Enhancing Testing of VDM-SL models

Peter W. V. Tran-Jørgensen\textsuperscript{1} René S. Nilsson\textsuperscript{1,2} Kenneth Lausdahl\textsuperscript{3}

16th Overture workshop
Oxford University, UK – July 14, 2018
Agenda

Problem/motivation

Unit/integration testing in VDM

VDMUnit and code-generation extensions

Conclusion and future plans
Agenda

Problem/motivation

Unit/integration testing in VDM

VDMUnit and code-generation extensions

Conclusion and future plans
Background: unit and integration testing in VDM

- Testing SL models
  - Tedious and error-prone
  - Lack of tool support
- VDM++/VDM-RT is supported by VDMUnit
- VDMUnit does not work for VDM-SL
  - Relies on features not available in VDM-SL
    - Object-orientation and exception handling
Background: unit and integration testing in VDM

- Testing SL models
  - Tedious and error-prone
  - Lack of tool support
- VDM++/VDM-RT is supported by VDMUnit
  - VDMUnit does not work for VDM-SL
    - Relies on features not available in VDM-SL
      - Object-orientation and exception handling
Background: unit and integration testing in VDM

- Testing SL models
  - Tedious and error-prone
  - Lack of tool support
- VDM++/VDM-RT is supported by VDMUnit
- VDMUnit does not work for VDM-SL
  - Relies on features not available in VDM-SL
    - Object-orientation and exception handling
Objectives

- Improve unit/integration testing in VDM-SL
  - by extending VDMUnit
- Reuse model tests to validate model realisation
  - by extending Overture’s VDM-to-Java code-generator
- Improve continuous integration in a VDM setting
  - by generating VDM test reports for Jenkins
Objectives

- Improve unit/integration testing in VDM-SL
  - by extending VDMUnit
- Reuse model tests to validate model realisation
  - by extending Overture’s VDM-to-Java code-generator
- Improve continuous integration in a VDM setting
  - by generating VDM test reports for Jenkins
Objectives

- Improve unit/integration testing in VDM-SL
  - by extending VDMUnit
- Reuse model tests to validate model realisation
  - by extending Overture’s VDM-to-Java code-generator
- Improve continuous integration in a VDM setting
  - by generating VDM test reports for Jenkins
Agenda

Problem/motivation

Unit/integration testing in VDM

VDMUnit and code-generation extensions

Conclusion and future plans
VDMUnit architecture

- Consists of a VDM and a Java component
  - VDM: exposes VDMUnit features
  - Java: identifies/executes tests using Java reflection
- VDM and Java connected via Overture’s Java bridge
VDMUnit architecture

- Consists of a *VDM* and a *Java* component
  - VDM: exposes VDMUnit features
  - Java: identifies/executes tests using Java reflection
- VDM and Java connected via Overture’s *Java bridge*
VDMUnit tests in VDM++

```vdm
class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
testOne () == Assert'assertTrue("Expected 'someFeature' to generate " ^ 
   "an even number ", someFeature() mod 2 = 0);
end MyTest
```

- Test classes extend **TestCase**
  - Test operations begin with “test”
  - Features validated using assertions
- `setUp` and `tearDown` invoked before/after each test
- Tests execution: all vs. selectively
  - All: `new` TestRunner.run()
- **OO features not available in VDM-SL**
VDMUnit tests in VDM++

```vdm++
class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
  testOne () == Assert'assertTrue("Expected 'someFeature' to generate " ^
     "an even number ", someFeature() mod 2 = 0);
end MyTest
```

- **Test classes extend** `TestCase`
  - Test operations begin with “test”
  - Features validated using assertions
- `setUp` and `tearDown` invoked before/after each test
- Tests execution: all vs. selectively
  - **All**: `new TestRunner.run()`
- **OO features not available in VDM-SL**
VDMUnit tests in VDM++

```vdm
class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
end
```

testOne () == Assert.'assertTrue("Expected 'someFeature' to generate " ^
    "an even number ", someFeature() mod 2 = 0);

class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
end MyTest
```

• Test classes extend **TestCase**
  • Test operations begin with “test”
  • Features validated using assertions

• **setUp** and **tearDown** invoked before/after each test
• Tests execution: all vs. selectively
  • All: new TestRunner.run()

• **OO features not available in VDM-SL**
class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
    testOne () == Assert‘assertTrue("Expected ‘someFeature’ to generate " ^
          "an even number ", someFeature() mod 2 = 0);
end MyTest

- Test classes extendTestCase
  - Test operations begin with “test”
  - Features validated using assertions
- setUp and tearDown invoked before/after each test
- Tests execution: all vs. selectively
  - All: new TestRunner.run()
- OO features not available in VDM-SL
VDMUnit tests in VDM++

class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
testOne () == Assert`assertTrue("Expected 'someFeature' to generate " ^
    "an even number ", someFeature() mod 2 = 0);
end MyTest

• Test classes extend TestCase
  • Test operations begin with “test”
  • Features validated using assertions
• setUp and tearDown invoked before/after each test
• Tests execution: all vs. selectively
  • All: new TestRunner.run()
• OO features not available in VDM-SL
VDMUnit tests in VDM++

