



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

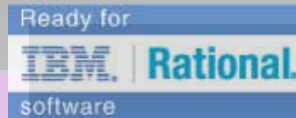
- 1 – Smartesting presentation
- 2 – MBT for Enterprise Application Software
- 3 – From Requirements to Tests
- 4 – Process summary

Company Profile



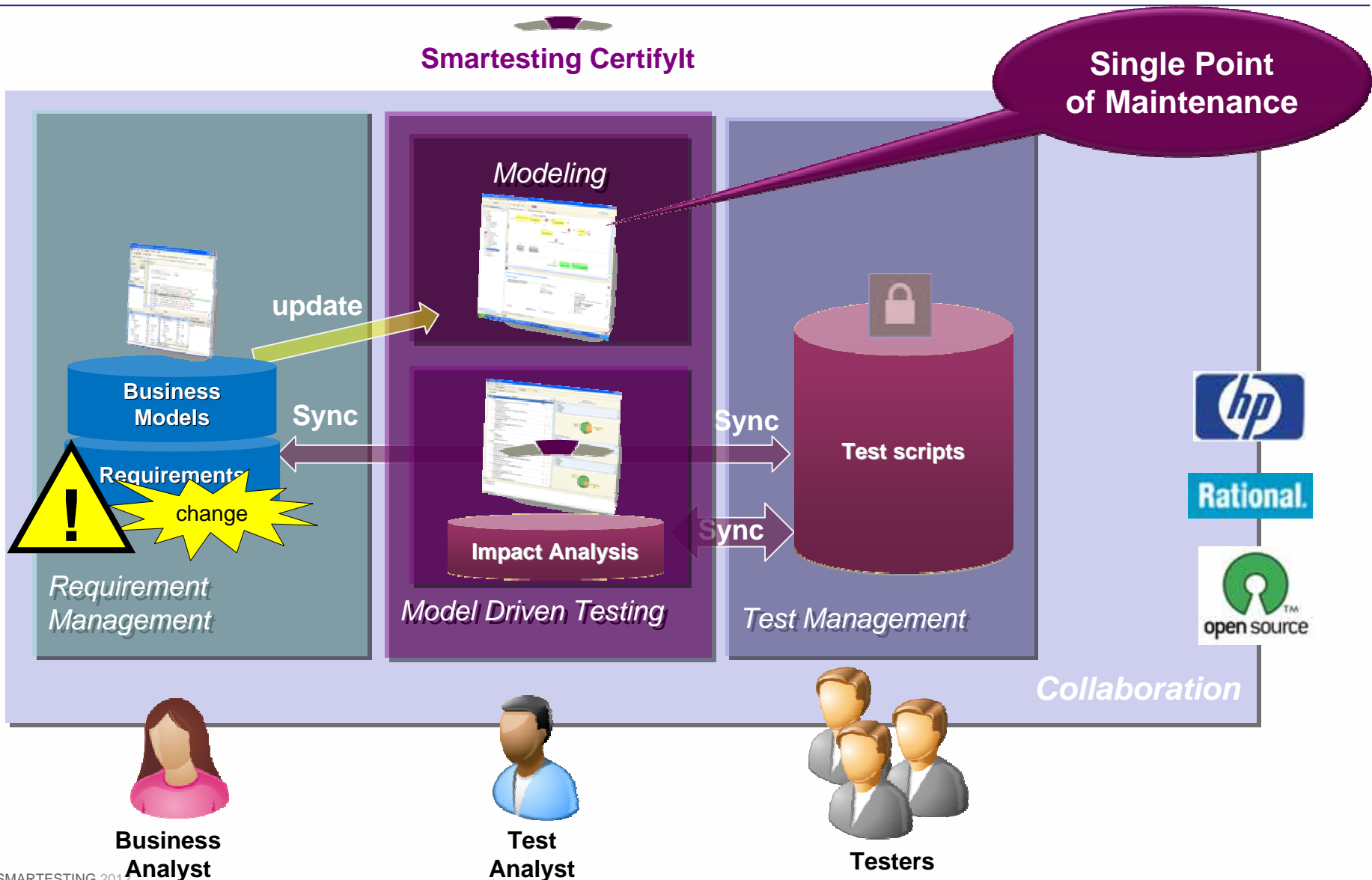
History

- Founded in **2003**, **privately held**
- **Spin-off** of a Computer Science Lab in France (CNRS / INRIA)
- Supported by **venture capital**
- **Innovation Awards** in 2006, 2007, 2010



