicSignatureComponentSignature was not-checking.

Bug 21: Dependency and adjacency lists were not updated on connection deletion

symptom: When a connection was deleted, the dependency and adjacency list of the respective modules were not updated.

reason: The case of removing modules was not thought of during implementation of module's dependency and adjacency list.

repair: adding a private method in FlowChart which is called on arrow removal and deletes the adjacencies or dependency if necessary with help from the detail::ModuleHelper.

Bug 22: Stored points in the GR produced memory loss.

symptom: Memory loss while execution

reason: The points in the GR are situated unmanaged and on the heap. Thus they were easily addressable from other classes, which only had to hold pointers on it

repair: storing points as attributes, and providing proper getters and setters

Bug 23: Methods returning text are returning `char*` which are null-pointers

symptom: empty strings – content missing

reason: these texts should not be strings because of some store/load issues, the nullpointers originated in the unknown size of the `char*`, thus the allocator instantly removed the whole text while returning it.

repair: return void instead, and get a stream as parameter, where the text is written on.

Bug 24: `FunctionModule` could not properly access `InputPins`

symptom: `run()` was not executable -> `FunctionModule` could not simulate

reason: Layer of different possible Templates could apply, and it is not clear, which to use.

repair: Temporarily restricting the `FunctionModule` on only using the (double) type

Bug 25: The `FunctionParser` could not recognize its own `toString()` text

symptom: unintuitiveness for users

reason: Recognition and Representation are disjointed (`FormulaElement <-> UnitD`), thus parsing the representation of a Formula was not always working

repair: added special shortcuts to the unit-recognition unit to also recognize inputs like `[m38]`, or `[kg-1]`, now the representation should be parsable

Bug 26: Segmentation fault during instantiation in model and view

symptom: `draw()` potentially calls the GR before the initialization of it

reason: the `draw()` method is part of `wxWidgets` management to provide content on the screen. It can always be called. -Even mid-construction.

repair: Only drawing and accessing the GR, when the View has been allowed to. This happens after the GR has been created.

3.3 Controller

Since the Controller really was not that much to test there were no issues we came across while testing it.

4 Unresolved known issues

Although we tried to resolve as many bugs as possible in the given time, there are still unresolved issues which we will outline here.

4.1 View

- Move modules in scrolled areas is not working correctly.
- In some cases, e.g. if two pins are in front of each other and the user clicks one time, a connection is created.

4.2 General

- The program segfaults on simulation start when there are two outgoing connections from one pin. However, we know the cause of this bug, which consists of ownership issues of the data on arrows. See: StdArrow.cpp for tried solution which unfortunately did not work. Furthermore, resetting data for the simulation is at the moment done in a not-so-nice way because of data ownership on the connections, see also StdArrow.cpp for more information.
- Very rarely: the program segfaults during simulation because of pure virtual method called.
- On Linux: saving segfaults.
- On Linux: after loading 3COM2C.flow: deleting lowest module followed by deleting middle module leads to uncaught exception.
- On Windows: some crashes which do not occur on linux (the linux application is more stable than the windows one)
- Loading works only onto empty flowcharts, otherwise the references from the connections are invalid
- The Module cannot alter its state. Our implementation of Model management is packed into a GUI class, which hinders the Model to change itself from within. (Because the GUI is not reachable by the Model)
This problem has a attempted fix in a different branch, which completely changes how the View gets updated. In the other version, all changes propagate to the Model. When the Model is changed, the View updates itself automatically. Unfortunately not all relevant bugs were found. -But the implementation is included anyway to hint some useful changes.

4.3 Modules

- FktModule cannot change the input pin count.
This is unresolved because we cannot alter the module from itself. The changes have to be managed by the FlowChartPanel, which is not accessible in modules.
- The Function Module segfaults when edit configuration is accessed a second time
- The live units layed on the inputs of a FktModule do not propagate to the output of the FktModule.
This is due to the assertion, that the output does not change mid simulation. It was planned to adapt the correct unit in the validate() method directly before the simulation. But with its implementation missing, there is no nice way of adapting the output to include unit-changes due to e.g. other FktModules.
- FktModule does not calculate in integer-mode. When run() is called only doubles are used. This behaviour is due to our common usage of doubles, to make the FktModule compatible to other Modules.
- Simulation does not work, if a FktModule is connected to a ChartModule while the FktModule is emitting a unit. Normaly the ChartModule should ignore the unit, and calculate anyway.

