Enhancing Testing of VDM-SL models

Peter W. V. Tran-Jørgensen[1] René S. Nilsson[1,2]
Kenneth Lausdahl[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

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

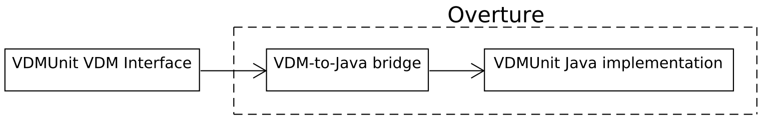
- Improve unit/integration testing in VDM-SL
  - by extending VDMUnit
- Reuse model tests to validate model realisation
  - by extending Overture's VDM-to-Java code-generator
- Improve continuous integration in a VDM setting
  - by generating VDM test reports for Jenkins

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

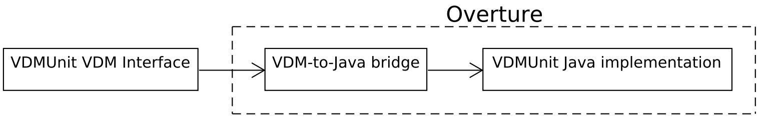
Agenda

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++

```
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 valided 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 valided 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 valided 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 valided 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 valided 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 valided 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

VDM-SL modules expose VDMUnit features

```
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

```
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

```
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

```
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
```

VDM-SL modules expose VDMUnit features

```
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
```

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 " ^
    "to be odd", x mod 2 = 0); );

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

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

VDM-SL test example

Java/JUnit4 translation

```
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
```

VDM-SL test example

```
...
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 translation

Java/JUnit4 translation

```
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
```

VDM-SL test example

```
...
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 translation

Java/JUnit4 translation

```
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
```

VDM-SL test example

```
...
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 translation

Java/JUnit4 translation

```
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
```

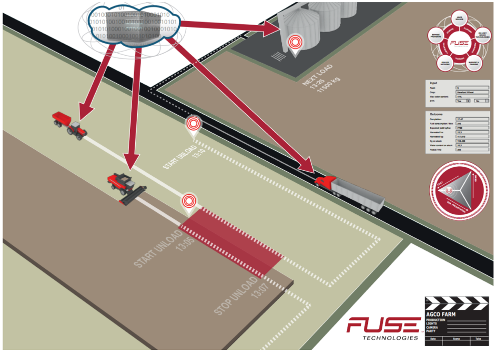
VDM-SL test example

```
...
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 translation

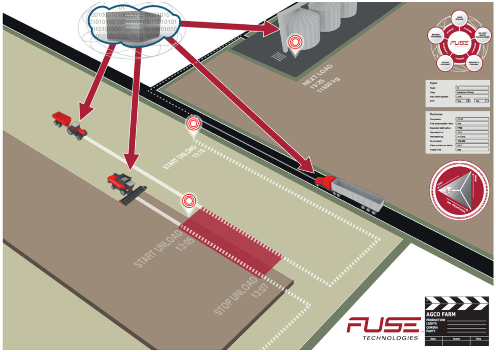
Harvest planning case study

- 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.



