MODEL BASED TEST DESIGN FOR PERFORMANCE TESTING AND OTHER NON-FUNCTIONAL REQUIREMENTS

MATTIAS ARMHOLT
ERICSSON AB
Agenda

Introduction

Model Based Test Design for Performance Testing and other Non-Functional Requirements

Model Design Techniques

Conclusions
Introduction

My work
› Function tester at Ericsson AB
› Testing IP functionality in a middleware platform
› Working with MBT and automation for 3 years

Environment
› Conformiq Modeler – Model Design
› Conformiq Designer – Test generator
› Java and TCL/Expect – Test automation framework
Test models
Non-Functional Requirements (NFR)

› Capacity
› Performance Requirements
   – Response time
   – Throughput
   – Processor-utilization
› Interoperability
   – IP Standards
› Robustness
› Stability
› And more …
Problems With NFR

Lookahead depth
- Tool algorithm do not want to repeat the same transitions multiple times without fulfilling new requirements or covering new states/transitions.

Parametersation
- Requirements need transitions with several parameters to fire many times.
- The number of parameter combinations becomes unable to handle.
- Would generate thousands of test cases.
Problems With NFR

Robustness/Stability
- Nothing new should happen during test

Measurements
- No transition available in SUT for measurements

Capacity
- No clear boundary value to test with.
Method For MBT Of NFR

› Group the Non-Functional requirements based on similarities
› Evaluate if the group is possible to include in the model
› Design a test model including non-functional requirements
› Generate Test Cases
NFR Testability

- NFR requirements logic can be included in the test harness/environment
  - Use test applications for iterations
  - External equipment for interoperability
  - Add functionality to test harness
  - Increase SUT testability with test commands

- Testability is one criteria for MBT of NFR
NFR Test Logic Support

ITERATIONS

SUT MODEL

MANUAL INSTRUCTIONS

STABILITY

PERFORMANCE

TEST EQUIPMENT

SUT
NFR Model Design Techniques

› Design the non-functional requirements in the model with
  – Requirement keyword
  – Ad hoc requirements
  – States
  – Transitions
  – Parameters
NFR Model Design Techniques

› Group iterations together
  – Don’t create 1 host 100 times, create 100 hosts at 1 time
  – Removes risk of parametersation and lookahead depth
  – Reduces test case length makes it easier to read
  – Add logic in test harness or test environment
NFR Model Design Techniques

1. SetupNSessionReq
   - sessionid
   - setupNSessions
   - to in
   - 0.0
   - 1 (0x1 0x0 1 0b1)

2. SetupNSessionCfm
   - numberOfSetupSessionCfm
   - result
   - sessionId
   - msg
   - 999 (0x3E7 0x1747)
   - 0 (0x0 0x0 0b0)
   - 0 (0x0 0x0 0b0)

"999 sessions is set"

3. SetupSessionReq
   - sessionId
   - to in
   - 0.0
   - 1000 (0x3E8 0x175f)

4. SetupSessionCfm
   - result
   - sessionId
   - msg
   - 0 (0x0 0x0 0b0)
   - 1000 (0x3E8 0x175f)

"Session is setup"
NFR Model Design Techniques

› Use different abstraction levels
  - Focus the transitions to the parameters that counts for NFR
  - Use precondition when modeling NFR
  - Reduces the risk of unnecessary parameter combination testing
NFR Model Design Techniques
Conclusions

› In order to develop a good model covering non-functional requirements, you need to practice and learn how the tool generate test cases
› Support for testing of NFR must be possible to include in the test harness or test environment
› In general NFR increases logic and complexity in test harness and test environment
Conclusions

› Model cost of NFR the same compared to functional requirements
› Test harness/environment support development for NFR cost more compared to functional requirements
› Most valuable when NFR and functional requirements are modeled together
› Gain maintenance cost by MBT for all requirements
  – Cost less to maintain model + test harness compared to separate test scripts