Extended Petri Net 3D Simulator
Test documentation

Group A

Introduction

Author: Mikko

This paper documents the software tests performed on the Extended Petri net 3D Simulator developed by Group A. Two types of testing have been defined: component and functional testing. For component testing the JUnit framework was used, while functional testing was performed by comparing the end result of the software with the expected result during several typical use cases.

Component testing

Author: Mikko

Components testing is performed on manually created methods. Furthermore, methods that can be tested in functional testing are omitted from component testing, to save time. As a result unit tests were done for petri net engine. The unit tests are divided in tables by software components, classes and methods. The tables are labeled according to the tested methods and they consist of:

- Test case name. Descriptive name of the test
- Test case description. Description of what was done to test the implementation.
- Expected result. Results that the are excepted from the test.
- Obtained result. Results that were actually obtained from the test.
- Verdict. Ok if test obtained and expected results match, failed otherwise.

Petri net engine unit tests

Author: Albert

To test the Petri net Engine, unit tests have been carried out using JUnit. All tests have been passed successfully, and can be found at:

/dk.dtu.se2.simulator.petrinet/src/dk/dtu/se2/simulator/petrinet/PetriNetEngineTest.java

In order to test it, a basic Petri Net has been modeled programmatically, as shown in Figure 1.

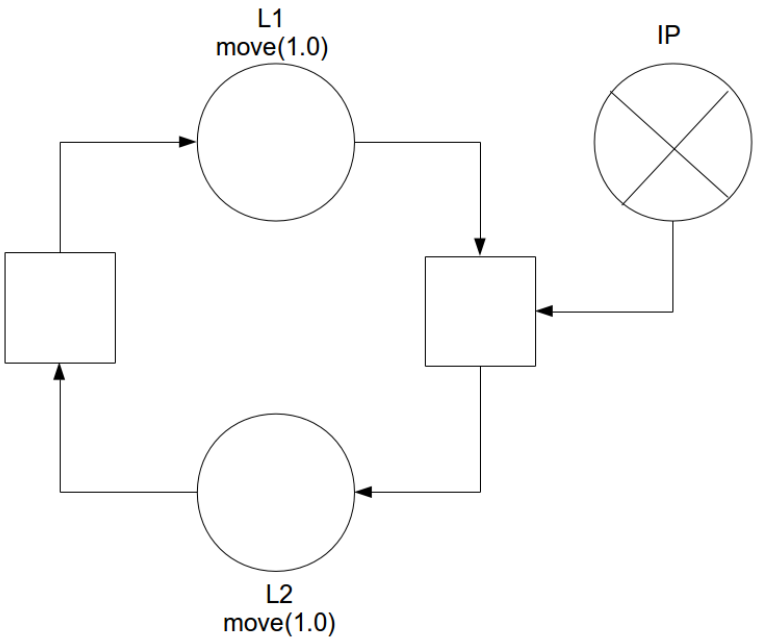
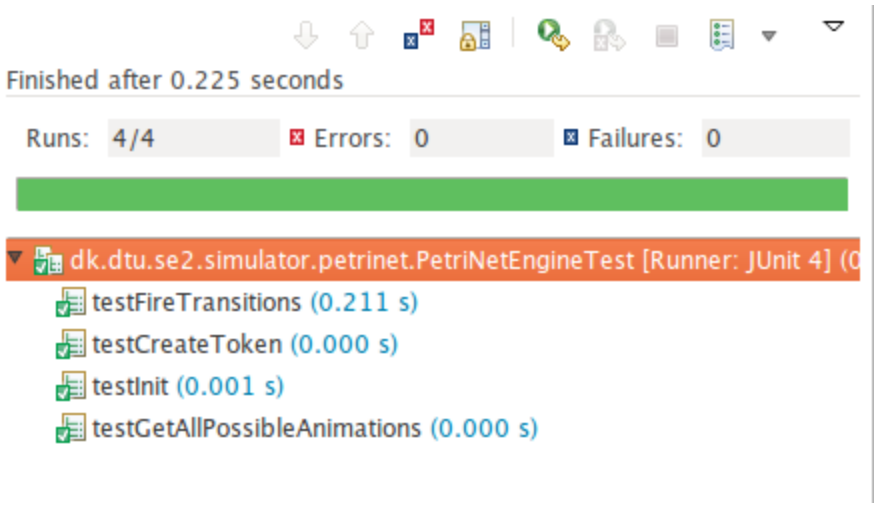