```
class MyTest is subclass of TestCase
operations
public testOne : () ==> ()
  testOne () == Assert`assertTrue("Expected ‘someFeature’ to generate " ^
    "an even number ", someFeature() mod 2 = 0);
end MyTest
```

- Test classes extend `TestCase`
  - Test operations begin with “test”
  - Features validated using assertions
- `setUp` and `tearDown` invoked before/after each test
- Tests execution: all vs. selectively
  - All: `new TestRunner.run()`
- **OO features not available in VDM-SL**
Agenda

Problem/motivation

Unit/integration testing in VDM

VDMUnit and code-generation extensions

Conclusion and future plans
VDM-SL modules expose VDMUnit features

```vdm
module Assert
imports from TestRunner all
exports all

definitions

operations

assertTrue: bool ==> ()
assertTrue (pbool) ==
  if not pbool then
    TestRunner`markFail();

assertTrueMsg: seq of char * bool ==> ()
assertTrueMsg (pmessage, pbool) ==
  if not pbool then (  
    TestRunner`setMsg(pmessage);  
    TestRunner`markFail();  
  );  

...other assertion omitted

end Assert
```
VDM-SL modules expose VDMUnit features

```vdm
module Assert
imports from TestRunner all
exports all

definitions

operations

assertTrue: bool ==> ()
assertTrue (pbool) ==
  if not pbool then
    TestRunner\'markFail();

assertTrueMsg: seq of char * bool ==> ()
assertTrueMsg (pmessage, pbool) ==
  if not pbool then (  
    TestRunner\'setMsg(pmessage);  
    TestRunner\'markFail();  
  );

...other assertion omitted

end Assert
```
VDM-SL modules expose VDMUnit features

```vdm
module Assert
imports from TestRunner all
exports all

definitions

operations

assertTrue: bool ==> ()
assertTrue (pbool) ==
  if not pbool then
    TestRunner'markFail();

assertTrueMsg: seq of char * bool ==> ()
assertTrueMsg (pmessage, pbool) ==
  if not pbool then (  
    TestRunner'setMsg(pmessage);
    TestRunner'markFail();
  );

...other assertion omitted

end Assert
```
VDM-SL modules expose VDMUnit features

```vdm
module Assert
imports from TestRunner all
exports all

definitions

operations

assertTrue: bool ==> ()
assertTrue (pbool) ==
  if not pbool then
    TestRunner`markFail();

assertTrueMsg: seq of char * bool ==> ()
assertTrueMsg (pmessage, pbool) ==
  if not pbool then
    TestRunner`setMsg(pmessage);
    TestRunner`markFail();

...other assertion omitted

end Assert
```

```vdm
module TestRunner
exports all

definitions

operations

run : ()==>()
run() == is not yet specified;

markFail : () ==> ()
markFail () == is not yet specified;

setMsg : seq of char ==> ()
setMsg (msg) == is not yet specified;

end TestRunner
```
module Assert
imports from TestRunner all
exports all

definitions

operations

assertTrue: bool ==> ()
assertTrue (pbool) ==
    if not pbool then
        TestRunner‘markFail();

assertTrueMsg: seq of char * bool ==> ()
assertTrueMsg (pmessage, pbool) ==
    if not pbool then
        TestRunner‘setMsg(pmessage);
        TestRunner‘markFail();
    );

...other assertion omitted

end Assert

module TestRunner
exports all

definitions

operations

run : ()==>()
run() == is not yet specified;

markFail : () ==> ()
markFail () == is not yet specified;

setMsg : seq of char ==> ()
setMsg (msg) == is not yet specified;

end TestRunner
module MyTest
...
state St of
  x : int
end;
operations
setUp : () ==> ()
setUp () == initState();

tearDown : () ==> ()
tearDown () == cleanUp();

testOdd: ()==>() 
testOdd() == ( 
  x := x + 1;
  Assert 'assertFalseMsg("Expected x " ^
   "to be odd", x mod 2 = 0); );

testInverse: ()==>() 
testInverse() == ...

testPos: ()==>() 
testPos() == ...
...
end MyTest
Java/JUnit4 translation

VDM-SL test example

```vdm
module MyTest
...
state St of
  x : int
end;
operations
setUp : () ==> ()
setUp () == initState();
tearDown : () ==> ()
tearDown () == cleanUp();
testOdd: ()==>()
testOdd()== (
  x := x + 1;
  Assert'assertFalseMsg("Expected x " ^ "to be odd", x mod 2 = 0); );
testInverse: ()==>()
testInverse()== 

testPos: ()==>()
testPos()== ...
... end MyTest
```

Java translation

```java
final public class MyTest {

  private static St St = new St(null);

  @Before
  public void setUp() { initState(); }

  @After
  public void tearDown() { cleanUp(); }

  @Test
  public void testOdd() {
    St.x = St.x.longValue() + 1L;
    Assert.assertFalse("Expected x " + "to be odd", Utils.equals(Utils.
      mod(St.x.longValue(), 2L),0L));
  }

  @Test
  public void testInverse() { ...
  ...

