Test Result

02162 Software Engineering 2

Fall 2009

December 16th 2009

Group 1

s090709 - Asta Maknickaite
s091370 - Khurram Bashir
s091771 - Eirini Arvaniti
s090842 - Olivier Rouiller
s092527 - Lucie Urbanova
s052608 - Jonas Frederiksen
s072643 - Maysa Turki Abed Jamil
s071312 - Jakob Hommelhoff Jensen
s090975 - Jing Lv

Table of contents

1. Introduction

We have tested our product as we planned. We had implemented Junit test on the parser part and the evaluation part. We had implemented the system test (functional test) on the product, which is based on comparison of the expected value with the actual value. We will show the results of the test in tables in the next sections. We will show also our system test to some of the use cases we have defined it in our system. Furthermore we will show some graphic figures that show our coverage test to the codes using dJunit.

2. The Unit test:

Is the test that we implemented on the classes using junit test, and here is the three different parts of the case tool that we implemented the junit on.

2.1 The evaluation test: (Asta)

We have tested the evaluation part using junit and we will illustrate the results in the following tables:

The first table for the Operator application:

	BoolenOp	StringOp	ArithmeticOp	ComparisonOp
-isValid()	+	-	+	+
-toString()	-	-	-	-
-eval()	+	-	+	+
-getType()	+	+	+	+

Table -1- **Testing the Operator Application**

this table for testGetType() method:

Operation	**ComparisonOp**	**Expected result**	**Result**
0 == 1	-equal	boolean	boolean
0 != 1	-notEqual	boolean	boolean
0 < 1	-less	boolean	boolean
0 <= 1	-lessOrEqual	boolean	boolean

0 > 1	-greater	boolean	boolean
0 >= 1	-greaterOrEqual	boolean	boolean
Operation	**ArithmeticOp**	**Expected result**	**Result**
att1 = 0 + 5	-adition	integer	integer
att1 = (5+0)*5	-multiplication	integer	integer
Operation	**BooleanOp**	**Expected result**	**Result**
(0>=1)&(1==1)	-and	boolean	boolean
(0>=1)|(1==1)	-or	boolean	boolean
!(1==1)	-not	boolean	boolean
(0>=1) ^ (1==1)	-xor	boolean	boolean
Operation	**StringOp**	**Expected result**	**Result**
string test	-concatenation	string	string
test	-substring	string	string

Table -2- Test method **testGetType()**with attribute

This table for testEval() method:

Operation	**ComparisonOp**	**Expected result**	**Result**
0 == 1	-equal	false	false
1 == 1	-equal	true	true
0 != 1	-notEqual	true	true
0 < 1	-less	true	true
0 <= 1	-lessOrEqual	true	true
0 > 1	-greater	false	false
0 >= 1	-greaterOrEqual	false	false
! (0 >= 1)	-less	true	true
3.14 < 5.5	-less	true	true

Operation	ArithmeticOp	Expected result	Result
att1 = 0 + 5	-adition	att1 = 5	att1 = 5
att1 = (5+0)*5	-multiplication	att1 = 25	att1 = 25
Operation	**BooleanOp**	**Expected result**	**Result**
(0>=1)&(1==1)	-and	false	false
(0>=1)|(1==1)	-or	true	true
!(1==1)	-not	false	false
(0>=1) ^ (1==1)	-xor	true	true

Table -3- Test method **testEval()** with attribute

This table for the testIsValid() method:

Operands Type	ComparisonOp	Expected result	Result
0 == 0	-equal	true	true
0 != 1	-notEqual	true	true
0 < 1	-less	true	true
0 <= 1	-lessOrEqual	true	true
0 > -1	-greater	true	true
0 >= 0	-greaterOrEqual	true	true
Operation	**ArithmeticOp**	**Expected result**	**Result**
att1 = 5 - 2	-substraction	true	true
att1 = (5+0)*5	-multiplication	true	true
Operation	**BooleanOp**	**Expected result**	**Result**
(0>=1)&(1==1)	-and	true	true
(0>=1)|(1==1)	-or	true	true
(0>=1) ^ (1==1)	-xor	true	true

Table -4- Test method **testIsValid()** with attribute

For the case isValid() should return false, the parser raise an exception so the tests are commented but give the right results.