Figure 1: Petri net used for testing purposes

Test cases	description	excepted	obtained	verdict
init	After init, the list of animations to be runs at start is returned.	“L1”, “move(1.0)”	“L1”, “move(1.0)”	OK
getAllPossibleAnimations	The Petri net engine returns the list of all possible animations.	[(“L1”, “move(1.0)”), (“L2”, “move(1.0)”]	[(“L1”, “move(1.0)”), (“L2”, “move(1.0)”]	OK
fireTransitions	The Petri net engine returns the list of next animations after marking “L1” as finished.	[“L2”, “move(1.0)”]	[“L2”, “move(1.0)”]	OK
createToken	A token is created in IP.	Token in IP.	Token in IP.	OK



Functional testing

Author: Mikko

Functional testing is performed on several predefined use cases. Their aim is to test that each individual software component delivers the expected results as defined in their functional requirements. The structure of the tests consists of:

- Test name. Name of the test.
- Test description. Short description of what the test aims to test.
- Prerequisites. Tasks that need to be done before actual test can be tested.
- Steps. Tasks that are taken to do the test
- Expected result. What should happen when the steps have been done.
- Actual result. What happened when the steps were done.
- Verdict. Passed for when expected and actual results are the same, failed otherwise.

Geometry

Author: Mikko

Test 1

Name	**Add geometry objects to canvas**
Description	Define input point, two connectors and curved line on the graphical editor.
Prerequisites	None
Steps	1. Create input point

	2. Create two connectors 3. Create line 4. Create bend points to the line
Expected results	The input point, connectors and the line are visible on the canvas and tree view of the geometry has correct labels, appearance labels, token appearance labels and bend points for each geometry object.
Actual result	As expected result
Verdict	Passed

Test 2

Name	**Add duplicate label to geometry canvas**
Description	Put a label to a line, that the other line has.
Prerequisites	Two lines are defined in geometry.
Steps	1. Copy the name of line to other line 2. Open tree view of geometry 3. Validate geometry
Expected results	When geometry model is validated error message pops out.
Actual result	As expected result
Verdict	Passed

Test 3

Name	**Validate geometries on the simulation view**
Description	Validate that simulation has the same geometries as geometry has.
Prerequisites	Simple geometry and petri net defined, with correct appearance and configuration definitions.
Steps	1. Start simulation

Expected results	Graphical view of the geometry looks the same as simulator view looks.
Actual result	As expected result
Verdict	Passed

Petri Net Editor

Author: Thibaud

Test 1

Name	Create, edit, and delete Places
Description	The user should be able to create and edit and delete places.
Prerequisite	A PNML Model
Steps	1. Create a child to the Petri Net Doc with the name (ExtendedPetrinet) 2. Create a child “Page” on this newly created ExtendedPetrinet model 3. Double-click on the “Page” to launch the Graphical Editor. 4. Select a Place from the tool palette on the right 5. Click on the canvas where the place should be added Once a place is selected it can be deleted by clicking the “delete” button on the keyboard
Expected Result	A place is created and can be edited and deleted
Actual Result	A place is created and can be edited and deleted
Pass / Fail	Passed

Test 2

Name	Create, edit, and delete Tokens
Description	The user should be able to add tokens to places before launching the visualisation.
Prerequisite	A place created in the graphical editor
Steps	Select the place Open the properties view Select the “tokens” line and click the “…” rectangle at the end of the line Type a name for the token and click the “add” button to add it to the place To delete a token, select an already created token and click on “remove”
Expected Result	A token is created on the place and a circle is created in the graphical editor to notify the user that a token is present on the

	place. The number below the circle should indicate the number of tokens on the place
Actual Result	A circle is created with “1” written next to it
Pass / Fail	Passed

Test 3

Name	Create, edit, and delete Transitions
Description	The user should be able to add, edit and delete transitions.
Prerequisite	A PNML Model and a Page created (in the same way of the first test with places)
Steps	1. Double-click on the “Page” to launch the Graphical Editor. 2. Select a Transition from the tool palette on the right 3. Click on the canvas where the transition should be added Once a transition is selected it can be deleted by clicking the “delete” button on the keyboard
Expected Result	A transition is created
Actual Result	A transition is created
Pass / Fail	Passed

Test 4

Name	Create, edit and delete Arcs.
Description	The user should be able to link places to transitions with arcs. The user shall not be able to link two places together.
Prerequisite	Two places and a transition created in the page. (to test the second requirement)
Steps	Select the “Arc” function in the tool palette Click on the desired source and drag the cursor to the desired target.
Expected Result	An arc is created between a place and a transition An arc cannot be created between two places or two transitions
Actual Result	The same
Pass / Fail	Passed

Test 5

Name	Define a Geometry label and Animation Label to a Place
Description	The user should be able to link a geometry label and animation label to a place.

