A Structured Approach for Efficient Model-Based Testing in Large IT Projects

UCAAT 2013
22 – 24 October - Paris

Jean-Pierre Schoch – Bruno Legeard
{jean-pierre.schoch, bruno.legeard}@smartesting.com
Agenda

ábado

MBT for Large IT Systems
– Current challenges of testing large-scale applications
– Levels of testing addressed by model-based testing

Building the Test Generation models
– Understanding and controlling your test requirements
– Understanding and composing business process models
– Designing the test generation models

Reuse and Multi-Model Systems
– Enabling reuse and collaborative work
– Structuring models as a layered architecture
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 (and for some part will remain manual)
Model-Based Testing in a Nutshell

Model Assets for Automated Test Generation

Business Needs

Test Design Automation

Test Repository

MBT Test Production

MBT Automated Traceability

Test Execution

automated

manual
Roles in the Model-Based Testing Process

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

**Model Assets**

**Test Assets**

**INDUSTRIALIZED PROCESS**

- **Business model**
- **Test Generation Model**
- **Business Rules & Flows**
- **Expected Behavior & Data**

- Defines action-word based testing automation

Model refines:

Test assets:

- Business Analyst
- Test Analyst
- Automation engineer
- Tester

Test assets:

- Business Analyst
- Test Analyst
- Automation engineer
- Tester
Models for Automated Test Generation

Business Process Model (BPMN)

Business Rules and Behavioral Model (UML/OCL)

Business Entities and Logical Test Data (UML)

Modeling notations
What Types of Tests?

- **End-to-end testing**, core business processes
- **Acceptance testing** of multi-applications
- **Functional Testing** of single applications

- **Model-Based Testing**
- **Integrated application services**
- **Standalone business application**
Agenda

MBT for Large IT Systems
- Current challenges of testing large-scale applications
- Levels of testing addressed by model-based testing

Building the Test Generation models
- Understanding and controlling your test requirements
- Understanding and composing business process models
- Designing the test generation models

Reuse and Multi-Model Systems
- Enabling reuse and collaborative work
- Structuring models as a layered architecture
The Test Generation Process avec Smartesting CertifyIt

Phases

- Preparation & Analysis
  - Test Objective Analysis

- Design
  - Modeling
  - Test Generation and Validation

- Test Publication & Test Management
  - Publishing
  - Test Management and Execution (1)
  - Script Development (Automation only) (1)

Major Activities

- Project Management and Supporting Activities (1)

(1) Not covered in this tutorial
Preparation & Analysis
1. Defining Test Requirements

Tests typically created to verify specific requirements
- Formal or not
- Capture all test requirements in a Test Objective Charter (next slide)
- The TOC is used as the “contract” between:
  The different stakeholders, typically represented by business analysts and functional experts
  AND
  The test analysts responsible for designing the behavioral model(s)

References in the models to the covered requirements provide the basis for:
- Automatic traceability between requirements and generated tests
  • Traceability links are part of the info published into the test environment
- Accurate progress tracking
Preparation & Analysis
2. Test Objective Charter

Test Objective Charter

- 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

Use Cases
Textual Requirements
Application Mockups
And all Other Sources…
Preparation & Analysis
3. Capturing System Flows

⇒ System Flows = Sequences of operations and/or activities performed by human users and/or external systems, and the system’s various responses
  − Many types: textual or graphical, technical or business-oriented (e.g. workflows, business process diagrams, use case flows of events)

⇒ Business Flows = business view of the system under test
  − Identify Business Actions (BAs) = elementary business units
  − Tests = sequences of BAs
  − Business flows represented as Business Processes

⇒ Application Flows = technical details of the business flows
  − Identify Test Actions (TAs) = the “implementation” for the BAs
  − BAs are sequences of TAs
CertifyIt supports Business Process Modeling using BPMN (Business Process Modeling Notation)
Preparation & Analysis
5. Strategy for the Test Generation Process

❖ Top-Down Strategy
   – Most natural approach:
     • Create the BPs (BAs are produced)
     • Implement the BAs (TAs are produced)
     • Generate the tests

❖ Meet-in-the-Middle Strategy
   – If you have trouble identifying the “right” BPs:
     • Identify the BAs first
     • Implement them
     • Generate test cases using “test-only” BPs

Business processes as sequences of BAs
BPMN

BAs implemented as sequences of TAs
UML

Generated tests ready to be published
Tests
Building the Test Generation Models

1. Business and Application Scenarios

Business View:

- Create Timesheet
- Approve Timesheet

Application View:

- Login as Regular Employee
- Click ‘Create Timesheet’
- Find Employee
- Approve Selected
- Submit Timesheet
- Login as Admin
- Click ‘Approve Timesheets’
Building the Test Generation Models

2. Business and Application Scenarios (cont’d)

Business tasks are mapped to BA operations in the Test Generation model.

