Model-Based Testing for Enterprise Application Software: From Business Processes and Business Rules to Tests

SoftNet 2012
November 18-23, 2012 – Lisbon, Portugal

Author: Fabien PEUREUX
Contact: peureux@smartesting.com
Table of Contents

1 – Smartesting presentation

2 – MBT for Enterprise Application Software

3 – From Requirements to Tests

4 – Process summary
Company Profile

History

- Founded in 2003, privately held
- Spin-off of a Computer Science Lab in France (CNRS / INRIA)
- Supported by venture capital

- Independent Software Vendor & test solution provider
- HQ and R&D Center in Besançon, France
- Sales Office in Paris, France and Bangalore, India
Iterative Test Generation

Smartesting Certified

Test Management

Requirement Management

Business Models

Modeling

Impact Analysis

Test Management

Test scripts

Collaboration

Single Point of Maintenance

Business Analyst

Test Analyst

Testers

Requirements change

update

Sync
Test generation: global view
Table of Contents

1 – Smartesting presentation

2 – MBT for Enterprise Application Software

3 – From Requirements to Tests

4 – Process summary
Large-scale Enterprise Information Systems

- **System of systems & Complex composite systems**
  - Multiple applications
    - Mix of Bespoke and Packaged applications
    - Mix of data-oriented and process-oriented applications
  - Multiple targeted platforms (PC, Smartphone, Pad)

- **Testing needs**
  - Business workflow and business rules oriented
  - Application testing, but also end-to-end testing
  - Requirements and Business Process coverage
  - 80% of test execution still manual!
Model-Based Testing in a Nutshell

Model Assets for Automated Test Generation

Business Needs & Requirements

MBT Test Production

Automated Test Generation

MBT Automated Traceability

Test Repository

Test Execution

automated  manual
Model-Based Testing using Business Process models and Requirements

- Business Processes
- Requirements
- Smartesting CertifyIt
  Test Generation
- Test Plan
- Scripts for Automation
- Test-Requirement traceability
- Test Management Tool
- Tester
- Project Manager
- Validation:
  - Scenarios
  - Generated Tests
- Reporting:
  - Project Metrics
  - Progress Tracking
  - Coverage

- Modeling Functional Behavior and Business Rules
- Test Objectives
- Business Analyst
- Business Analyst
- Smartesting
- Business Analyst
Roles in the Model-Based Testing Process

- **Business Analyst**
- **Test Analyst**
- **Tester**
- **Automation engineer**

INDUSTRIALIZED PROCESS

**Model Assets**

- **Business Models & Flows**
- **Expected Behavior & Data**

**Test Assets**

- Defines action-word based testing automation

**Defined in**

- BPMN
- UML

© SMARTESTING 2012
Models for Automated Test Generation

- Business Process Model (BPMN)
- Business Entities and Logical Test Data (UML)
- Business Rules and Behavioral Model (UML/OCL)

Modeling notations
What Types of Tests?

- End-to-end testing, core business processes
- Acceptance testing of multi-applications
- Functional Testing of single applications
- IS qualification
- Integrated application services qualification
- Standalone business application qualification
- Model-Based Testing
Table of Contents

1 – Smartesting presentation

2 – MBT for Enterprise Application Software

3 – From Requirements to Tests

4 – Process summary
MBT Process for Information Systems

1. Phases & Activities

<table>
<thead>
<tr>
<th>Phases</th>
<th>Major Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>Project Management and Supporting Activities</td>
</tr>
<tr>
<td></td>
<td>Test Objective Analysis</td>
</tr>
<tr>
<td>Design</td>
<td>Modeling</td>
</tr>
<tr>
<td></td>
<td>Test Generation and Validation</td>
</tr>
<tr>
<td>Deployment</td>
<td>Publishing</td>
</tr>
<tr>
<td></td>
<td>Test Management and Execution</td>
</tr>
<tr>
<td></td>
<td>Script Development (Automation only)</td>
</tr>
</tbody>
</table>

Time:
- Start
- Iteration 1
- Iteration 2
## MBT Process for Information Systems

2. Major Inputs and Outputs by Phase

<table>
<thead>
<tr>
<th>Phase</th>
<th>Inputs</th>
<th>Activities</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preparation</td>
<td>• Test Objective Analysis</td>
<td></td>
<td>Test Objectives</td>
</tr>
<tr>
<td>Design</td>
<td>• Modeling</td>
<td>• Test Generation and Validation</td>
<td>Test Gen. Models</td>
</tr>
<tr>
<td></td>
<td>• Test Objectives</td>
<td></td>
<td>Gen. Test Plan</td>
</tr>
<tr>
<td>Deployment</td>
<td>• Publishing</td>
<td>• Test Management and Execution</td>
<td>Test Repo</td>
</tr>
<tr>
<td></td>
<td>• Test Plan</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Inputs:**
  - BP, Specs, Reqs
  - Test Objectives

- **Activities:**
  - Test Objective Analysis
  - Modeling
  - Test Generation and Validation
  - Publishing
  - Test Management and Execution