- **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



Test generation: global view

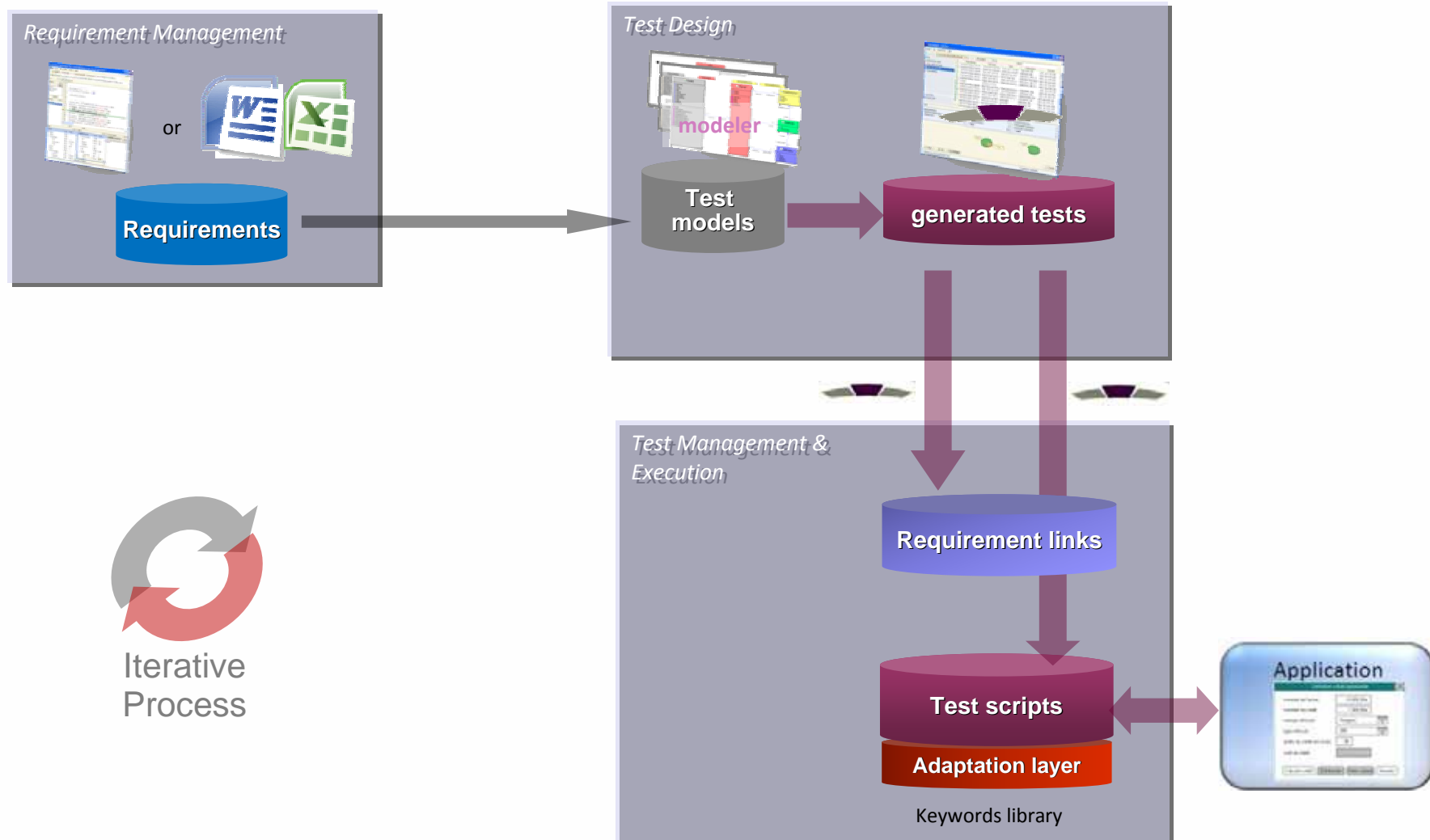




Table of Contents

- 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**