5 Local test cases from the requirements specification

5.1 Modules

We implemented one way to use a feature, so some of them are only available with one specific action.

5.1.1 Add a module to a flowchart (T10.1)

Bob would like to place a module.

prereq.: The module list is not empty. The flowchart is empty.

action: Bob clicks on the module and on the position he wants the module to be.

desired result: The module is placed on the given position.

test case succeeded: yes

abnormalities: none

5.1.2 Add a module to a flowchart (T10.2)

Bob would like to place a module.

prereq.: The module list is not empty. The flowchart is empty.

action: Bob clicks on the module in the list and on a position inside the list.

desired result: Nothing happens.

test case succeeded: yes

abnormalities: none

5.1.3 Select a module in a flowchart (T11)

Bob would like to select a module.

prereq.: The flowchart is not empty and contains a module.

action: Bob clicks on the module.

desired result: The module is selected and its sidebar is shown.

test case succeeded: yes

abnormalities: none

5.1.4 Deselect a module in a flowchart (T12)

Bob would like to deselect a module.

prereq.: The flowchart is not empty and contains a selected module.

action: Bob clicks outside of the module.

desired result: The module is deselected and its sidebar is hidden.

test case succeeded: yes

abnormalities: none

5.1.5 Deselect one module and select another one (T13)

Bob would like to deselect one module and select another one.

prereq.: The flowchart is not empty and contains one selected module and one not selected module.

action: Bob clicks in the module he wants to select.

desired result: The module is deselected and the other one is selected. The sidebar shows the data of the now selected module.

test case succeeded: yes

abnormalities: none

5.1.6 Delete a module (T14)

Bob would like to delete a module.

prereq.: The flowchart is not empty and contains one selected module.

action: Bob presses the key “Entf”.

desired result: The module is deleted and the sidebar is hidden.

test case succeeded: yes

abnormalities: none

5.1.7 Shift a module (T15)

Bob would like to shift a module.

prereq.: The flowchart is not empty and contains one module.

action: Bob left clicks the module and drag and drops it to the desired position. (arrow keys cannot be used, because they are used for scrolling)

desired result: The module is shifted and all its connections are adapted.

test case succeeded: yes

abnormalities: none

5.1.8 Turn a module (T16)

This feature is not implemented, but the context menu is.

5.1.9 Scale a module (T17)

This feature is not implemented, but the context menu is.

5.1.10 Change the number of FunctionModule inputs (T18)

This feature is currently deactivated. In the current state of the ViewManagement there is no “clean” way to control severe Model changes from concrete Modules, or the Model in general. This is the case because the FlowChartPanel class acts as a central GUI and thus model control unit. So there is no way to inform the FlowChartPanel of the Pin add / removal in the FunctionModule. If we just delete one pin from the view, the whole GUI will segfault.

To mitigate this problem we have implemented a whole new Observer based View life cycle system. With the new implementation, the Model will be automatically represented by the View. If the GUI wants to manipulate the FlowChart, it will request the changes in the Model. The View itself will automatically adapt.

Unfortunately we were not able to find all relevant bugs in this new implementation. Thus we had to stay with the old but working design. You can find the rather buggy, newer version in a different branch on GitLab.

5.1.11 Calculating with the FunctionModule (T19)

Bob wants to calculate m/s with in total three FunctionModules

prereq.: There are three FunctionModules in the FlowChart. Two of them are connected to the third. The third Module should be configured to X / Y

action: Click on first FunctionModule and configure it to output some meter value. Repeat it for the second one, but configure it with a time unit instead

desired result: when a simulation is ran, the output of the third Module should be outputting a calculated value with unit [m/s]

test case succeeded: no

abnormalities: The FunctionModule is not dynamically forwarding the current unit(must stay the same during simulation), thus the output value is correct, but the unit itself is not.

5.1.12 Connecting two module instances (T20)

Bob wants to connect two module instances.

prereq.: The two modules have to be drawn to the grid and their pins should be matching.

action: Click Output/Input Pin of the first module and then the Input/Output Pin of the second.

desired result: If a valid connection can be drawn it will be and is created in the Model.

test case succeeded: yes

abnormalities: none

5.1.13 Connecting two modules that cannot be connected (T21)

Bob wants to connect two module instances that do not match.

prereq.: The two modules have to be drawn to the grid.

action: Click Output/Input Pin of the first module and then the Input/Output Pin of the second.

desired result: Connection is not drawn

test case succeeded: yes

abnormalities: none

5.1.14 Modules stay connected under rotation (T22)

This feature is not implemented and therefore no tests for it exist.