- **Outputs:**
  - Test Objectives
  - Test Gen. Models
  - Gen. Test Plan
  - Test Repo

- **Legend:**
  - Orange = Read-Only Input Documents
  - Blue = Artifacts Produced by the Process
Managing Test Requirements

Test Objectives

• Unique reference for “test” requirements
• Can be exported from existing requirement repositories
• Includes attributes such as priority, criticality, target release, etc.
• The “contract” between the BAs and the modeling team
Test Generation **process**

Smartesting Certifylt 5.3

- **Business Models**
- **Requirement Management**
- **Model Based Testing**
- **Test Management**

**Collaboration**

- **Business Analyst**
- **Test Analyst**
- **Testers**

**Smartesting Certifylt 5.3**

- **Formalize Capitalize Reuse**
- **Tests are Ready to be executed!**

Tests are ready to be executed!
Test generation

What do you want to test?

- Expected behavior
- Observation point
- Processes and flows
- Business rules to be tested
- Documentation of actions

How do you want to test it?

Testing Strategy
- Model coverage
- Test objectives
- Initial state

Test Analyst

Automated Test Generation
Models used for test generation

1- Business process models using BPMN

A business process with sub-processes in BPMN
Models used for test generation
2. Modeling Actions/Observations and Logical Data

🔹 UML Class
  - A generic way to capture the characteristics and operations
  - May have associations with other classes

🔹 UML Object
  - An instance of a class
A precise description of the requirements and business rules defines the expected behavior.

---@REQ: SALES/ADD_ORDER_ITEM

if (p_itemid <> ITEMID::INVALIDID) then
    ---@AIM: Possible to add a valid item to the order
    mess = MSGORDER::NONE
else
    ---@AIM: Impossible to add a invalid item to the order
    mess = MSGORDER::INVALIDITEMID
endif
Controlling Test Generation

Business Scenarios

Business Scenario = Instance of a Business Process
- Many possible scenarios
- Each task may have multiple outcomes (both valid (✓) and error (✗) cases): e.g. Task E has 2 valid cases + 3 error cases

How does it work:
- The business process defines all possible routes, each route is a scenario
- The user:
  - Builds scenarios by specifying 0 to n mandatory stops
  - Selects the task outcomes to exercise: combinations are possible!
- Test generator calculates the optimal route

How many scenarios can you imagine?
How many valid variations of A-E-F?
How many scenarios to test all cases of Task B?
Managing Test Data

1. Logical Data Vs. Physical Data

- A keyword driven approach for Model-Based Testing
  - Structured approach through the use of equivalence classes (the UML enumerations)
    - Enumeration literals → the “logical data” of the system (e.g. 
      \texttt{TS\_WEEK::CURRENT\_WEEK, USER\_TYPE::ADMIN})
  - Fits nicely in the paradigm of data-driven testing
  - But not a replacement to test data management

- Mapping Logical Data to Physical Data
  - Typically using a spreadsheet-like or table-like format
    - Logical data (enumerations) → headers of the columns
    - Physical data → values in the columns
    - Each line or row → one test execution
  - Applies to both manual and automated tests
    - Example for Automated Test Execution
Managing Test Data
2. Example for Automated Test Execution

Example of a `login()` keyword in a test automation tool

The initial form in the SUT. The Test Analyst created a `login(pLoginName, pPassword)` operation in the model. The login name corresponds to enumeration literals like `UNKNOWN_USER, ADMIN, ESS_USER1` (a regular user), `ESS_USER2, SUPERVISOR`, etc., that appear in the header of the data table below.

The implementation of the login keyword by the Automation Engineer: lines 99-110 perform required initialization based on input parameters; lines 111-113 automate the action of logging in with the right parameters.

The Data Table (created by importing the manually created Excel spreadsheet). The header correspond to the logical data, rows 1 to 3 to the physical data to use.
Generating the test plan
Publishing in the test management systems

Test cases are published to the test repository:
• In natural language for manual execution

• In robot language for automation, when needed
Table of Contents

- 1 – Smartesting presentation
- 2 – MBT for Enterprise Application Software
- 3 – From Requirements to Tests
- 4 – Process summary
From Requirements to Test: Summary

- **Input to start the behavioral modeling phase:**
  - Test Objectives that capture every test requirement (including business rules)
  - Business Process model

- **A minimal test project should include:**
  - A test generation model containing:
    - A UML class representing the SUT
      - With operations representing possible user/system actions
    - A UML package containing an instance of the SUT (“Initial Data”)
  - A Test Suite:
    - Pointing to the UML package “Initial Data”
    - No Test Selection Criteria (all tests targeted)
Summary – From Requirements to Tests

- Business Process models formalize the business or application workflows to be tested
  - Facilitating the communication between QA team and BAs
  - Modeling for test generation: Business Processes + Business Rules + Logical Test Data

- Automated test generation creates the test plan ready to be used in the test management tool
  - For manual testing
  - For automated testing

- Automated test generation based on Requirements coverage ensures high quality test plan
Thank you for your attention