Model-Based Testing for Information Systems

From Business Requirements to Test Repositories

Author: Bruno LEGEARD
Contact: legeard@smartesting.com

October 2011
Agenda

  • Testing challenges in the context of Large-scale Enterprise Information Systems
  • Types of testing addressed by model-based testing
  • Composing Business Process models, Behavioral models and Test data models for automated test generation

🔹 Part 2: Step-by-Step Example with Smartesting CertifyIt........14
  • Use BPMN to describe Business Processes for Test Generation
  • Link Business Processes, Business Rules and Logical Test data
  • Define Requirement Traceability information
  • Generate tests with Smartesting CertifyIt
  • Export your tests in the test repository for manual test execution
  • Generate scripts for automated test execution

🔹 Part 3: How to start with MDT, adoption principles.................. 44
  • Test team organization, roles and responsibilities
  • Agile and model-based testing, a perfect fit
  • Model-based testing and Risk-based testing
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)

❖ **Risk mitigation**
  ➢ Quality Assurance (Testing) ensures a key role for risk mitigation
  ➢ Importance of compliance and regulation rules (SOX…)

❖ **Software quality has been becoming a Must have**
  ➢ Users don’t want to use buggy systems anymore
Testing focus areas of IT organizations

QA/Test function maturity: shift from tactical ad hoc process to a more strategic & centralized approach

**Top Four Focus Areas – Across Western Europe**

1. Choosing a testing methodology to address **agile/component based development life cycle**

2. Provide **automated test coverage** to build agility in testing

3. More focus on the **non-functional aspects** like performance, availability, security etc.

4. Having a test strategy that **optimizes use of testing services** (traditional and cloud based)

Source IDC - European Services, Enterprise Application Testing Survey, March 2011
Separate software testing from software development

Source: IDC - European Services, Enterprise Application Testing Survey, March 2011
Testing levels

Model-Based Testing for IT Systems
Where Does It Fit?

- Model-based Testing
- Integrated applications services qualification
- IS qualification
- End-to-end testing, core business processes
- Functional Testing of single applications

- Acceptance testing of multi-applications
- Functional Testing of single applications

‘Reducing risks in large projects’
‘Increasing Business Value of testing’
‘Traditional functional testing’
Testing methodology

RRBT – Risk & Requirements Based Testing

- Driving test effort on the basis of risk analysis & linking risks and requirements
- Several documented methodology inspired by RRBT:
  • Sogeti Tmap®
  • Logica RRBT
  • ISTQB / CFTL
Testing tools
Model-based testing for IT systems

Model Assets for automated test generation

Business Needs → Test Production → Test design automation → Automated Traceability → Test Repository → Test Execution

- Automated
- Manual
Test Generation Process

**Test Design**
- 3rd party Eclipse-based modeler
- Smartesting CertifyIt™
- Models → Generated Tests

**Requirements**
- Business Processes
- Tracability: Req ↔ Tests

**Manual Tests**
- Executable Scripts
- Automation layer

**Test Management**
- Requirement Management
- Test Design

**Short iterations**

**Agility**

- Test Analyst
- Business Analyst
- Manual Testers
- Test Automation Engineer
Models for Automated Test Generation

Business Process Model (BPMN)

Business Rules and Behavioral Model (UML)

Models used with Smartesting
Agenda

• Testing challenges in the context of Large-scale Enterprise Information Systems
• Types of testing addressed by model-based testing
• Composing Business Process models, Behavioral models and Test data models for automated test generation

азв Part 2: Step-by-Step Example with Smartesting CertifyIt.........14
• Use BPMN to describe Business Processes for Test Generation
• Link Business Processes, Business Rules and Logical Test data
• Define Requirement Traceability information
• Generate tests with Smartesting CertifyIt
• Export your tests in the test repository for manual test execution
• Generate scripts for automated test execution

азв Part 3: How to start with MDT, adoption principles .................. 44
• Test team organization, roles and responsibilities
• Agile and model-based testing, a perfect fit
• Model-based testing and Risk-based testing
Test Repository is fully driven from models + the testing strategy

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
- Configuration
- Initial state

Automated Test Generation
Models of Business Processes structure the Smartesting MBT solution for IT

- Business Process models formalize the business workflow to be tested
  - Facilitating the communication between QA team and Business Analysts
  - Simplifying modeling activities for Test Analysts (Business Processes + Business Rules + Logical Test Data)

- Business Process models may be reuse from upfront activities
  - From Business Process Analysis
  - From Requirements Elicitation

- Business Process modeling is based on standard BPMN notation from OMG
A business process with sub-processes.
Modeling Business Entities and Expected Behavior

Business entities and boundaries are described:
• at corporate level (shared by all apps)
• at application level