2.2 The parser test: (maysa)

We have tested the parser and we think the parser has passed the test partially because there were just two bugs. We tried to correct them but couldn't figure out the reason and we didn't have much time to correct these bugs. The first bug is when you are in the component definition diagram editor, if you change the transition label in the box that there is on the editor, it doesn't update correctly. It always keeps the information of the previous transition label. But if you change the label in the properties tab everything works Fine.
The second bug is when renaming the attributes from the properties tab; one cannot use them in the transitions. But if the renaming of the attributes is done in the box on the diagram, everything works well.
Here we will illustrate the most important test case's results of the test cases of the parser:

Test Cases	Description	Expected Result	Actual Result	Approve
LegalTransitionLabel	enter a whole legal transition label	No Exception	No Exception	ok
TransitionLabel without assignment	Enter a transition label without assignment	No Exception	No Exception	ok
TransitionLabel Without condition	Enter a transition label without condition	No Exception	No Exception	ok
TransitionLabel without InMessage	Enter a transition label without In message	No Exception	No Exception	ok
TransitionLabel without Out Message	Enter a transition label without Out Message	No Exception	No Exception	ok
TransitionLabel without Condition	Enter a transition label without A condition	No Exception	No Exception	ok
TransitionLabel with OutMessage	Enter a transition label with OutMessage has more than 1 parameter	No Exception	No Exception	ok
illegal TransitionLabel	Enter a transition label with no separators between them	Throw Exception	Exception is thrown	ok

TransitionLabel with condition	Enter a transition label with condition (integer < true)	Throw Exception	Exception is thrown, operand are not compatible	ok
TransitionLabel with InMessage	Checking the parameter of the InMessage	No Exception	No Exception	ok
TransitionLabel with OutMessage	Checking the parameters of the Out Message	No Exception	No Exception	ok
TransitionLabel with condition	Use an identifier in the condition, which is not defined	Throw Exception	Exception is thrown	ok
TransitionLabel with InMessage	The parameter of the In Message is not defined	Throw Exception	Exception is thrown	ok
TransitionLabel with OutMessage	The parameters of the Out Message is not defined	Throw Exception	Exception is thrown	ok
TransitionLabel with assignment	Use an identifier in the assignment, which is not defined	Throw Exception	Exception is thrown	ok
Empty TransitionLabel	Enter empty transition with only separators	No Exception	No Exception	ok
TransitionLabel with identical name for attribute and message parameter	Enter a message parameter with same name of an attribute	No Exception and rename the attribute name	No Exception and the attribute name is renamed	ok

Table -5- the parser Test

2.3 The dashboard test: (Jing)

We will illustrate the test cases that we have tested in table 6 and will show the actual result as a use cases in the following section.

Name	Description	expected result	result
Create Dashboard	The user create a dashboard	Dashboard created	OK
Load deployment resource	The user load a deployment to the dashboard	deployment loaded	OK
Create ValueProvider	Add a Bus value provider to the dashboard	Bus VP created	OK
	Add a Port value provider to the dashboard	port VP created	OK
	Add a Summarization value provider to the dashboard	Summarization VP created	OK
	Add a State Name value provider to the dashboard	State Name VP created	OK
	Add an Attribute value provider to the dashboard	Attribute VP created	OK
Create Visuals Add visual to the dashboard	Add label visual to the dashboard	label created	OK
	Add Image visual to the dashboard	Image Visual created	OK
	Add Range Image visual to the dashboard	Range Image Visual created	
Create Action	Add Action to the dashboard		
	Add Set State Action to the dashboard	Set State action created	OK
	Add Set Attribute Action to the dashboard	Set Attribute action created	OK
Create Action Visual	Add a Action Visual to the dashboard		
	Add a button to the dashboard	button created	OK
	Add a Arrow Scroll Bar to the dashboard	Arrow Scroll Bar Created	OK
	Add a RadioButtonComposite to the dashboard	A RadioButtonComposite Created	OK
	Add a RadioButton as a child of the RadioButtonComposite	A RadioButton Created	OK
	Add a Text Box to the dashboard	The text Box created	OK

Table -6 Check out

For the Text box a label feature for name is not working. When user types the name of text box, it will not be set. So it will also not appear as a Label.

