EXPECTATIONS MATTER.

WE’RE COMMITTED TO EXCEED YOURS

USING FORMAL SPECIFICATIONS TO SUPPORT MODEL BASED TESTING

ASDSPEC: A TOOL COMBINING THE BEST OF TWO TECHNIQUES

Rachid Kherrazi
6-4-2014 Grenoble
PEOPLE INVOLVED

• Rachid Kherrazi
  – Senior Consultant @ Nspyre
  – Domain
    • Process and Product Improvement
    • RAMS (Reliability Availability Maintenance and Safety)
    • Model Based Testing and Model Driven Engineering

• Also : Arjan van Der Meer & Marc Hamilton
ABOUT NSPYRE

MARKET SEGMENTS

• High Tech
• Traffic & Infra
• Industry

AREA’S OF EXCELLENCE

(Model Based) Systems Engineering / Model Driven Engineering / Model Based Testing / Industrial Automation / Simulation / Big Data / Mobile Solutions
CONTENTS

1) MDSD Introduction
2) MDE with ASD:Suite
3) MBT with MS Spec Explorer
4) ASDSpec
V-MODEL AND MODEL DRIVEN SYSTEM DEVELOPMENT

Model Driven System Development

- Requirements Analysis
- System Functional Analysis
- Design Synthesis
- SW Analysis & Design
- SW Implementation & Unit Test
- Module Integration & Test
- (Sub-)System Integration & Test
- Model/Requirements Repository
- System Acceptance
V-MODEL AND MODEL DRIVEN SYSTEM DEVELOPMENT

Some of related technology

- Requirements Analysis
- System Functional Analysis
- Design Synthesis
- SW Analysis & Design
- SW Implementation & Unit Test
- Module Integration & Test
- (Sub-)System Integration & Test
- System Acceptance

Model Driven Engineering

ASD

emf

NDSPYRE
V-MODEL FOR TESTING

3 main steps in test process

System

Subsystem

Module

Requirements

Design

Coding

Integration

System test

Test case specification
(design of logical test case)
(reduction of number of test cases by application of test techniques)

Test case generation
(design of physical test case)
(selection of input values and calculation of expected results)

Test execution
(reporting)
MBT IS THE AUTOMATION OF TEST CASE GENERATION

<table>
<thead>
<tr>
<th>Manual</th>
<th>Automatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test specification</td>
<td></td>
</tr>
<tr>
<td>Test generation</td>
<td></td>
</tr>
<tr>
<td>Test execution</td>
<td></td>
</tr>
</tbody>
</table>

- **Manual**
  - Test specification
  - Test generation
  - Test execution
  - Testing skills

- **Automatic**
  - Manual modeling
  - Manual scripting
  - Model-based testing
  - + Modeling skills
  - + Scripting skills

**Testing skills**

**Nineth Workshop on Model-Based Testing**

April 6, 2014

ETAPS 2014
PERCEIVED BENEFITS

• Increased productivity (increased automation)
• Better test script maintenance
• Improved product reliability (new type of bugs, Increased test coverage)
• Agility (Easily react to new feature changes, Reusability of test semantics, Early test engagement, Drive quality upstream)
• Increased employee satisfaction (challenging, new horizon, fun)

• Question: How can we improve MBT e.g. increase further the productivity?
MBT IS THE AUTOMATION OF TEST CASE GENERATION

<table>
<thead>
<tr>
<th>Manual</th>
<th>Automatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test specification</td>
<td>Test specification</td>
</tr>
<tr>
<td>Manual testing</td>
<td>Traditional testing</td>
</tr>
<tr>
<td>Model-based testing</td>
<td>Manual Modeling</td>
</tr>
<tr>
<td>Scripting skills</td>
<td>+ Scripting skills</td>
</tr>
<tr>
<td>Testing skills</td>
<td>Modeling skills</td>
</tr>
<tr>
<td>MBT next</td>
<td></td>
</tr>
</tbody>
</table>
IDEA: USING BEST OF BOTH WORLDS

• Automatic generation of (partial) Spec Explorer test model from existing ASD interface model