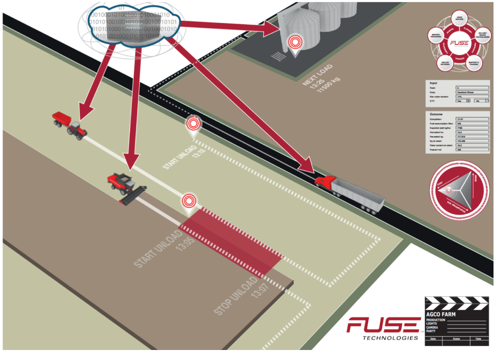
Harvest planning case study

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



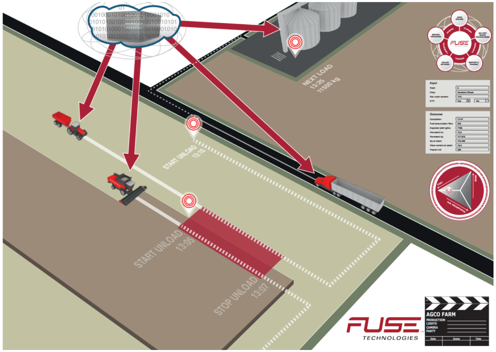
Harvest planning case study

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



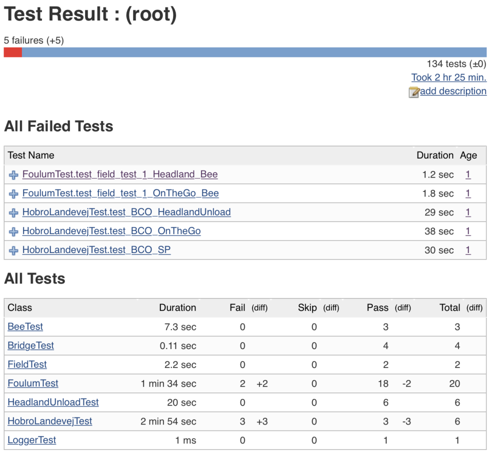
Harvest planning case study

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



Jenkins integration

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



Test Result : (root)

5 failures (+5)

134 tests (±0)
Took 2 hr 25 min.
add description

All Failed Tests

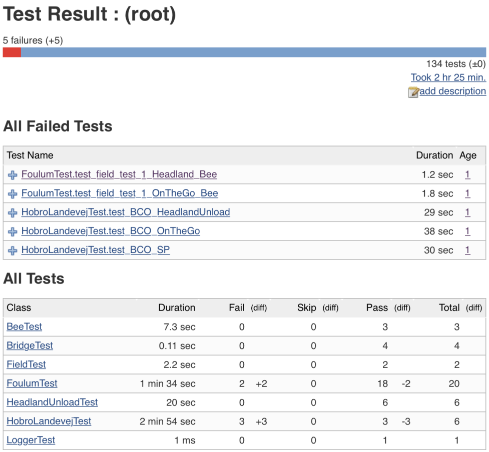
Test Name	Duration	Age
FoulumTest.test_field_test_1_Headland_Bee	1.2 sec	1
FoulumTest.test_field_test_1_OnTheGo_Bee	1.8 sec	1
HobroLandevejTest.test_BCO_HeadlandUnload	29 sec	1
HobroLandevejTest.test_BCO_OnTheGo	38 sec	1
HobroLandevejTest.test_BCO_SP	30 sec	1

All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
BeeTest	7.3 sec	0		0		3		3	
BridgeTest	0.11 sec	0		0		4		4	
FieldTest	2.2 sec	0		0		2		2	
FoulumTest	1 min 34 sec	2	+2	0		18	-2	20	
HeadlandUnloadTest	20 sec	0		0		6		6	
HobroLandevejTest	2 min 54 sec	3	+3	0		3	-3	6	
LoggerTest	1 ms	0		0		1		1	

Jenkins integration

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



Test Result : (root)

5 failures (+5)

134 tests (±0)
Took 2 hr 25 min.
add description

All Failed Tests

Test Name	Duration	Age
FoulumTest.test_field_test_1_Headland_Bee	1.2 sec	1
FoulumTest.test_field_test_1_OnTheGo_Bee	1.8 sec	1
HobroLandevejTest.test_BCO_HeadlandUnload	29 sec	1
HobroLandevejTest.test_BCO_OnTheGo	38 sec	1
HobroLandevejTest.test_BCO_SP	30 sec	1

All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
BeeTest	7.3 sec	0		0		3		3	
BridgeTest	0.11 sec	0		0		4		4	
FieldTest	2.2 sec	0		0		2		2	
FoulumTest	1 min 34 sec	2	+2	0		18	-2	20	
HeadlandUnloadTest	20 sec	0		0		6		6	
HobroLandevejTest	2 min 54 sec	3	+3	0		3	-3	6	
LoggerTest	1 ms	0		0		1		1	

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`