2.3.1 Adding value provider to Visual –Lucie

The RadioButtonComposite can be used for types: Boolean, Integer, Float. It should be used also for String. This is an error. RadioButtonComposite has in types twice Integer, instead of one there should be a String.

For all of these connections between value provider and visual:
If the type of Value provider is not equal with one of the visual's types, then the warning is shown and the connection is not created. You can see it in the tree editor for dashboard. For GMF: The connection is also not set, but when you click again to the list of value providers, you will see the wrong one still in the right hand side (Feature). If you switch to the tree editor and then back to the GMF the value provider is not shown.

Type of Value provider	Types for Label	expected result	result
String	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Boolean	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Integer	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Float	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK

Table 7 Test Result for adding value provider to the Label

Type of Value provider	Types for Textbox	expected result	result
String	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Boolean	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Integer	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Float	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK

Table 8 Test Result for adding value provider to the Textbox

Type of Value provider	Types for ImageVisual	expected result	result
String	Boolean, String	The system created the reference between the value provider and the visual	OK
Boolean	Boolean, String	The system created the reference between the value provider and the visual	OK
Integer	Boolean, String	The system displayed a warning and the reference is not set	OK
Float	Boolean, String	The system displayed a warning and the reference is not set	OK

Table 9 Test Result for adding value provider to the Image Visual

Type of Value provider	Types for RangeVisual	expected result	result
String	Integer, Float	The system displayed a warning and the reference is not set	OK
Boolean	Integer, Float	The system displayed a warning and the reference is not set	OK
Integer	Integer, Float	The system created the reference between the value provider and the visual	OK
Float	Integer, Float	The system created the reference between the value provider and the visual	OK

Table 10 Test Result for adding value provider to the Range Visual

Type of Value provider	Types for RadioButton	expected result	result
String	Boolean, Integer, Float,	The system created the reference between the value provider and the visual	FAILED
Boolean	Boolean, Integer, Float	The system created the reference between the value provider and the visual	OK
Integer	Boolean, Integer, Float	The system created the reference between the value provider and the visual	OK
Float	Boolean, Integer, Float	The system created the reference between the value provider and the visual	OK

Table 11 Test Result for adding value provider to the RadioButtonComposite

Type of Value provider	Types for Slider	expected result	result
String	Integer, Float	The system displayed a warning and the reference is not set	OK
Boolean	Integer, Float	The system displayed a warning and the reference is not set	OK
Integer	Integer, Float	The system created the reference between the value provider and the visual	OK
Float	Integer, Float	The system created the reference between the value provider and the visual	OK

Table 12 Test Result for adding value provider to the ArrowScrollBar (slider)

Type of Value provider	Types for Button	expected result	result
String	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Boolean	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Integer	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK
Float	Boolean, Integer, Float, String	The system created the reference between the value provider and the visual	OK

Table 13 Test Result for adding value provider to the Button

2.3.2 Connect Action to Action Visual – Jing

Connect Action to Action Visual	Connect a action to a action visual	The system create the reference between the action and the action visual	
	Connect Button to Set Attribute Action	The system create the reference between the button and the action	OK
	Connect Button to Set State Action	The system create the reference between the button and the action	OK
	Connect RadioButtonComposite to Set Attribute Action	The system create the reference between the button and the action	OK
	Connect RadioButtonComposite to Set State Action	The system create the reference between the button and the action	OK
	Connect Arrow Scroll Bar to Set Attribute Action	The system create the reference between the button and the action	OK
	Connect Arrow Scroll Bar to Set	The system create the reference between	OK

	State Action	the button and the action	
	Connect Text Box to Set Attribute Action	The system create the reference between the button and the action	OK
	Connect Text Box to Set State Action	The system create the reference between the button and the action	OK

Action and Action visuals - types - Lucie

Every action has a reference to CASE Tool type. For SetAttributeAction the type depends on the data type of the attribute. For the SetState action the type is always String. Action visuals have (like Visuals) list of CASETool types for which they can be used. When connecting the particular Action to the Action visual, it needs to be check, it the type of Action is equals with one of Action visual's type. This is not checked and it is an error. There is a command CheckActionsTypesCommand, but this command is not invoked anywhere, because it causes some problems. So for now the functionality is not implemented.