BA operations are mapped to “application” scenarios, i.e. sequences of TAs.

Business scenarios are instances of business processes.

Business scenarios are higher-level constructs that exploit the fine-grained UML model.
Building the Test Generation Models

3. Design Driven by BAs

💡 Business Actions

- The building blocks of your test projects
- Every test is a sequence of BAs
- Understanding the BAs is the key to a successful test project
  - This means first analyzing the BAs based on the test strategy and on the previous artifacts (TOC, Business Processes), and capturing the results in a BA specification

💡 The BA specification

- Prerequisites to use the BA
- Factors of variability: all the elements that impact the behavior of the BA (and that requires testing)
- Usage context: all the valid configurations for use of the BA (corresponding to possible combinations of the factors of variability)
- Application workflow: the actions that a user would need to take to achieve the desired outcome

White-box view of the BA
Building the Test Generation Models

4. Example of a BA Specification

Prerequisites: To be connected (any user)

Factors of variability:
- Connected user: Regular, Admin, Manager
- Leave type: annual leave, sick leave, family leave, etc.
- User inputs:
  - Success: 1 day or less, 2 days or more
  - Error: mandatory field(s) missing, invalid date format, etc.

Usage context:

<table>
<thead>
<tr>
<th>Connected user</th>
<th>Leave type</th>
<th>User inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>Each type</td>
<td>Success (2 days or more)</td>
</tr>
<tr>
<td>Regular</td>
<td>Indifferent</td>
<td>Success (1 day or less)</td>
</tr>
<tr>
<td>Regular</td>
<td>Indifferent</td>
<td>Each error case</td>
</tr>
<tr>
<td>Other users</td>
<td>Indifferent</td>
<td>Success (any case)</td>
</tr>
</tbody>
</table>

Application workflow:
1. Select the menu Leave > Apply
2. Fill out the form (based on the cases being tested) and click ‘Apply’
5. Equivalence Classes as Enumerations

Equivalence Classes

- Black-box testing technique
- Divides all possible inputs (and outputs) into equivalence classes:
  - The test that results from the representative value for a class is said to be “equivalent” to the other values in the same class
  - Example: UNDER\_AGE (less than 18), YOUTH\_AGE (between 18 and 25), ADULT\_AGE (over 25)

Modeled as enumerations to represent the values the factors of variability

- Each value documented in natural language, e.g.:
  - ANNUAL\_LEAVE = “Annual Leave”
  - EXCEEDS\_LEAVE\_ENTITLEMENT = “Enter a duration that exceeds the number of days available”
6. Modeling the BA

**BA = model operation**
- **Prerequisite:**
  - Precondition of the operation
- **Factors of variability:**
  - Parameters of the operation based on enumerations

- **Usage context:**
  - "Decision table" associated with the operation
- **Application workflow:**
  - Several solutions: here using "structured descriptions" in the BA operation
Double-click the task in the business process to open the decision table.

We will see later on how to select criteria to generate tests based on one or more lines in the decision table.
Building the Test Generation Models
8. Application Workflow as a Structured Description

CertifyIt provides the ability to use “multiple-step” description, which will eventually produce separate tests

- The two columns correspond to the “design steps” and “expected results” in the future tests
- On step #2, note the reference to the BA parameters and the use of “.description” to access their underlying descriptions in natural language
Building the Test Generation Models

9. Associate Test Requirements with the Model

❖ To associate a requirement with the model:
  – Drag-and-drop the requirement from the imported TOC (imported into the project) to the proper location in the model

❖ Using AIMs as Refinement of REQs
  – Requirements often too coarse-grained: “AIM” tags are used to provide additional information

The REQ is now divided into two AIMs. It will be completed when (and only when) the AIMs have been processed.
Building the Test Generation Models

10. Using Test Suites to Select Test Criteria

✔ Use business scenarios to create test objectives
  – Use dedicated keywords to target specific objectives (#behaviors below to select all NOMINAL_CASEs specified in decision table)
Building the Test Generation Models

11. Generating Tests
# Building the Test Generation Models

## 12. Publishing Tests

Generated tests can be published to most standard test environments (HP ALM, IBM RQM, Microsoft Excel, etc.)