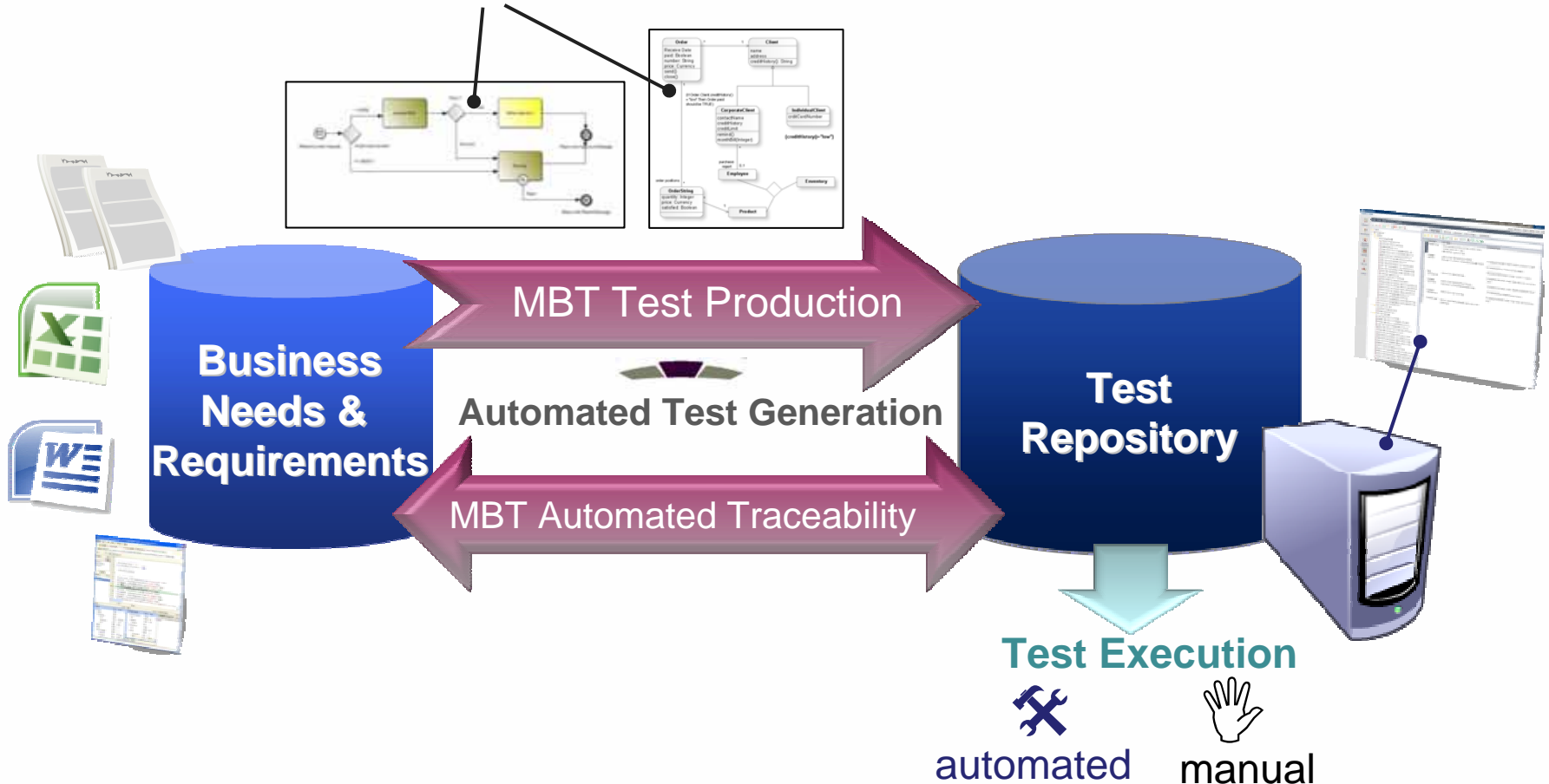
- Multiple applications
 - Mix of Bespoke and Packaged applications
 - Mix of data-oriented and process-oriented applications
- Multiple targeted platforms (PC, Smartphone, Pad)