A precise description of the requirements and business rules define the expected behavior

```cpp
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
```
Publication to the Test Repository for Test Execution

Test cases are published to the test repository:

• In natural language for manual execution

• In robot language for automation, when needed
Composing BPMN models with behavioral models

- Business Processes described with BPMN
- Each “Service Task” in BPMN is linked to a Smartesting UML operation
- The Smartesting simplified UML stereotype is used to:
  - Simplify modeling for testing
  - Make the Business Scenarios issued from Business Processes executable (manual or auto)
  - Capture the business rules and expected behaviors
  - Provide requirements coverage
How to link BPMN with Smartesting UML (BPMN Side)

1. Define Interface Operations for each Service Task

2. Associate each Service Task with Interface Operations
How to link BPMN with Smartesting UML (UML Side)

- Declare UML operations for each Interface Operation
How to define detailed business rules

With OCL:

- OCL stands for Object Constraint Language
- Its goal is to express constraints on UML elements to overcome the limitations inherent to any graphical representation
- It manipulates objects and collections of objects
- Operations can be called in OCL
- OCL scripts are always in the context of a class
- OCL is used to express both the conditions under which an action is possible and the effects of this action

Add OCL when needed for each operation

Code snippet:

```plaintext
if (p_days = DAYSREQUESTED::NEGATIVE or p_days = DAYSREQUESTED::NONE) then
  ---@AIM: days input error
  self.mess = MSG::INVALIDDAYDATA and
  self.days = DAYSREQUESTED::NONE
else
  ---@AIM: correct form content
  self.mess = MSG::NONE and
  self.days = p_days
endif
```
How to determine the coverage of the Business Scenarios w.r.t. the BPMN model

Process Usage provides an accurate view of what paths in the business process have been covered or not.
A step-by-step example

**Objectives**

- How to develop test generation models from BPM and Smartesting UML diagrams
- How to use test generation models to automatically create test cases
- How to use the generated tests in the test management environment
Analyze Requirements:
BPM Example: OrangeHRM Recruitment Process
Analyze/Refine Your Business Flows: 1 - Building a List of Keywords (1/2)

Keywords as operations of the SUT

- Operation signature
  - Keyword specification
  - Contract between business logic and technical implementation

- Operation parameters can be used to model
  - Multiple choices/options (e.g. selecting a menu item)
  - User forms (e.g. login)

(continues next slide)
Analyze/Refine Your Business Flows:
1 - Building a List of Keywords (2/2)

- Documenting the keywords
  - Natural language documentation of what the user should do to perform the action
  - Example: documentation of <<business>> offerJob()

- Keywords as the basis for automation
  - Signature of the keyword-operations = interface between models and technical implementation
  - Documented in the “Adaptation Layer Specification” for the test automation engineer
Develop Behavioral Model:
2 - Input Parameters for Business Actions

Purpose: add variability to business actions
- Contribute to the documentation of the business actions
  - Example: (parameter type not shown)

```plaintext
«business» applyForJob ( pJobTitle )
```

Supported types:
- Enumeration classes
- Primitive types: Integers and Booleans
Develop Behavioral Model: 3 - Business Entities

Definition:
- Business entities correspond to the terms specific to the business domain (e.g. flight, traveler, reservation for an on-line flight reservation site) and are modeled as UML (entity) classes
- They have:
  - Characteristics modeled as UML attributes
  - Relationships with other classes modeled as UML associations
  - Behavior modeled as UML operations

![UML diagram of Company and Employee classes with relationships](image-url)
Develop Behavioral Model:
4 - Modeling Behavior

- OCL is used to express both the conditions under which an action is possible and the effects of this action.
- OCL is attached to operations (pre- and post-conditions).

Precondition - Login:

1. \( p\text{.LoginId} \neq \text{E\_EMPLOYEE\_ID::NONE} \) and
2. \( \text{self\_connectedUser\_oclIsUndefined()} \) and
3. \( \text{self\_employees\_exists( p : Person | p\_id = p\_LoginId)} \)
Develop Behavioral Model:
5 - Expressing Conditions in Pre/Post-Conditions

- In post-conditions, conditions represent possible application behaviors (including error cases) that can be tested.
- In pre-conditions, conditions are used to tune the model and filter out behaviors that are not possible.
- The *Decision Table* can be used to capture easily the business rules.
Develop Behavioral Model: 6 - Requirement Traceability and Test Aims

**Purpose:**
- Provide traceability links between test cases and requirements
- Keep track of the atomic behaviors covered by each test case

**In practice, tags are added to the model:**
- @REQ tags for requirements and @AIM tags for test aims
- In the effect of operations only

--- @REQ: example TO DO
--- @AIM:

**In most cases it’s easier to use the Smartesting tabular view:**

<table>
<thead>
<tr>
<th>p_creditduration : CREDIT DURATION</th>
<th>p_cat : GOOD CATEGORY</th>
<th>appCredit.message : MESSAGE</th>
<th>REQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREDIT DURATION::BETWEEN_12_48_MONTH</td>
<td>GOOD_CATEGORY::DOMESTIC_APPLIANCE</td>
<td>MESSAGE::CREDIT_ACCEPTED</td>
<td>Credit accepted</td>
</tr>
<tr>
<td>CREDIT DURATION::BETWEEN_12_48_MONTH</td>
<td>GOOD_CATEGORY::SOUND_VIDEO_GAME</td>
<td>MESSAGE::CREDIT_ACCEPTED</td>
<td>Credit accepted</td>
</tr>
<tr>
<td>&lt;Empty&gt;</td>
<td>else</td>
<td>MESSAGE::CREDIT_REFUSED</td>
<td>Good category excluded from offer</td>
</tr>
<tr>
<td>CREDIT DURATION::SUP_48_MONTH</td>
<td>&lt;Empty&gt;</td>
<td>MESSAGE::CREDIT_REFUSED</td>
<td>Duration restriction</td>
</tr>
<tr>
<td>CREDIT DURATION::INF_12_MONTH</td>
<td>&lt;Empty&gt;</td>
<td>MESSAGE::CREDIT_REFUSED</td>
<td>Personal amount restriction</td>
</tr>
</tbody>
</table>
Develop Behavioral Model:  
7 - Observation points

- Observation points are operations stereotyped <<observation>>

- Two approaches:
  - When the watched data has changed OR
  - When specific operations are called

- Like other operations, they can be documented and they can have OCL conditions (typically to further limit the situations when they are triggered)
Build Test Data:
8 - Defining the initial state of the system

- Modeled as a package containing the actual objects (instances of classes) to use in the test generation
  - Possible to define different sets of objects
  - Generated tests may differ from one set of objects to another
  - Impact of the test strategy in the object definition (see next slide)
  - Mapped to the physical objects during test execution
Validate Model:
9 - Requirement coverage

- Tag statistics in the modeling tool
- Traceability links
  - In Smartesting CertifyIt
  - In the target test management tool

Smartesting maintains a matrix of requirements covered in the test model
Can be accessed from the Smartesting console when checking/exporting the model

Indicates the requirement is included in the input requirement list
Indicates that the requirement is covered in the model
The requirement is included in the set that will be exported
Automated test generation

The test generation engine computes from the models all the test cases necessary to reach the defined coverage.
Requirements in Smartesting CertifyIt

Stories, tests and requirements are automatically linked together in Smartesting

The Stories view displays all the stories (i.e. the test objectives) computed by Smartesting CertifyIt
Execute Tests:
10 - Test Publication

- Target testing environment with requirement management traceability information if available
- Synchronization: Obsolete tests are deleted or replaced
- Publication can be customized (Java code provided, open API of the Smartesting publisher provided)

Requirements traceability is automatically managed by Smartesting and maintained in the Test Repository for coverage analysis and impact analysis.
Publication to the Test Repository for Test Execution

Test cases are published to the test repository:
- In natural language for manual execution
- In robot language for automation, when needed
Generated Repository is complete, executable and fully documented

Rational Quality Manager

Display Time-Track for Current Month (27-43-3d) Execution

Test Environment
Test Script Name
Display Time-Track for Current Month (27-43-3d)

Script Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Result</th>
<th>Description</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✅</td>
<td>Setup: initTest Technical init for automation</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✅</td>
<td>Body: login Fill the Username and Password fields with the values Caroline (password: pass), then click the &quot;Login Now&quot; button.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✅</td>
<td>Body: enter Time Identify the line matching the task Telesales. Then enter 5 in the box corresponding to the date Saturday last week. The displayed message is equal to &quot;MODIFICATIONS_NOT_SAVED&quot;. The remaining time for Saturday last week must be 5 hour(s). Time used on task Telesales for Saturday last week must be 5 hour(s).</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✅</td>
<td>Body: save TimeTrack Click the &quot;Save Time-Track&quot; button. The displayed message is equal to &quot;MODIFICATIONS_SUCCESSFUL_SAVE&quot;.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>✅</td>
<td>Body: confirmation</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>✅</td>
<td>Body: select ViewTime Select the &quot;View My-Time Track&quot; tab.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>✅</td>
<td>Body: show TimeTrack In the calendar, select the date (month or year) corresponding to CURRENT_MONTH. The displayed message is equal to &quot;DISPLAYED_UNTIL_TODAY&quot;.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>✅</td>
<td>Tear down: endTest</td>
<td></td>
</tr>
</tbody>
</table>
MBT for Automated Testing – Reuse of existing test components (‘HP BPT’ like approach)