2.3.3 Dashboard during runtime - Lucie

All visual and action visuals from Dashboard editor are correctly created as a runtime instances. When a value provider is not connected to action visual, user can interact with this action visual, but action visual has not a functionality of visual. When the value provider is connected, the action visual has also the functionality of visual. You can see if the value (for example from attribute value provider) is changed via this action visual. This is working for all action visuals, except button.

Functionality of ArrowScroll Bar

Increment	Minimum	Maximum	expected result	result
1	0	4	Slider can be moved from 0 to 4	OK
1	2	2	Slider cannot be moved	OK
1	10	4	An error or warning should appear during edit mode	FAILED

Table 14 Functionality of ArrowScroll Bar

Functionality of Range image visual

Value	Mapped to
0	Image Sun
1	Image Light rain
3	Image Heavy rain

Table 1 Definition of mapping

Value	Expected image	Result image	Result
0	Image Sun	Image Sun	OK
1	Image Light rain	Image Light rain	OK
2	Image Light rain	Heavy rain	FAILED
3	Image Heavy rain	Image Heavy rain	OK

Tabel 26 Behaviour during runtime

There are no other known errors for the Dashboard runtime. Interaction visuals like TextBox, Image mapping, and RadioButton work properly. All Visuals connected to the same value provider (included interaction visuals with value provider set) are shown the same value at the same time.

2.4 Use case test

The following are the use cases from the analysis. They are used to test whether the complete system lives up to the requirements initially specified.

2.4.1 ValueProvider

Actor	User
Pre-Condition	The deployment is created and loaded
Post-Condition	The value provider is created
Scenario	1. The user Right-click the Dashboard root element or the free space 2. The user choose the ’New child’. 3. The user select the Value Provider which can be ’Sumarization VP’, ’Attribute VP’, ’Bus VP’, ’Port VP’, ’State Name VP’ from the menu. 4. The user gives the name of ValueProvider and fill in some other specific properties if needed. 5. The system creates an object of the Value Provider.
Alternative scenario	None

Table 7: Create value provider

Actor	User
Pre-Condition	A value provider is created
Post-Condition	The value provider is deleted
Scenario	1. The user selects the value provider which must be deleted 2. The user right-click the value provider and select ’Delete’ 3. The system deletes the reference between the value provider and the source in the deployment 4. The system deletes the value provider from the dashboard
Alternative scenario	None

Table 8: Delete value provider

Actor	User
Pre-Condition	The deployment is loaded and the value provider is created
Post-Condition	The value provider is connected to its source
Scenario	1. Choose source from the property window of the value provider 2. The system is binding the source from the deployment model to the value provider
Alternative scenario	None

Table 9: Connect value provider to source

2.4.2 Visual

Actor	User
Pre-Condition	The Dashboard is launched
Post-Condition	The visual has been created
Scenario	1. The user Right-click the Dashboard root element or the free space 2. The user choose the ’New child’. 3. The user chooses one of the visual types which can be ’Image Visual’, ’Range Image Visual’ and ’Label’ from the menu 4. The user gives the visual a name 5. The system creates a visual
Alternative scenario	None

Table 10: Create visual

Actor	User
Pre-Condition	The Dashboard is launched; the value provider and visual objects are created
Post-Condition	The visual object and its reference to value provider object are destroyed
Scenario	1. The user selects the visual which must be deleted 2. The user right-click the visual and select 'Delete' 3. The System destroys the reference between the visual object and the value provider object 4. The system destroys the visual object Alternative scenario None
Alternative scenario	None

Table 11: Delete visual

Actor	User
Pre-Condition	The value provider and the visual are created
Post-Condition	The visual has been connected to a value provider
Scenario	1. The user choose a value provider from the property window of the visual 2. The system creates a reference between the value provider and the visuals
Alternative scenario	None

Table 12: Connect visual to a value provider

2.4.3 Action

Actor	User
Pre-Condition	The Dashboard is launched
Post-Condition	The Action object is created
Scenario	1. The user Right-click the Dashboard root element or the free space 2. The user choose the 'New child'. 3. The user chooses one of the Actions which can be 'Set State', 'Set Attribute' from the menu. 4. The system creates an object of Action
Alternative scenario	None

Table 13: Create action

Actor	User
Pre-Condition	An action is created
Post-Condition	The action is deleted
Scenario	1. The user selects the action which must be deleted 2. The user right-click the action and select 'Delete' 3. The system deletes the reference between the action and the source in the deployment 4. The system deletes the action from the dashboard
Alternative scenario	None