⇒ **Testing needs**

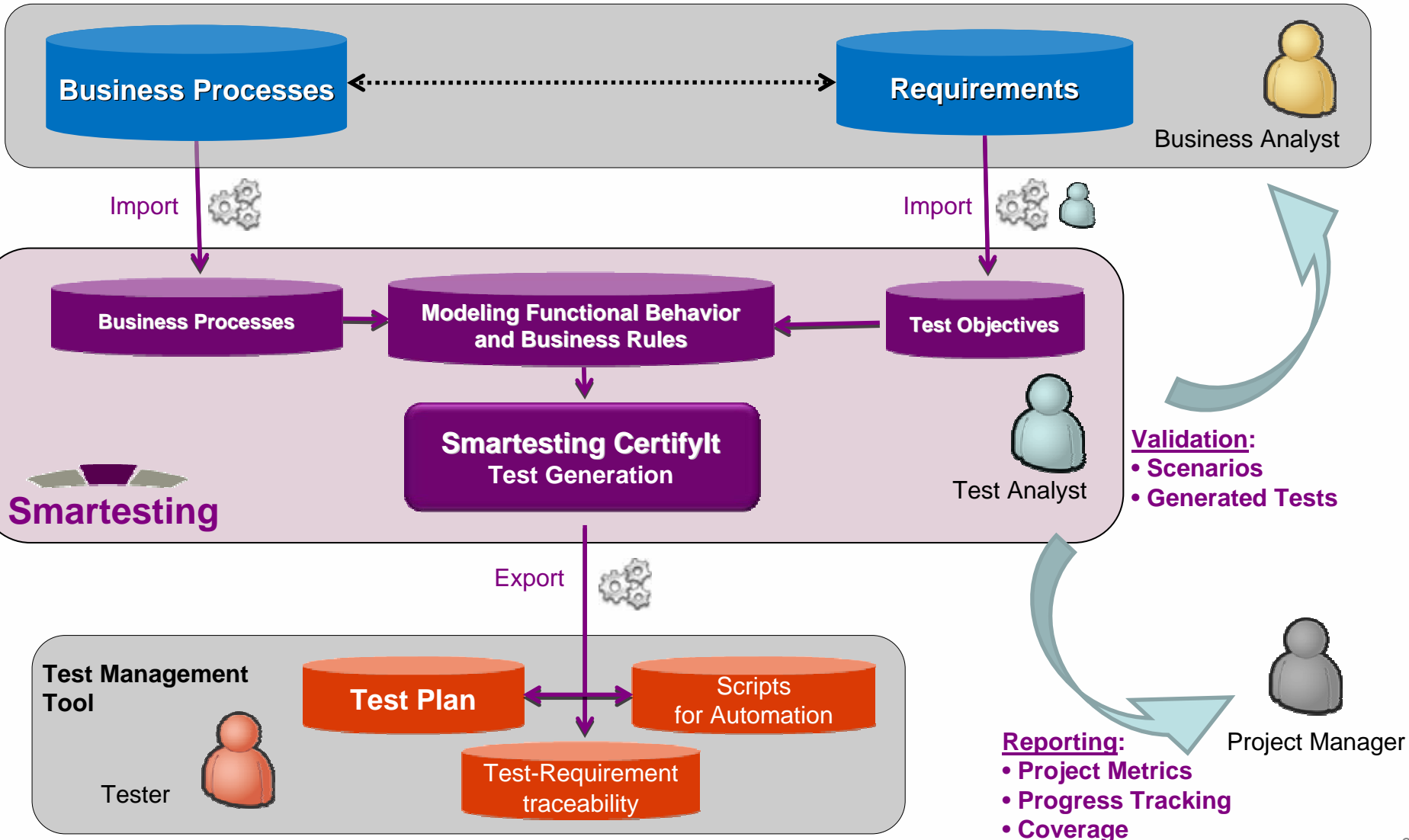
- Business workflow and business rules oriented
- **Application testing**, but also **end-to-end testing**
- Requirements and Business Process **coverage**
- 80% of test execution still **manual** !