Benefits
• Less effort for model creation
  – reuse of existing work
• Testing of complete system including
  – Legacy code
  – External components
  – Data combination testing
  – Interaction testing
• Results: High Quality, Reduced cost
Multiple components
Components need to interact to function
Controller needed to coordinate interaction
<table>
<thead>
<tr>
<th>Stage</th>
<th>Use case /SUT</th>
<th>Goal</th>
<th>Technique</th>
<th>Tool</th>
<th>Method</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td>Container Terminal (CT)</td>
<td>Development of control software for the CT</td>
<td>Model Driven Engineering + hand written sw</td>
<td>ASD:Suite</td>
<td>Create design Models + interface models in ASD Generate code</td>
<td>Productivity (code generation) c.t. trad. dev Testing complete system, Interaction, data, external code</td>
</tr>
<tr>
<td>Stage 2</td>
<td>Container Terminal (CT)</td>
<td>Verification of the developed control software of the CT (generated + Hand written)</td>
<td>Model based Testing</td>
<td>Spec Explorer</td>
<td>Create Test Model in Spec Explorer, generate test suite</td>
<td>Test Productivity Modeling skills, complexity costs</td>
</tr>
<tr>
<td>Stage 3</td>
<td>Container Terminal (CT)</td>
<td>Verification of the developed control software of the CT (generated + Hand written)</td>
<td>Model based Testing</td>
<td>ASDspec &amp; Spec Explorer</td>
<td>Generate Test Model from existing ASD interface Model, complete Test Model, generate test suite</td>
<td>Productivity (code generation, partial test generation) Testing complete system, Interaction, data, external code (benefits only in case of existing of ASD models)</td>
</tr>
</tbody>
</table>
CONTENTS

1) MDSD Introduction
2) MDE with ASD:Suite
3) MBT with MS Spec Explorer
4) ASDSpec
ASD (ANALYTICAL SOFTWARE DESIGN)

- Model Driven Engineering (code generation)
- Component Based Development
- Models are verified mathematically at design time (formal methods)
COMPONENT BASED DEVELOPMENT ASD-STYLE

For each component a interface mode and design model are created in ASD.
ASD: WORKFLOW

• Designer defines behavior in component models
• ASD:suite verifies models using model checking
• ASD:suite generates implementation code

But some shortcoming
• No support for Legacy/External code
• No support for indirect component interaction
• Limited data interaction

⇒ Testing is needed.
RESULTS:
ASD-CONTAINER TERMINAL

Higher overall productivity compared to traditional development.

Main remaining problems:
- Testing interaction and complete system.
- Debugging of third party library (some bugs found in manual written code, legacy code)
CONTENTS

1) MDSD Introduction
2) MDE with ASD:Suite
3) MBT with MS Spec Explorer
4) ASDSpec
MBT WITH SPEC EXPLORER

C# Model
(or other .Net Language)

Explore & Analyze

Remodel

Generate

Test Suite

Model Graph

Execute

http://www.nunit.com/

Nunit

Nineth Workshop on Model-Based Testing
April 6, 2014
ETAPS 2014
MODELING IN SPEC EXPLORER

```csharp
[TypeBinding("CraneComponent")]

class Crane
{
    public Cranestates Cranestatevar;
    internal ICrane_NIModel ICrane_NIimpl;
    internal ICrane ICraneimpl;

    [Rule(Action = "new CraneComponent(craneNr)",
          ModeTransition = "crane->crane0")]

    Crane(int craneNr)
    {
        ICraneimpl = new ICraneCranecomponent(this);
        craneList.Add(this);
    }
}
```

Spec Explorer represents models as annotated C# code

State declaration

Used components

Initialization
MODELING IN SPEC EXPLORER

public void ICrane_PlaceContainer(GripperHeightEnum height) {
    switch (basecomponent.Cranestatevar) {
        case Cranestates.Positioned: {
            basecomponent.Cranestatevar = Cranestates.PlacingContainer;
            basecomponent.ICrane_NImple.PlaceContainer(height);
            return;
        }
        default: {
            Condition.IsTrue(false);
            throw new InvalidOperationException();
        }
    }
}

Every model method represents an action, its conditions and its effects.
Completed models can be explored to discover all possible behaviour
Explored model contains all possible action sequences
Spec Explorer creates test sequences to cover explored behaviour.
Nunit test scripts are used to implement and execute tests.
RESULTS: SPEC EXPLORER-CONTAINER TERMINAL