### Successful Vacation Request - Annual Leave, Valid Request Multiple Days

<table>
<thead>
<tr>
<th>Business Action</th>
<th>Logical Data</th>
<th>Values</th>
<th>Expected Results</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple sources (see comment)</td>
<td>Standard profile required for this test</td>
<td>ESS (any ESS user)</td>
<td>Your user name is displayed in the page header on the right</td>
<td></td>
</tr>
<tr>
<td>Multiple sources (see comment)</td>
<td>Special profile required for this test</td>
<td>ADMIN (Admin)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Business Action</th>
<th>Step #</th>
<th>Action</th>
<th>Expected Results</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consult_Leave_Summary</td>
<td>1</td>
<td>Login as any ESS user.</td>
<td>Your user name is displayed in the page header on the right</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Select the menu 'Leave &gt; Leave Summary'.</td>
<td>The Leave Summary page. For an ESS, it shows for each leave type, the leave entitlements (in days), the number of days scheduled and taken, and the balance.</td>
<td></td>
</tr>
<tr>
<td>Apply_for_Leave</td>
<td>3</td>
<td>Select the menu 'Leave &gt; Apply'.</td>
<td>The Apply for Leave form is displayed.</td>
<td>CUSTOM: NOMINAL_CASE</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>• Select a leave of type 'Congés annuels'.</td>
<td>The message &quot;Successfully Submitted&quot; is displayed above the Leave Request form.</td>
<td>REQ: Leave/Apply/Leave Types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Then enter valid values in all form fields and a duration of at least two days.</td>
<td></td>
<td>ALM: Annual Leave</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Finally, click 'Apply'.</td>
<td></td>
<td>REQ: Leave/Apply/Submitting a Request</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ALM: Valid Request Multiple Days</td>
</tr>
<tr>
<td>Consult_Leave_List</td>
<td>5</td>
<td>Select the menu 'Leave &gt; My Leave'.</td>
<td>The leave request you successfully submitted in the previous step is now visible in the Leave list. The date, leave type and number of days should match what you had entered. The status is set to 'Pending Approval' with the number of days appended to the label. The Select Action list contains only one possible action: 'Cancel'.</td>
<td></td>
</tr>
<tr>
<td>Consult_Leave_Requests to Process</td>
<td>6</td>
<td>Log out</td>
<td>Your user name is displayed in the page header on the right</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Login as Admin.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Your should see the complete menu bar: Admin, PIM, Leave, Time, Recruitment, Performance, Help.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approve_or_Reject_Leave</td>
<td>8</td>
<td>Select the menu 'Leave &gt; Leave List'.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>in the 'Actions' column for the request to process, select</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

 Française MBT for Large IT Systems
 – Current challenges of testing large-scale applications
 – Levels of testing addressed by model-based testing

 Française Building the Test Generation models
 – Understanding and controlling your test requirements
 – Understanding and composing business process models
 – Designing the test generation models

 Française Reuse and Multi-Model Systems
 – Enabling reuse and collaborative work
 – Structuring models as a layered architecture
Multi-Model Systems and Reuse

1. Introduction to Multi-Model Systems

- **Applicability**
  - Large applications divided into modules
  - IT systems divided into separate applications

- **Each separate module/application becomes of separate test project**
  - See next slide for a representative architecture

- **Purpose of multi-model systems**
  - Need different levels of testing
    - Functional tests at the level of individual applications/modules
    - End-to-end tests involving two or more individual applications/modules
  - Enable collaborative work with minimal impact/cost
  - Enable reuse
Multi-Model Systems and Reuse
2. A Layered Architecture

Layered Architecture
- Involves a top-down, hierarchical structure of models
  - Models at one level use only models at lower levels, and
  - Are independent of client models

Different Types of Projects
- *E2E* project produces end-to-end tests based on high-level business processes
- *Module-n* projects are projects that produce functional tests at the application/module level
- *Common* projects capture enumerations and classes common to several other modules
Multi-Model Systems and Reuse

3. Reuse

葸 Reuse of UML elements: Common elements captured in a Common model project (or more) offer a first level of reuse but it remains limited (relatively few truly reusable elements) (see notes)

葸 True reuse is found in the reuse of business processes and the reuse of behavioral models (case of the E2E project)
  • Makes it possible to reuse full model projects “as is” (… when well designed)
  • Imagine for instance systems built around SAP modules or any other ERP…
Multi-Model Systems and Reuse

4. Collaborative Work in a Graphical Environment

✔ Best Practice: NEVER allow a model to be modified by more than one user at a time

✔ Recommended to use an architecture with multiple projects

✔ In all cases, use a version control tool to control access to your separate units (models and other artifacts)
  – The tool must support locking a file before it is modified: only one person at a time can make changes to a given unit
  – Many version-control tools available: open-source (such as CVS and SVN), IBM Rational ClearCase, ...
Thank you for your attention

twitter.com/smartesting