Model-Based Testing in a Nutshell

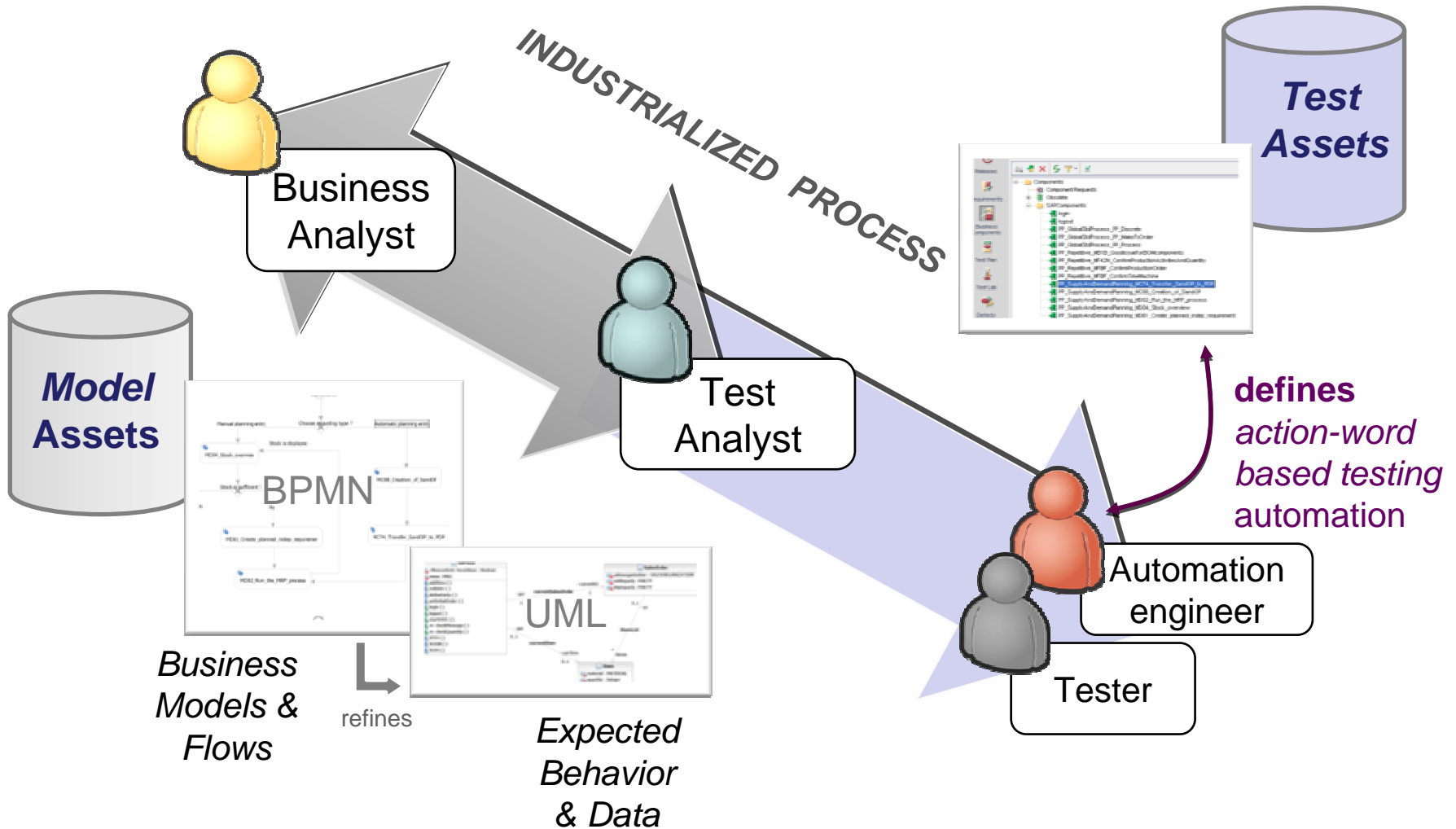
Model Assets for Automated Test Generation