  @Test
  public void testPos() { ...
  ...
```

16th Overture Workshop, July 14, 2018

Peter W. V. Tran-Jørgensen, René S. Nilsson, Kenneth Lausdahl
Java/JUnit4 translation

VDM-SL test example

```vdm
module MyTest
...
state St of
  x : int
end;
operations
setUp : () ==> ()
setUp () == initState();
tearDown : () ==> ()
tearDown () == cleanUp();
testOdd: ()==>()  
testOdd() == (  
  x := x + 1;
  Assert'assertFalseMsg("Expected x " ^  
  "to be odd", x mod 2 = 0); );
testInverse: ()==>()  
testInverse() == ...
testPos: ()==>()  
testPos() == ...
...
end MyTest
```

Java translation

```java
final public class MyTest
{
  private static St St = new St(null);
  
  @Before
  public void setUp() { initState(); }
  
  @After
  public void tearDown() { cleanUp(); }
  
  @Test
  public void testOdd() {
    St.x = St.x.longValue() + 1L;
    Assert.assertFalse("Expected x " +  
    "to be odd", Utils.equals(Utils.
    mod(St.x.longValue(), 2L),0L));
  }
  
  @Test
  public void testInverse() { ...
  }  
  
  @Test
  public void testPos() { ...
  }  
  
  ...
}
```
Java/JUnit4 translation

**VDM-SL test example**

```vdm
class MyTest
  state St of
    x : int
  end;
operations
  setUp : () ==> ()
  setUp () == initState();
end;
operations
  tearDown : () ==> ()
  tearDown () == cleanUp();
end;
operations
  testOdd: ()==>()
  testOdd() == (x := x + 1;
    Assert.'assertFalseMsg("Expected x " ^ 
    "to be odd", x mod 2 = 0); );
end;
operations
  testInverse: ()==>()
  testInverse() == ...
end;
operations
  testPos: ()==>()
  testPos() == ...
end;
```

**Java translation**

```java
import static org.junit.Assert.assertFalse;
import org.junit.Before;
import org.junit.After;
import org.junit.Test;

public class MyTest {
  private static St St = new St(null);

  @Before
  public void setUp() { initState(); }

  @After
  public void tearDown() { cleanUp(); }

  @Test
  public void testOdd() {
    St.x = St.x.longValue() + 1L;
    assertFalse("Expected x " + 
    "to be odd", Utils.equals(Utils.
    mod(St.x.longValue(), 2L),0L));
  }