Table 14: Delete action

Actor	User
Pre-Condition	The Dashboard is launched
Post-Condition	The action visual object is created
Scenario	1. The user Right-click the Dashboard root element or the free space 2. The user choose the 'New child'. 3. The user select one of the action visuals which can be 'Arrow Scroll Bar', 'Button', 'Radio Button Composite', 'TextBox'. 4. The System creates an object of the action visual
Alternative scenario	None

Table 15: Create action visual

Actor	User
Pre-Condition	A Radio Button Composite is created
Post-Condition	The Radio Button is created
Scenario	1. The user Right-click one of the Radio Button Composite 2. The user Right-click the Radio Button Composite 3. The user choose the ’New child’. 4. The user select the ’Radio Button’
Alternative scenario	None

Table 16: Create radio button

Actor	User
Pre-Condition	The Dashboard is launched, and the action visual is created
Post-Condition	The action visual and the reference to the Action object are destroyed
Scenario	1. The user chooses an action visual object from the Dashboard to delete 2. The system destroys the object of action visual 3. The system destroys the reference to the Action object
Alternative scenario	None

Table 17: Delete action visual

Actor	User
Pre-Condition	The Dashboard is launched; the action visual and Action objects are created
Post-Condition	The connection is established between Action and action visual objects
Scenario	1. The user chooses an Action Source from action visual object property window 2. The System binds the Action object Source to the action visual object
Alternative scenario	None

Table 18: Connect action an action visuals

3. The integration test:

We tested some of classes which contain more than one function as parser class. We test it as a whole class to check whether the class works properly or not. The test was successful. (See source code in appendix).

4. The system test (Maysa)

We will illustrate the test of some of use cases:

4.1 Component editor part:

1. Create component definition **(common case)**

- **Test case specification identifier:** Component editor.
- **Test items:** testing Component editor.
- **Input specifications:** run the case tool.
- **Output specification:** the expected output is component editor is launched and component definition object with an empty automaton is created.
- **Environment needs:** standard pc, Eclipse software.
- **Special procedural requirements:** None.
- **Inter-case dependencies:** the case tool is launched
- **The Test Result:** the component editor is opened and component definition object with an empty automaton are created.

2. Add attribute to the component definition **(special case)**

- **Test case specification identifier:** attribute definition.
- **Test items:** testing adding attribute to the Component definition with same name of the message parameter.
- **Input specifications:** adding attribute by pressing on attribute in plate menu.
- **Output specification:** the expected output is an attribute is created and renaming the attribute with another name.
- **Environment needs:** standard pc, Eclipse software.
- **Special procedural requirements:** None.
- **Inter case dependencies:** the component definition exists.
- **The Test Result:** An attribute is created and the attribute is renamed with another name.

3. Create transition of an automaton **(illegal input)**

- **Test case specification identifier:** automaton's transition.
- **Test items:** testing the transition label parser.
- **Input specifications:** an automaton is created and creating a transition between the states of the automaton and entering an illegal label.
- **Output specification:** the expected output is exception is thrown with an error message showing there is an error in the transition label.
- **Environment needs:** standard pc, Eclipse software.
- **Special procedural requirements:** None.
- **Inter case dependencies:** the automaton object is created. Source and target states exist and are attached to the automaton object.
- **The Test Result:** an error message is shown that the transition label is illegal**.**

5. Conclusion

All level of tests is implemented and at the first stage we found some bugs and mistakes in the parser, which we fixed. According to our test we conclude that parser is working partially, because we have just two bugs as we mentioned earlier.

We used DJunit to show the coverage report to the codes, but the problem is when we wanted to install the DJunit again (because I faced some problems with my laptop) we couldn't do that, because there are some problems in the DJunit site. So we couldn't show the graphical figures for the coverage reports.