Model-Based Testing using Business Process models and Requirements

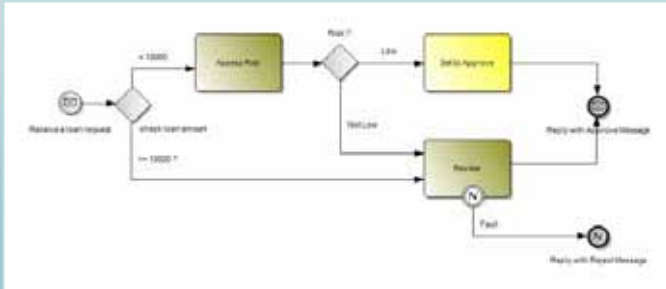


Roles in the Model-Based Testing Process

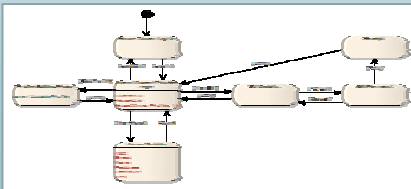


Models for Automated Test Generation

Business Process Model (BPMN)

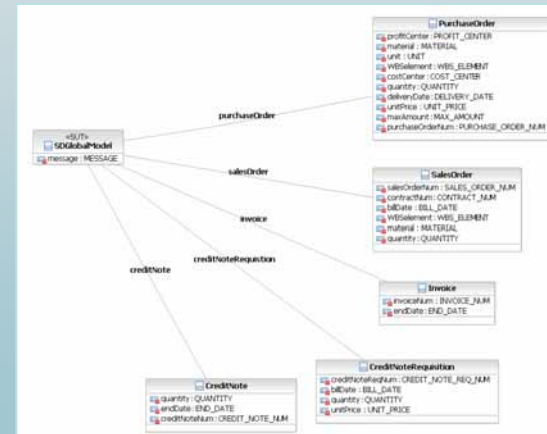


Business Rules and Behavioral Model (UML/OCL)



Conditions	Rules				
< 3 years	✓	✗	✗	✗	✗
>= 5 and < 10	✗	✓	✗	✗	✗
>= 10 and < 55 with concession card	✗	✗	✓	✗	✗
>= 10 and < 55 no concession card	✗	✗	✗	✓	✗
>= 55	✗	✗	✗	✗	✓

Business Entities and Logical Test Data (UML)



Modeling notations

What Types of Tests?