5.1.15 Show data types on the connections (T23)

This feature is not implemented and therefore no tests for it exist.

5.1.16 Adding a rigid connection(T25)

Bob wants to connect two module instances with a rigid connection.

prereq.: The two modules have to be drawn to the grid. The Pins the user clicks have to be rigid Pins.

action: Click Output/Input Pin of the first module and then the Input/Output Pin of the second.

desired result: RigidConnection is drawn.

test case succeeded: yes

abnormalities: none

5.1.17 Adding a StdArrow(T26)

Bob wants to connect two module instances with a standard arrow.

prereq.: The two modules have to be drawn to the grid. The Pins the user clicks have to be StdPins.

action: Click Output/Input Pin of the first module and then the Input/Output Pin of the second.

desired result: Arrow is drawn.

test case succeeded: yes

abnormalities: none

5.1.18 Splitting an Arrow (T27)

This feature is not implemented and therefore no tests for it exist.

5.1.19 Splitting an Arrow incorrectly (T28)

This feature is not implemented and therefore no tests for it exist.

5.2 Simulation

5.2.1 Starting the Simulation (T30)

Bob would like to run one simulation cycle.

prereq.: No Prerequisites

action: Click “Step forward”.

desired result: The simulation starts stepping for the specified time of the simulation configuration.

test case succeeded: yes

abnormalities: none

5.2.2 Running one simulation cycle (T31)

Bob wants to start the simulation.

prereq.: No Prerequisites

action: Click “Start simulation”.

desired result: The simulation starts stepping until it is stopped or paused.

test case succeeded: yes

abnormalities: none

5.2.3 Pausing the simulation (T32)

Bob would like to pause the simulation.

prereq.: The simulation has been started. Two ConstantOutputModules were connected with each other as well as two RandomOutputModules.

action: Click “Pause simulation”.

desired result: The simulation finishes the correct simulation cycle and stops afterwards.

test case succeeded: yes

abnormalities: none

5.2.4 Resetting the simulation (T33)

Bob would like to reset(stop) the simulation.

prereq.: state as in desired result of T32.

action: Click “Stop simulation”.

desired result: The simulation stops completely, i.e. resetting all data to the standard value.

test case succeeded: yes

abnormalities: none

5.3 Saving and loading of a flowchart

Saving and loading of a flowchart is only working in most cases on windows. This may be caused by a used library, because the errors seem to be platform independent (segfault). Loading restores only the position and type of a module or connection, but it can be exchanged by adding some lines of setting configuration in the load method. Quite all available information from the flowchart elements is stored and restored in the corresponding DOM, but not set in the elements again.

Loading just adds flowchart elements to the already existing ones, so it should only be used at the start of the program with an empty flowchart. The possible solutions would be to use a new tab (but this ships with switching ownership of unique pointers), or to delete all elements in the flowchart.

5.3.1 Saving (T40.1)

Bob would like to save the current state of the flowchart.

prereq.: An unsaved flowchart has been created. Bob opened the file menu.

action: Click “Save”.

desired result: The program opens a file saving dialog and saving the flowchart succeeds.

test case succeeded: yes

abnormalities: some pop-up windows

5.3.2 Saving (T40.2)

Bob would like to save the current state of the flowchart.

prereq.: An formerly saved flowchart has been changed. Bob opened the file menu.

action: The user clicks “Save”.

desired result: The save-state is overwritten.

test case succeeded: no

abnormalities: The flowchart is not saved.

5.3.3 Loading (T41.1)

Bob would like to open a flowchart.

prereq.: The flowchart is empty. The user opened the file menu.

action: The user clicks “Open”.

desired result: The program opens a file opening dialog and opening the flowchart succeeds.

test case succeeded: yes

abnormalities: the user is notified that his changes (even if there are none) are not stored, some pop-up windows

5.3.4 Loading (T41.2)

Bob would like to open a flowchart.

prereq.: The flowchart is not empty and modified. The user opened the file menu.

action: The user clicks “Open”.

desired result: The program notifies that his changes are not stored and continues as the user wishes.

test case succeeded: yes

abnormalities: none

5.3.5 Add modules (T42)

Dynamic loading is not implemented, so this feature is not possible and is therefore not implemented.