  @Test
  public void testInverse() { ... }

  @Test
  public void testPos() { ... }
}
```

---

**Problem/motivation**

Unit/integration testing in VDM VDMUnit and code-generation extensions

**Conclusion and future plans**

16th Overture Workshop, July 14, 2018 Peter W. V. Tran-Jørgensen, René S. Nilsson, Kenneth Lausdahl
Java/JUnit4 translation

VDM-SL test example

module MyTest
...
state St of
  x : int
end;
operations
setUp : () ==> ()
setUp () == initState();
tearDown : () ==> ()
tearDown () == cleanUp();
testOdd: ()==>() 
testOdd()== ( 
  x := x + 1;
  Assert assertFalseMsg("Expected x ", x mod 2 = 0); ); 

testInverse: ()==>() 
testInverse()== ... 

testPos: ()==>() 
testPos()== ... 
...
end MyTest

Java translation

final public class MyTest
{
  private static St St = new St(null);

  @Before
  public void setUp() { initState(); }

  @After
  public void tearDown() { cleanUp(); }

  @Test
  public void testOdd() {
    St.x = St.x.longValue() + 1L;
    Assert assertFalse("Expected x " ^ 
    "to be odd", x mod 2 = 0); }

  @Test
  public void testInverse() { ... }

  @Test
  public void testPos() { ... }

  ...
}
Harvest planning case study

- Master algorithm in VDM
  - Code-generated to Java
- 4200 lines of VDM
  - ≈ 3100 lines of model
  - ≈ 1100 lines of tests
- 134 VDM-SL tests
  - VDM tests: ≈ 7 hours
  - Java versions: ≈ 30 min.
Harvest planning case study

- Master algorithm in VDM
  - Code-generated to Java
- 4200 lines of VDM
  - \( \approx \) 3100 lines of model
  - \( \approx \) 1100 lines of tests
- 134 VDM-SL tests
  - VDM tests: \( \approx \) 7 hours
  - Java versions: \( \approx \) 30 min.
Harvest planning case study

- Master algorithm in VDM
  - Code-generated to Java
- 4200 lines of VDM
  - \( \approx 3100 \) lines of model
  - \( \approx 1100 \) lines of tests
- 134 VDM-SL tests
  - VDM tests: \( \approx 7 \) hours
  - Java versions: \( \approx 30 \) min.
Harvest planning case study

- Master algorithm in VDM
  - Code-generated to Java
- 4200 lines of VDM
  - $\approx 3100$ lines of model
  - $\approx 1100$ lines of tests
- 134 VDM-SL tests
  - VDM tests: $\approx 7$ hours
  - Java versions: $\approx 30$ min.
Jenkins integration

- Overture CLI extension
  - Pass property
    - `Dvdm.unit.report`
- VDM test reports
- Generate XML reports
- Jenkins visualisation

### Test Result: (root)

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Duration</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoulumTest</td>
<td>1.2 sec</td>
<td>1</td>
</tr>
<tr>
<td>FoulumTest</td>
<td>1.8 sec</td>
<td>1</td>
</tr>
<tr>
<td>HobroLandevejTest</td>
<td>29 sec</td>
<td>1</td>
</tr>
<tr>
<td>HobroLandevejTest</td>
<td>38 sec</td>
<td>1</td>
</tr>
<tr>
<td>HobroLandevejTest</td>
<td>30 sec</td>
<td>1</td>
</tr>
</tbody>
</table>

134 tests (+0)
Took 2 hr 25 min.
Jenkins integration

- Overture CLI extension
  - Pass property
    - `Dvdm.unit.report`
- VDM test reports
  - Generate XML reports
  - Jenkins visualisation

![Test Result: (root)](image)

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Duration</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FoulumTest.test_field_test_1_Headland_Bee</code></td>
<td>1.2 sec</td>
<td>1</td>
</tr>
<tr>
<td><code>FoulumTest.test_field_test_1_OnTheGo_Bee</code></td>
<td>1.8 sec</td>
<td>1</td>
</tr>
<tr>
<td><code>HobroLandevejTest.BCO_HeadlandUnload</code></td>
<td>29 sec</td>
<td>1</td>
</tr>
<tr>
<td><code>HobroLandevejTest.BCO_OnTheGo</code></td>
<td>38 sec</td>
<td>1</td>
</tr>
<tr>
<td><code>HobroLandevejTest.BCO_SP</code></td>
<td>30 sec</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Duration</th>
<th>Fail (diff)</th>
<th>Skip (diff)</th>
<th>Pass (diff)</th>
<th>Total (diff)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BeeTest</code></td>
<td>7.3 sec</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><code>BridgeTest</code></td>
<td>0.11 sec</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><code>FieldTest</code></td>
<td>2.2 sec</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><code>FoulumTest</code></td>
<td>1 min 34 sec</td>
<td>2 (+2)</td>
<td>0</td>
<td>18 (-2)</td>
<td>20</td>
</tr>
<tr>
<td><code>HeadlandUnloadTest</code></td>
<td>20 sec</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td><code>HobroLandevejTest</code></td>
<td>2 min 54 sec</td>
<td>3 (+3)</td>
<td>0</td>
<td>3 (-3)</td>
<td>6</td>
</tr>
<tr>
<td><code>LoggerTest</code></td>
<td>1 ms</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Agenda

Problem/motivation

Unit/integration testing in VDM

VDMUnit and code-generation extensions

Conclusion and future plans
Conclusion and future plans

- **VDMUnit for VDM-SL**
  - Features exposed using VDM-SL modules
  - Java-component implements test execution
- Translation of VDM-SL tests to JUnit4 tests
  - Model tests used to validate model realisation
  - Jenkins integration
- **Future plans**
  - Propose library extension to the Language Board
  - Integrate feature in Overture releases
Conclusion and future plans

- VDMUnit for VDM-SL
  - Features exposed using VDM-SL modules
  - Java-component implements test execution
- Translation of VDM-SL tests to JUnit4 tests
  - Model tests used to validate model realisation
  - Jenkins integration

- Future plans
  - Propose library extension to the Language Board
  - Integrate feature in Overture releases
Conclusion and future plans

- **VDMUnit for VDM-SL**
  - Features exposed using VDM-SL modules
  - Java-component implements test execution
- **Translation of VDM-SL tests to JUnit4 tests**
  - Model tests used to validate model realisation
  - Jenkins integration
- **Future plans**
  - Propose library extension to the Language Board
  - Integrate feature in Overture releases
Thank you

VDMUnit for VDM-SL:
https://github.com/overturetool/overture/pull/671