Test Analyst

Test Repository

manual tests

automated tests

Test Generation

HP Business Components Requests

HP Quality Center

Automation engineer
Agenda

  - Testing challenges in the context of Large-scale Enterprise Information Systems
  - Types of testing addressed by model-based testing
  - Composing Business Process models, Behavioral models and Test data models for automated test generation

- **Part 2: Step-by-Step Example with Smartesting CertifyIT**
  - Use BPMN to describe Business Processes for Test Generation
  - Link Business Processes, Business Rules and Logical Test data
  - Define Requirement Traceability information
  - Generate tests with Smartesting CertifyIT
  - Export your tests in the test repository for manual test execution
  - Generate scripts for automated test execution

- **Part 3: How to start with MDT, adoption principles**
  - Test team organization, roles and responsibilities
  - Agile and model-based testing, a perfect fit
  - Model-based testing and Risk-based testing
Roles: Separation of Concerns

Model-Based Testing PROCESS

Business Analyst

Test Analyst

Automation engineer

Tester

BPMN

Smarttesting UML stereotype

Model Assets

Business Models & Flows

Expected Behavior & Data

Test Assets

defines action-word based testing automation

realized
## Business Analyst Role and profile

<table>
<thead>
<tr>
<th>Role</th>
<th>Actions</th>
<th>Skills (existing / new)</th>
</tr>
</thead>
</table>
| Business Analyst (BA) | • Write and draw business needs  
                        • Pilot testing by the risks             | ✔ Functional documentation capabilities  
                                                                                     | ✔ process notation (BPMN)  
                                                                                     | ✔ Test strategy |
| Profile            | • Knows the business processes and functional rules  
                        • Knows how to find missing information in the organization  
                        • Requirement oriented  
                        • Risk oriented  
                        • Test oriented  
                        • Ability to abstract (e.g. BPMN)  
                        • Customer oriented |
## Test Analyst Role and profile

<table>
<thead>
<tr>
<th>Role</th>
<th>Actions</th>
<th>Skills (existing / new)</th>
</tr>
</thead>
</table>
| Test Analyst (TA) |  ▪ Apply the test strategy  
▪ Model application behaviors and validate specifications  
▪ Generate test plans  
▪ Pilot test execution | ✔ Test professional  
➢ Smartesting modeling and test generation |
| Profile       | ▪ Good knowledge of functional testing methodologies within AGS  
▪ Knows Object Oriented methodology basics (or dev. experience)  
▪ Modeling experience can be a plus  
▪ Knows corp. testing tools  
▪ Knows project organization, lifecycle and test needs  
▪ Basic knowledge of test automation concerns |
### Other roles in the Smartesting project

<table>
<thead>
<tr>
<th>Role</th>
<th>Actions</th>
<th>Skills (existing / new)</th>
</tr>
</thead>
</table>
| Project Manager (PM)        | - Manage the test team  
  - Pilot testing by the risks                                           | ✓ Project management capabilities  
  ✓ Functional documentation capabilities  
  ➢ process notation (BPMN)  
  ➢ Test strategy                                                            |
| Testers                     | Execute the test cases manually                                         | ✓ Light knowledge on the application is a +                                               |
| Automation expert           | Develop the specified keyword library                                   | ✓ Skills on the robot required  
  ➢ Smartesting management of technical assets                                     |
| Implementation solution team | - Develop/customize the application for the customer                  | ✓ Development  
  ✓ Customizing  
  ✓ Writing technical specification  
  ✓ Unit Test                                                                     |
Agile Projects Challenges

Black box functional testing is poorly used..

Is it against the agile philosophy?

– NO! It’s just costly in highly iterative processes!

Agility renews the testing challenge
Model-Based Testing in Agile team
MBT in a Scrum Process
Model Based Testing gives you agility ‘by design’

- Responding to change
  - Test models are easier to update than large test suites
- Individuals & interactions
  - Testers & developers are working together
- Working software
  - Increasing test coverage and quality
- Customer/User collaboration
  - Test models are unambiguous communication tools
MBT and Risk-based testing

Risk analysis & assessment

Influences levels of modeling

Modeling Phase

Influences test selection criteria

Test generation Phase

Influences level of test automation

Test execution Phase

Requirements analysis
• **Test case production and maintenance time accelerated**
• **Complete coverage and** traceability of selected business rules
• **Support agility** based on easier test maintenance and rapid feedback
• **Easier interaction** between Business experts and test analysts

---

‘Align BAs And Quality Assurance Professionals To Drive Higher Quality — And Happier Customers’

*by Mary Gerush and Margo Visitacion, Aug 2010*