5.3.6 Reload module list (T43)

Dynamic loading is not implemented, so this feature has no use case and is therefore not implemented.

5.4 Others

5.4.1 Module specific settings (T50 changed)

Bob would like to see a chart of the incoming data.

prereq.: There is a RandomOutputModule and a ChartModule which are connected. The simulation runs.

action: Bob clicks on the ChartModule and the “module sidebar” appears. Bob clicks on the “edit module” button.

desired result: The chart appears and shows the data coming from the previous module.

test case succeeded: yes

abnormalities: none

5.4.2 Changing the name of a module (T51)

Bob would like to change the name of the selected module.

prereq.: A module instance is selected and the “Info-Tab” is active in the Module-Sidebar.

action: Bob clicks inside the “module sidebar” on “representation”, enters the instance name “Demo”.

desired result: The module is now shown with its new name.

test case succeeded: yes

abnormalities: none

5.4.3 Hide module list sidebar (T52 changed)

Bob would like to have more space for his flowchart. Therefore, he would like to hide the module list sidebar.

prereq.: A flowchart is already created.

action: View -> Show module-list

desired result: The module list sidebar is hidden and the raster is made larger, so that it also covers the region which formerly was occupied by the sidebar.

test case succeeded: yes

abnormalities: none

6 Global test cases from the requirements specification

6.0.1 Create a flowchart (T100)

Bob would like to create and simulate a flowchart.

prereq.: The flowchart is empty and the module list is not empty.

action: Bob adds two modules to the flowchart (by clicking the module and the desired position in the flowchart, not by drag and drop as initially intended). Bob connects both modules by clicking on the output of the first one and the input of the second one. Bob runs a simulation cycle. Bob saves the flowchart.

desired result: The modules and connection are created in the flowchart, it is simulated and stored afterwards.

test case succeeded: yes

abnormalities: none

6.0.2 Error while creating a flowchart (T101)

Bob would like to create a flowchart but makes several mistakes trying to create it.

prereq.: The flowchart is empty and the module list is not empty.

action: Bob draws a Module from the Module sidebar but releases it on top of the Module sidebar. Afterwards Bob drags and drops two modules to the correct spot. Bob then tries to connect the two modules not regarding the fact that their inputs and outputs are not compatible.

desired result: The first module is not added to the current flowchart. Nothing happens after Bob releases it in the wrong spot. When Bob tries to connect the modules also nothing happens, so no connection is drawn.

test case succeeded: yes

abnormalities: none

6.0.3 Trying to open a corrupted flow file, thereafter opening a correct file and starting the simulation (T102 changed)

Bob tries to open a corrupted file, which fails. Thereafter he opens a correct file and starts the simulation.

prereq.: The flowchart is empty.

action: Bob tries to open a corrupted file. Thereafter he opens a correct file and starts the simulation.

desired result: After the corrupted file is tried to be opened, the file is not loaded and a popup window appears, showing that the file is incorrect. After opening the correct file, the opened flowchart appears and after starting the simulation, the simulation starts.

test case succeeded: yes

abnormalities: two Pop-Ups appear, a correct one and a wrong one

7 Known Windows/VS issues while developing

missing main	Linker->System: Subsystem:Windows
missing GetAdaptersInfo()	Linker->Input: Iphlpapi.lib (needed by poco), PocoFoundationmtd.lib (not PocoFoundationd.lib)
u_int	needs windows header: #include "winsock.h"
GetModuleFileName() issues	poco blocks windows GetModuleFileName() method (which wxWidgets needs): #define POCO_NO_UNWINDOWS

8 Conclusion

The test phase of our project was quite a success. We tested a lot of things and found many issues that were not only solved but enabled us to improve the overall performance of our software. Some of them even put us in the nice position to be able to quickly implement features of the Implementation phase since some major functionalities were not working back then. Therefore we are glad to pronounce that features like automatically updating the graphical user interface and easily adding new modules to the software are now fully functional which eases the usage of our software dramatically. We want to also thank our advisers for the patience and skill that they have shown throughout the whole project.