Significantly less effort than traditional automated testing
• Model Based Testing + Software Analysis
• Support of data combination testing
• Support of Model composition, incremental

However some remaining problems:
• Modeling still comparatively expensive
• Modeling effort, complexity and skills (experienced tester needed)
HOW TO COMBINE BOTH BENEFITS

ASD generates verified software components

Complications
- Legacy code
- External code
- Component interaction
- Limited support of data

Spec Explorer generates automated software tests

Complications
- Modeling needed
- MBT skills needed

Diagram:
- Client
- Test model
- Test script
- A
- B
- C

Ninth Workshop on Model-Based Testing
April 6, 2014
ETAPS 2014
1) MDSD Introduction
2) MDE with ASD:Suite
3) MBT with MS Spec Explorer
4) ASDSpec
OUR SOLUTION: ASDSPEC

ASDSpec generates Test Model (push button) from existing ASD interface models
Tester need to refine manually the generated model (add behavior, data, slicing,...)
COMBINING THE BEST OF BOTH WORLDS

Automatic generation of partial Spec Explorer test model from existing ASD model (push button)

Benefits

• Effortless model creation
  – reuse of existing work

• Testing of complete system including
  – Legacy code
  – External components
  – Data combination testing
  – Interaction testing

Results:

– High Quality, Reduced cost
– Adds dynamic testing to ASD
Reuse existing ASD interface models to generate Spec Explorer MBT models automatically.
ASDSPEC: DATA AND MODEL COMPOSITION

Generated code can be extended with any Spec Explorer feature

- **ASD Interface Model**
- **ASDspec**
- **Spec Explorer Model**
- **Spec Explorer**
- **Test Cases**

- Add model composition
- Add data interaction
ASDSpec makes use of existing MDSD architecture
ASDSPEC: TECHNOLOGIES

Model transformations bridge gap between ASD and Spec Explorer

ASDSpec

ASD XML → QVTO → ASD EMF → QVTO → SE EMF → Acceleo → SE model

QVTO

ASD EMF

SE EMF

Cord EMF

Acceleo

Cord script
RESULTS: ASDSPEC - CONTAINER TERMINAL

- Low effort model creation
- Easy reuse of existing work
- Testing of complete system
- Legacy code, data interaction
- High Quality, Cost Reduction

<table>
<thead>
<tr>
<th></th>
<th>Spec Explorer</th>
<th>ASD + ASDSpec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td>Generate test suite</td>
<td>Generate test model</td>
</tr>
<tr>
<td>Techniques</td>
<td>MBT</td>
<td>MBT</td>
</tr>
<tr>
<td>Effort/complexity</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Test cases</td>
<td>89</td>
<td>93</td>
</tr>
<tr>
<td>Perceived eff.</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Bugs</td>
<td>Some bugs in hand written code /HAL</td>
<td>All known bus found</td>
</tr>
</tbody>
</table>
TRADITIONAL SW DEVELOPMENT

TRADITIONAL AUTOMATED TESTING

Design spec
Interface spec
Detail design
External code
Hand written code
Legacy code
Manual scripted Test suite
Integration testing
System testing
Unit testing
Test execution Tool

Summary
ASD BASED SW DEVELOPMENT
TRADITIONAL AUTOMATED TESTING

Design spec
Interface spec

Interface model
Design model

Hand written code
External code

Manual scripted Test suite

Generated code
Legacy code

System testing
Integration testing
Unit testing

Test execution Tool

Summary
ASD BASED SW DEVELOPMENT
MODEL BASED TESTING+ASDSPEC

Design spec
Interface spec

Generated Test model
Manual scripted Test suite
Test execution Tool

System testing
Integration testing
Unit testing

Interface model
Design model

ASDspec
Config, data

Test suite
Test execution Tool

MBT tool

Generated code
Legacy code

Hand written code
External code

Integration testing
System testing
TOOL STATUS

• Tool currently in prototype phase
• as an Eclipse plugin
• For now available on request

• Future steps
  – Support data handling/configuration aspects in ASDSpec, instead of relying on manual additions in Spec Explorer,
  – Supporting other MBT tools.
CONCLUSIONS

- MDE and MBT technologies have matured a lot in the latest years
- It is a matter of time..... Evolution.....
Using Formal Specifications to Support Model Based Testing

ASDSpec: A Tool Combining the Best of Two Techniques