6. Appendix (Maysa)

Test codes:

BinaryOpTest

```
package de.upb.swt.mcie.formulas;

import static org.junit.Assert.*;

import junit.framework.TestCase;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class BinaryOpTest extends TestCase {

        private Formula left;

        private Formula right;

        private BinaryOp binaryOpTest  = new BinaryOp(4 ,left ,right);

        @Before

        public void setUp() throws Exception {

        }

        @After

        public void tearDown() throws Exception {

        }

        @Test

        public void testIsBooleanFormula() {

                boolean test1=binaryOpTest.isBooleanOperator();

                boolean test2=binaryOpTest.isRationalOperator();

                assertTrue((test1 ||test2));

        }

        @Test

        public void testGetLeftOperand() {

                assertEquals(left,binaryOpTest.getLeftOperand());
        }
        @Test
        public void testBinaryOp() {
                assertEquals(true,binaryOpTest.isBooleanOperator());
        }
        @Test
        public void testGetRightOperand() {
                assertEquals(right,binaryOpTest.getRightOperand());
                }
        @Test
        public void testIsBooleanOperator() {
                boolean test1=binaryOpTest.isBooleanOperator();
                assertTrue((test1 ));
        }
        @Test
        public void testIsRationalOperator() {
        BinaryOp binaryOpTest2  = new BinaryOp(9 ,left ,right);
        boolean test2=binaryOpTest2.isRationalOperator();
        assertTrue((test2 ));
        }
        @Test
        public void testIsArithmeticOperator() {
                BinaryOp binaryOpTest3  = new BinaryOp(5 ,left ,right);
        boolean test3=binaryOpTest3.isArithmeticOperator();
        assertTrue((test3 ));
        }
        @Test
        public void testGetName() {
         BinaryOp binaryOpTest4  = new BinaryOp(4 ,left ,right);
         assertEquals("nand",binaryOpTest4.getName());
        }
```

```
                @Test
                public void testGetRepresentation() {
                                BinaryOp binaryOpTest5  = new BinaryOp(4 ,left ,right);
                 assertEquals("!&",binaryOpTest5.getRepresentation());
                }
}
```

FormulaAssignmentTest

```
package de.upb.swt.mcie.formulas;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
public class FormulaAssignmentTest {
private Formula formula = null;
private FormulaAssignment formulaassignment=new FormulaAssignment("portname", formula);
                @Before
                public void setUp() throws Exception {
                }
                @After
                public void tearDown() throws Exception {
                }
                @Test
                public void testGetAttr() {
                                assertEquals("portname",formulaassignment.getAttr());
                }
                @Test
                public void testGetFormula() {
                                assertEquals(formula,formulaassignment.getFormula());
                }
```

```
		@Test
		public void testIsEmpty() {
				assertNotNull(formulaassignment.getAttr());
				assertNull(formulaassignment.getFormula());
				assertTrue(formulaassignment.isEmpty());
		}
}
```

FormulaBooleanConstantTest

```
package de.upb.swt.mcie.formulas;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
public class FormulaBooleanConstantTest {
		private FormulaBooleanConstant formulaBoolConsTest = new
FormulaBooleanConstant(true);
		@Before
		public void setUp() throws Exception {
		}
		@After
		public void tearDown() throws Exception {
		}
		@Test
		public void testGetValue() {
				assertTrue(formulaBoolConsTest.getValue());
		}
}
```

FormulaFloatConstantTest

```
package de.upb.swt.mcie.formulas;
```

```
import static org.junit.Assert.*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class FormulaFloatConstantTest {

		private FormulaFloatConstant formulaFloatConsTest  = new FormulaFloatConstant((float)
2.5);

		@Before

		public void setUp() throws Exception {

		}

		@After

		public void tearDown() throws Exception {

		}

		@SuppressWarnings("deprecation")

		@Test

		public void testGetValue() {

				assertEquals(2.5,formulaFloatConsTest.getValue(),0.5);

		}

}
```

FormulaInMessageTest

```
package de.upb.swt.mcie.formulas;

import static org.junit.Assert.*;

import junit.framework.TestCase;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class FormulaInMessageTest extends TestCase{

		private    FormulaInMessage    formulaInMessage=new    FormulaInMessage("portname",
"smsname");

		public FormulaInMessageTest(String name){

				super(name);
```

```
        }
        @Before
        public void setUp() throws Exception {
        }
        @After
        public void tearDown() throws Exception {
        }
        @Test
        public void testFormulaInMessage() {
                assertEquals(formulaInMessage.getMessageName() ,"smsname");
        }
        @Test
        public void testGetPortName() {
                assertEquals(formulaInMessage.getPortName(),"portname");
        }
        @Test
        public void testGetMessageName() {
                assertEquals(formulaInMessage.getMessageName() ,"smsname");
        }
        @Test
        public void testisEmpty () {
                assertNotNull(formulaInMessage.getMessageName());
        assertNotNull(formulaInMessage.getPortName() );
        }
        }
```