	In the case of an animation label, the text should be parseable, otherwise a warning is displayed above the label
Prerequisite	A place created in the page
Steps	Select the “Label” in the tool palette. Click on the canvas where the label should be created Hover the mouse above the newly created label and click on the outgoing arc with a square at the end and drag the mouse to a place. Click once on the label and enter the desired value with the keyboard
Expected Result	A label is created and linked to a place A label can be edited A label displays a warning sign if the “Animation Label” text is not parseable
Actual Result	The same
Pass / Fail	Passed

Test 6

Name	Define an InputPlace label to a Place
Description	The user has to be able to define whether or not a place is an input place.
Prerequisite	A place created on a page
Steps	Select the place Open the properties view Click on the “InputPlace label” line and start typing either “false” or “true” and select the right one.
Expected Result	An InputPlace label is added to the place
Actual Result	The same
Pass / Fail	Passed

Test 7

Name	The Petri net editor should allow undo/redo of actions
Description	The user has to be able to undo or redo his previous actions.
Prerequisite	A PNML Document and a Page
Steps	Create a random child from the tool palette such as a Place Go to the edit menu of eclipse Press Undo Go to the edit menu of eclipse Press Redo
Expected Result	A place can be removed and then re-added (works with all the other children)
Actual Result	The same

Pass / Fail	Passed

Configuration Editor

Author: Monica

Test 1

Name	**Load Petri net doc resource**
Description	A .pnml file is loaded from a specified folder
Prerequisites	Eclipse running, Configuration file created, .pnml file created
Steps	1. Click Configuration file 2. Click Load resource… 3. Browse workspace 4. Select resource and load it
Expected results	The selected file appears as a resource in the configuration editor
Actual result	Petri net doc resource loaded in the configuration editor
Verdict	Passed

Test 2

Name	**Load Geometry resource**
Description	A .geometry file is loaded from a specified folder
Prerequisites	Eclipse running, Configuration file created, .geometry file created
Steps	1. Click Configuration file 2. Click Load resource… 3. Browse workspace 4. Select resource and load it
Expected results	The selected file appears as a resource in the configuration editor

Actual result	Geometry resource loaded in the configuration editor
Verdict	Passed

Test 3

Name	**Load Appearance resource**
Description	A .appearance file is loaded from a specified folder
Prerequisites	Eclipse running, Configuration file created, .appearance file created
Steps	1. Click Configuration file 2. Click Load resource… 3. Browse workspace 4. Select resource and load it
Expected results	The selected file appears as a resource in the configuration editor
Actual result	Appearance resource loaded in the configuration editor
Verdict	Passed

Test 4

Name	**Assign Petri net doc resource to Petrinet property**
Description	A Petri net doc file previously loaded is assigned to the Petrinet property of the Configuration object.
Prerequisites	Eclipse running, Configuration file created, Petri net doc resource loaded
Steps	1. Click Configuration file 2. Open properties view 3. Select Petrinet property 4. Click drop down menu and select resource

Expected results	The selected resource is assigned to the Configuration's property
Actual result	The Petrinet property has a resource value assigned
Verdict	Passed

Test 5

Name	**Assign Geometry resource to Geometry property**
Description	A Geometry file previously loaded is assigned to the Geometry property of the Configuration object.
Prerequisites	Eclipse running, Configuration file created, Geometry resource loaded
Steps	1. Click Configuration file 2. Open properties view 3. Select Geometry property 4. Click drop down menu and select resource
Expected results	The selected resource is assigned to the Configuration's property
Actual result	The Geometry property has a resource value assigned
Verdict	Passed

Test 6

Name	**Assign Appearance resource to Appearance property**
Description	An Appearance file previously loaded is assigned to the Geometry property of the Configuration object.
Prerequisites	Eclipse running, Configuration file created, Appearance resource loaded

Steps	1. Click Configuration file 2. Open properties view 3. Select Appearance property 4. Click drop down menu and select resource
Expected results	The selected resource is assigned to the Configuration's property
Actual result	The Appearance property has a resource value assigned
Verdict	Passed

