Peter Braun, Benjamin Flach, Reinhard Jeschull, Jan Philipps

Tactic-Based Testing

MBTUC 2011 | Berlin, 18-20 October 2011
Model-Based Testing

- Behavior model
  - Model determines search paths
  - Model provides reference values
  - Model returns coverage data

- Search criteria
- Test case generator
- Test cases
Lessons Learned

- Complex data spaces and algorithms
- Inclusion of prior knowledge of developers
- Modeling language and tool issues
Tactic-Based Testing

Main idea: Seamless combination of model- and script-based testing

Tactic determines search paths

Coverage over tactic, data and oracle models

Oracle model provides reference values

Tactic Interpreter & Coverage measurement

Test cases

Data model

Oracle model

Tactics
Tactic-Based Testing - Framework

Data model

Tactic Interpreter & Coverage measurement

Oracle model

Test cases

Tactics

JRuby as tactic language

Data models based on EMF, Java, ...

Oracle models in UML, Java or JRuby. Generic interface for other languages

Test case and coverage data output as XML

Validas AG
Tactic-Based Testing – Tactics

- Use of *tactics* to build up arbitrary search strategies
- The structure of the tactics defines the structure of the test cases

![Diagram showing Tactic parameters, Partial test case, and Set of partial test cases]

- **StepTactic**
- **StoreTactic**
- **OrElseTactic**
- **AndAlsoTactic**
- **IterativeDeepeningTactic**
Tactic Based Testing – Sample

```
incorrect_pins = [...] # List of 5 incorrect PINs

tacSelFile = step("select", {"file" => ["1C38", "69A5", "CA4B"]})
tacNothing = identity();
tacVerifyOnce = step("verify_chv", {"chvValue" => CORRECT_CHV})
tacVerifyTwice = step("verify_chv", {"chvValue" => incorrect_pins}) >> step("verify_chv", {"chvValue" => CORRECT_PIN})
tacRead = step("select", {"recNo" => [1, 3, 10], "mode" => [RecordMode::ABSOLUTE]})

# Try to read a record of a file after (in)correct/missing card holder verification
return tacSelFile >> (tacNothing | tacVerifyOnce | tacVerifyTwice) >> tacRead
```

Visualization of a tactic structure

Tactic written in JRuby

Results in 63 generated test cases
Validas AG

- Founded 2000
- 17 employees
- **Competences**
  - Model-based development
  - (Test-) Specification
  - Test automation
  - Tool qualification
  - AUTOSAR
- **Customers & Partners**
  - BMW
  - EADS
  - ESG
  - Giesecke & Devrient
  - Infineon
  - Audi/AEV
FreeRTOS – Interface Testing

Interface and robustness testing of off-the-self components

- Large parameter space
- Potential side effects
- Functional dependencies
- Hardware configuration (memory size, address layout, ...)

Sequences of interface calls
Parameter structures
Parameter data

Return code
Nontermination
Crash
FreeRTOS – Test Setup

- **Tactics**
  - Oracle model
  - Tactic interpreter
  - XML test cases

- **Code generator**
  - Test framework
  - Compiler
  - Compiled test cases

- **FreeRTOS**
  - Test control
  - Compiled test cases

**Test Case Generation**
- 41 tactics
- About 850 lines of JAVA code
- About 1600 lines of JAVA code

**Test Case Compilation**
- 30.444 test cases, 153.851 test steps
- 41 API-interfaces tested
- 30.444 test cases

**Test Case Execution on SUT (Cortex M3)**
- Execution time of about 7 hours

- 41 tactics
- 30.444 test cases, 153.851 test steps
- 41 API-interfaces tested

Validas AG
FreeRTOS – Test Results

Example: `pvPortMalloc (2^{32}-5)`
- Expected result: NULL-Pointer
- Observed result: Valid memory address
- Reason: Integer overflow

$(2^{32} - 5) + 16 = 2^{32} + 11 \Rightarrow 11$
Exhaustive testing of CREATE_FILE and its parameters to create elementary files on a smart card

- Complex parameters
- Functional dependencies
- High variability of parameter structures
- Configuration of the smart card (max. file size, reserved IDs, …)
Create File – Test Setup

Test Case Generation

- 16 tactics
- Data model
- Tactic interpreter
- Oracle model
- EMF-based, generated JAVA code

Test Execution on SUT

- About 1000 test cases
- Test cases
- Test executor
- Smart card
- Test results

Report Generation

- Report generator
- Report

About 900 lines of Ruby code (CREATE_EF and 10 supporting commands)

Execution time 7:30 minutes
Create File – Data Model

Generic and domain-specific data models based on EMF
Create File – Test Execution

Tactics to achieve test goals

Test reports & analysis support

Test execution environment
Lessons Learned

Complex data spaces and algorithms

Inclusion of prior knowledge of developers

Modeling language and tool issues
Summary

- Tactic-based testing for flexible control of test case generation
- Precise specification of test cases with varying parameters and structures
- Applicable to wide range of test objects