FormulaOutMessageTest

```
package de.upb.swt.mcie.formulas;
import static org.junit.Assert.*;
import java.util.ArrayList;
import org.junit.After;
import org.junit.Before;

import org.junit.Test;

public class FormulaOutMessageTest {

		private ArrayList<Formula> argTest;

		private FormulaOutMessage formulaOutMessTest = new FormulaOutMessage("port1"
,"InMess1" ,argTest);

		@Before

		public void setUp() throws Exception {

		}

		@After

		public void tearDown() throws Exception {

		}

		@Test

		public void testGetPortName() {

				assertEquals("port1",formulaOutMessTest.getPortName());

		}

		@Test

		public void testGetMessageName() {

				assertEquals("InMess1",formulaOutMessTest.getMessageName());

		}

		@Test

		public void testGetParameters() {

				assertEquals(argTest,formulaOutMessTest.getParameters());

				}

}
```

FunctionOpTest

```
package de.upb.swt.mcie.formulas;

import static org.junit.Assert.*;

import java.util.ArrayList;

import org.junit.After;
import org.junit.Before;

import org.junit.Test;

public class FunctionOpTest {

        private ArrayList <Formula> argsTest;

        private FunctionOp functionOpTest  = new FunctionOp(0 ,argsTest);

        @Before

        public void setUp() throws Exception {

        }

        @After

        public void tearDown() throws Exception {

        }

        @Test

        public void testIsBooleanFormula() {

                boolean test1=functionOpTest.isBooleanFormula();

                boolean test2=functionOpTest.isRationalOperator();

                assertFalse((test1 ||test2));

        }

        @Test

        public void testGetArguments() {

                assertEquals(argsTest,functionOpTest.getArguments());

        }

        @Test

        public void testIsBooleanOperator() {

                boolean test1=functionOpTest.isBooleanOperator();

                assertFalse(test1);

        }

        @Test

        public void testIsRationalOperator() {

                boolean test2=functionOpTest.isRationalOperator();

                assertFalse(test2);

        }
```

```
		@Test
		public void testIsArithmeticOperator() {
				boolean test3=functionOpTest.isArithmeticOperator();
				assertFalse(test3);
		}
		@Test
		public void testIsStringOperator() {
				boolean test4=functionOpTest.isStringOperator();
				assertTrue(test4);
		}
		@Test
		public void testGetName() {
				assertEquals("substring",functionOpTest.getName());
		}
		@Test
		public void testGetRepresentation() {
				assertEquals("substring",functionOpTest.getRepresentation());
		}
}
```

UnaryOpTest

```
package de.upb.swt.mcie.formulas;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
public class UnaryOpTest {
private Formula operand;
		private UnaryOp unaryOpTest  = new UnaryOp(0 ,operand);
		@Before
		public void setUp() throws Exception {
        }


        @After
        public void tearDown() throws Exception {
        }
        @Test
        public void testGetOperand() {
        assertEquals(operand,unaryOpTest.getOperand());
        }
        @Test
        public void testGetName() {
                assertEquals("not",unaryOpTest.getName());
        }
        @Test
        public void testGetRepresentation() {
                assertEquals("!",unaryOpTest.getRepresentation());
                }
        @Test
        public void testIsBooleanFormula() {
                boolean test1=unaryOpTest.isBooleanOperator();
                boolean test2=unaryOpTest.isRationalOperator();
                assertTrue((test1 ||test2));
        }
        @Test
        public void testIsBooleanOperator() {
boolean test1=unaryOpTest.isBooleanOperator();
                assertTrue(test1);
        }
        @Test
        public void testIsRationalOperator() {
                UnaryOp unaryOpTest2  = new UnaryOp(0,operand);
```

```
	boolean test2=unaryOpTest2.isRationalOperator();
				assertFalse(test2);
		}
		@Test
		public void testIsArithmeticOperator() {
				UnaryOp unaryOpTest3  = new UnaryOp(0,operand);
				boolean test3=unaryOpTest3.isArithmeticOperator();
				assertFalse(test3);
		}
}
```