Appearance Editor

Author: Morten

Test 1

Name	**Create AObject**
Description	Create an AObject in the Appearance file.
Prerequisites	Eclipse running, Appearance file created, Appearance file loaded.
Steps	1. Click on the left arrow of the loaded appearance file. 2. Select the Appearance node 3. Right-click the appearance node and select New Child > AObject.
Expected results	A new AObject will be created as child of the Appearance node.
Actual result	A new AObject was created as a child of the Appearance node.
Verdict	Passed

Test 2

Name	**Add label**
Description	Input the name of an appearance label for an

	AObject.
Prerequisites	Eclipse running, Appearance file created, Appearance file loaded, AObject created.
Steps	1. Select the AObject 2. In the Properties view change the “label” property.
Expected results	The user will be allowed to change the label to anything.
Actual result	The label was changed successfully.
Verdict	Passed

Test 3

Name	**Load 3D Object**
Description	Input the path of a predened 3D object file for an AObject.
Prerequisites	Eclipse running, Appearance file created, Appearance file loaded, AObject created, 3D object file placed in jMonkeyEclipse/assets/Models folder.
Steps	1. Select the AObject 2. In the Properties view press the “...” button in the Object3D property. 3. Select the 3D model file inside the Models folder.
Expected results	The Object3D property will change to the path of the chosen file, relative to the jMonkeyEclipse folder.
Actual result	The property was changed to “Models/filename”
Verdict	Passed

Test 4

Name	**Load texture**
Description	Input the path of a predened texture file for an AObject.
Prerequisites	Eclipse running, Appearance file created, Appearance file loaded, AObject created, texture file placed in the jMonkeyEclipse/assets/Textures folder.
Steps	1. Select the AObject 2. In the Properties view press the "..." button in the Texture property. 3. Select the texture file inside the Textures folder.
Expected results	The Texture property will change to the path of the chosen file, relative to the jMonkeyEclipse folder.
Actual result	The property was changed to "Textures/filename"
Verdict	Passed

Test 5

Name	**Save appearance**
Description	The user is able to save the appearance file.
Prerequisites	Eclipse running, Appearance file created, Appearance file loaded, changes to the Appearance file have been made.
Steps	1. Select File > Save or 1. Press Ctrl + S
Expected results	The appearance file will be saved with the changes.
Actual result	The appearance file was saved with the changes.
Verdict	Passed

Simulator

Author: Anders

Test 1

Name	**Play Simulation**
Description	The user should be able to change the state of the simulation to playing.
Prerequisites	Eclipse running, configuration file created correctly, simulation opened, the state of the simulation must be paused.
Steps	Use the left mouse button to click the play button on the screen.
Expected results	The simulation begins playing and the play button changes label and becomes a pause button.
Actual result	The simulation began playing and the play button changed label and became a pause button.
Verdict	Passed

Test 2

Name	**Pause Simulation**
Description	The user should be able to change the state of the simulation to paused.
Prerequisites	Eclipse running, configuration file created correctly, simulation opened, the state of the simulation must be playing.
Steps	Use the left mouse button to click the pause button on the screen.
Expected results	The simulation pauses and the pause button changes label and becomes a play button.
Actual result	The simulation paused and the pause button changed label and became a play button.

Verdict	Passed

Test 3

Name	**Interact with Input Point**
Description	The user should be able to interact with input points.
Prerequisites	Eclipse running, configuration file created correctly, simulation opened, the state of the simulation must be playing.
Steps	Use the left mouse button to click an input point.
Expected results	Depending on the type of input point, the outcome should be either that tokens stop at the input point, or that tokens change paths.
Actual result	When the input point is used as a stop/go: tokens stop and go as the input point is clicked. When the input point is used as a path changer: tokens move along the respective paths as the input point is clicked.
Verdict	Passed

Test 4

Name	**Reset simulation**
Description	The user should be able to reset the simulation to it's initial state.
Prerequisites	Eclipse running, configuration file created correctly, simulation opened, the state of the simulation must be playing or paused.
Steps	Use the left mouse button to press the reset button.
Expected results	All tokens should return to their initial positions, thus also resetting input points.

Actual result	All tokens returned to their initial positions, and input points returned to their initial configuration.
Verdict	Passed

Test 5

Name	**End simulation**
Description	The user should be able to end the simulation.
Prerequisites	Eclipse running, configuration file created correctly, simulation opened, the state of the simulation must be playing or paused.
Steps	Press the ‘esc’ keyboard button, or use the left mouse button to close the window.
Expected results	The window should close, thus ending the simulation.
Actual result	The window closed, and the simulation ended.
Verdict	Passed