POSSIBILITY OF KEYWORD-BASED MODEL-DRIVEN FUNCTIONAL SYSTEM TEST GENERATION

Jessica Schiffmann
Speaker – Jessica Schiffmann

**Education**

Since 2011
extra-occupational Phd-thesis, TU Freiberg and FH Kaiserslautern

2005
Diploma Computer Science (FH)

2003
Apprenticeship 'Fachinformatiker'

**Professional career**

Since 2015
IT Expert, FESTO AG & Co. KG

Since 2014
Online tutor Software Quality Engineering and IT- Project Management, IUBH

2011 - 2015
Senior Consultant Software Test and Process Design, PRISM Informatics

2008 - 2011
Inhouse Software Quality Assurance, IDS Scheer AG

2003 - 2008
Software Developer, T-Systems
Agenda

Aim(s)

Decisions

Proceeding
AIM(S)
Aim(s)

Useful executable functional GUI-based system test

Ready for first test cycle

Portable

Usable for different test cycles
Decisions

1. UML model-based generation process

2. Using and extension of MOdel Compiler for generating Complete Architecture (MOCCA)

3. Modelling structure and behaviour (UML with (X)OCL) of tests utilisation of system under test (SUT)
Proceeding – Using MOCCA (1/2)

Proceeding – Using MOCCA (2/2)

1. Parse PIM
2. Validate PIM
3. Parse TPM
4. Validate TPM
5. Transform PIM
6. Generate Code
Proceeding – Modelling details (1/5)
Proceeding – Modelling details (2/5)

test case structure
concret test case

test1 : Testcase = new Testcase();
update test1.testcaseID  = "01";
testaction1 : Testaction = new Testaction();
update testaction1.action = "click";
update testaction1.testelement = testcontroller.view.bn1;
update testaction1.addition = "";
update test1.testaction->including(testaction1);
Proceeding – Modelling details (4/5)

**test execution model (1/2)**

Adding concrete test data in test structure

Execute test case – test action for test action

Execute checks (if given)

```java
begin
    current: TestAction;
    update current = currentTestcase.testaction->at(1);
    testElement: Control;
    update testElement = current testelement;
    update className = "JButton";
    if className = "JButton"
        then
            begin
                testButton: Button;
                update testButton = testElementoclAsType(Button);
                update executeAction(current.action, testButton)
            end
end
```
Identify action
executeTestcase()

Determinate action type
executeAction(Typ, )

Execute respective action
executeButtonClick()

Call of method in DPM
doClick()
## Proceeding – Modelling Details (5/5)

behaviour modelling with (X)OCL

<table>
<thead>
<tr>
<th>OCL</th>
<th>XOCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition to UML, current version 2.4</td>
<td>Addition to OCL, full action language</td>
</tr>
<tr>
<td>Textuell modelling</td>
<td>Additions:</td>
</tr>
<tr>
<td>Bounderries to models</td>
<td>⇒ Condition changes</td>
</tr>
<tr>
<td></td>
<td>⇒ Controlflow</td>
</tr>
<tr>
<td><strong>context</strong> Person <strong>inv</strong>: self.alter &gt;=0</td>
<td>⇒ Event modelling</td>
</tr>
<tr>
<td></td>
<td>if t = 1 then begin</td>
</tr>
<tr>
<td></td>
<td>update test.startTesting();</td>
</tr>
<tr>
<td></td>
<td>endif</td>
</tr>
</tbody>
</table>
BACKUP
Proceeding – Modelling details
design model
Modelling details

design platform model
Proceeding – Modelling details

Integration in system model