ParserTest

```
package de.upb.swt.mcie.parser;
import static org.junit.Assert.*;
import java.io.IOException;
import java.io.Reader;
import java.io.StringReader;
import java.util.ArrayList;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import de.upb.swt.mcie.formulas.Formula;
import de.upb.swt.mcie.formulas.FormulaAssignment;
import de.upb.swt.mcie.formulas.FormulaBooleanConstant;
import de.upb.swt.mcie.formulas.FormulaInMessage;
import de.upb.swt.mcie.formulas.FormulaIntConstant;
import de.upb.swt.mcie.formulas.FormulaOutMessage;
import de.upb.swt.mcie.parser.token.Brack;
import de.upb.swt.mcie.parser.token.Id;
import de.upb.swt.mcie.parser.token.Sep;
import de.upb.swt.mcie.parser.token.Token;
```

```
public class ParserTest {

        private Formula formula;

        private Formula formula2;

        private ArrayList<Formula> argTest;

        private                    FormulaOutMessage                    outMes=new
FormulaOutMessage("port2","OutMess",argTest);

        @Before

        public void setUp() throws Exception {


        }

        @After

        public void tearDown() throws Exception {

        }

        @Test

        public void testParser() throws IOException {

        @Test

        public void testParseOutMessages() throws IOException, ParseException {

                Token test1,test2,test3,test4,test5,test6;

                String testlabel=" port2.OutMess(a)" ;

                StringReader input = new StringReader(testlabel);

          Parser parser = new Parser((Reader) input);

                Scanner scanner =new Scanner((Reader) input);

//              argTest.add(0,formula);

                assertEquals(outMes.getPortName(),"port2");

                assertEquals(outMes.getMessageName(),"OutMess");

//              assertEquals(1,outMes.getParameters().size());

                assertEquals(argTest,outMes.getParameters());

                test1= scanner.getNextToken();

                assertEquals("port2",((Id) test1).getName());
                test2=scanner.getNextToken();
               assertEquals(3,((Sep) test2).getType());

                test3= scanner.getNextToken();
                assertEquals("OutMess",((Id) test3).getName());

                test4= scanner.getNextToken();
                assertEquals(0,((Brack) test4).getType());

                test5= scanner.getNextToken();
                assertEquals("a",((Id) test5).getName());

                test6= scanner.getNextToken();
                assertEquals(1,((Brack) test6).getType());
}
@Test
public void testParseAssignment() {
                FormulaIntConstant formula3 =new FormulaIntConstant(4);
                FormulaAssignment Fassign=new FormulaAssignment("att",formula3);
                assertEquals("att",Fassign.getAttr());
}
@Test
public void testParseInMessageTry() throws IOException {
                Token test1;
                String testlabel=" :" ;
                StringReader input = new StringReader(testlabel);
   Parser parser = new Parser((Reader) input);
                Scanner scanner =new Scanner((Reader) input);
                FormulaInMessage atom=new FormulaInMessage("port1","InMes");
                assertEquals(atom.getPortName(),"port1");
                assertEquals(atom.getMessageName(),"InMes");
                test1= scanner.getNextToken();
                assertEquals(2,((Sep) test1).getType());
        }
        @Test
        public void testParseTransitionLabel() throws IOException, ParseException {
                String testlabel="a>1::;a=0" ;
                StringReader input = new StringReader(testlabel);
                Parser parser = new Parser((Reader) input);
                assertNotNull(parser.parseTransitionLabel());
        }
}
```

ScannerTest

```
package de.upb.swt.mcie.parser;
import static org.junit.Assert.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.StringReader;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import de.upb.swt.mcie.formulas.FormulaInMessage;
import de.upb.swt.mcie.parser.token.Token;
public class ScannerTest {
        @Before
        public void setUp() throws Exception {
        }
        @After
        public void tearDown() throws Exception {
        }
        @Test
        public void testScanner() throws IOException {
        }
        @Test
        public void testGetNextToken() throws IOException {
                String testlabel="a<1;port1:port2:a=3";
                StringReader input = new StringReader(testlabel);
                Parser parser = new Parser((Reader) input);
                Scanner scanner =new Scanner((Reader) input);
                assertNotNull(scanner.getNextToken());
        }
}
```