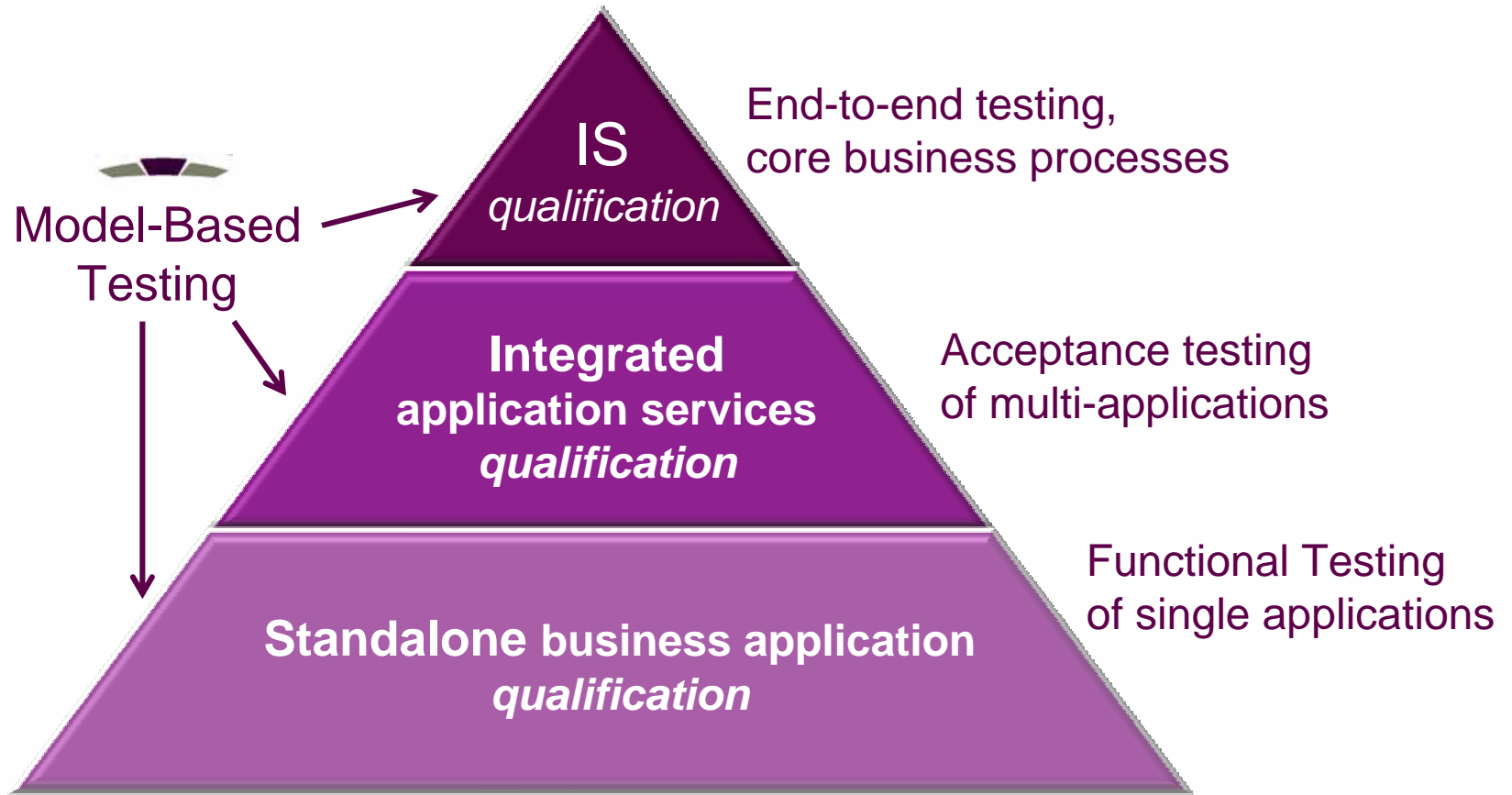




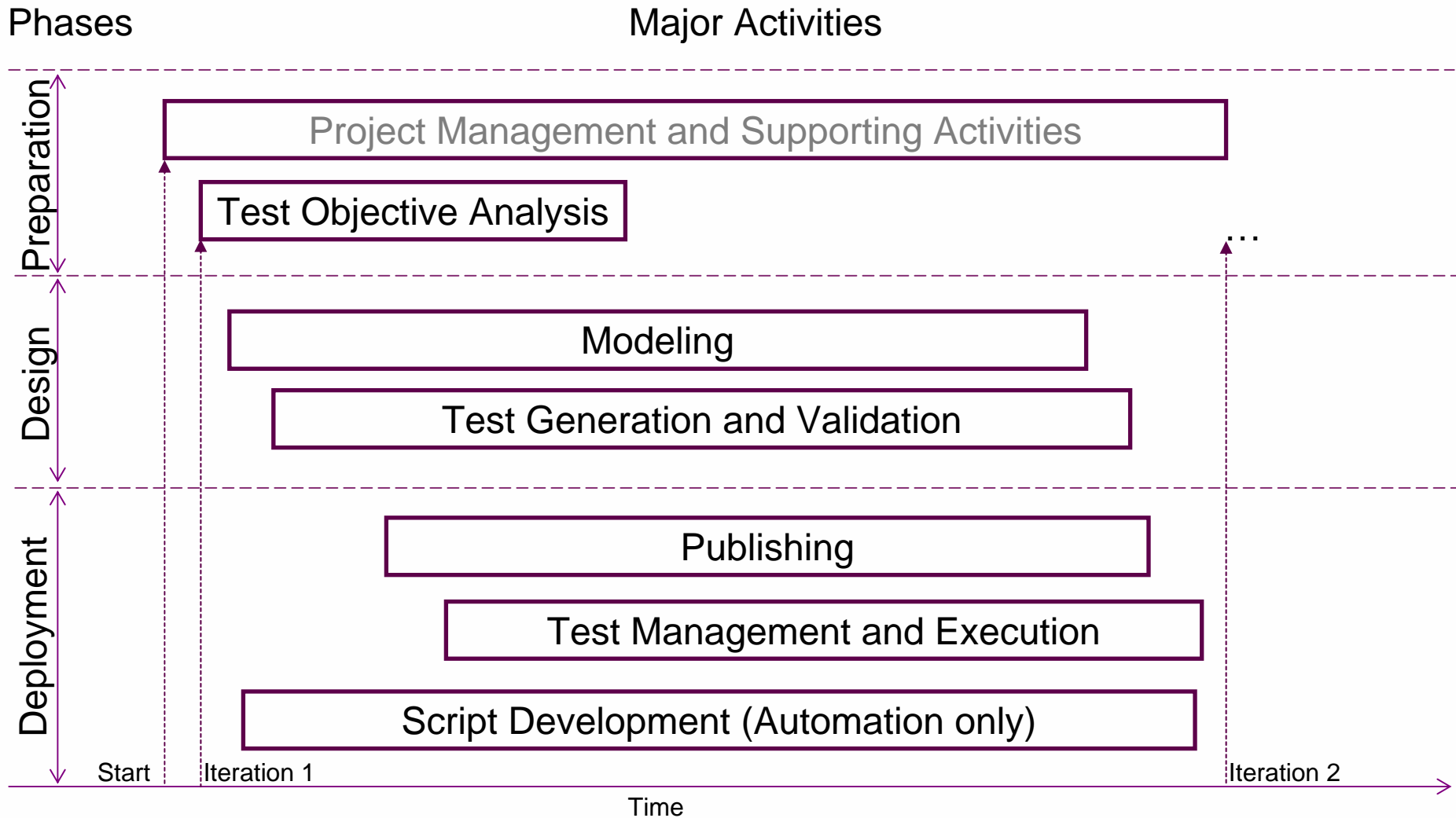
Table of Contents

- 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





MBT Process for Information Systems

2. Major Inputs and Outputs by Phase

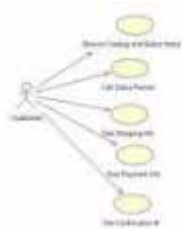
Phase	Inputs	Activities	Outputs
Preparation		<ul style="list-style-type: none">• Test Objective Analysis	
Design		<ul style="list-style-type: none">• Modeling• Test Generation and Validation	
Deployment		<ul style="list-style-type: none">• Publishing• Test Management and Execution	

= Read-Only Input Documents

= Artifacts Produced by the Process

Managing Test Requirements

Test Objectives

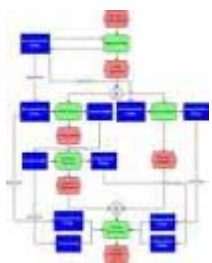


Use Cases

Textual Requirements



Application Mockups



Business Processes



And all Other Sources...

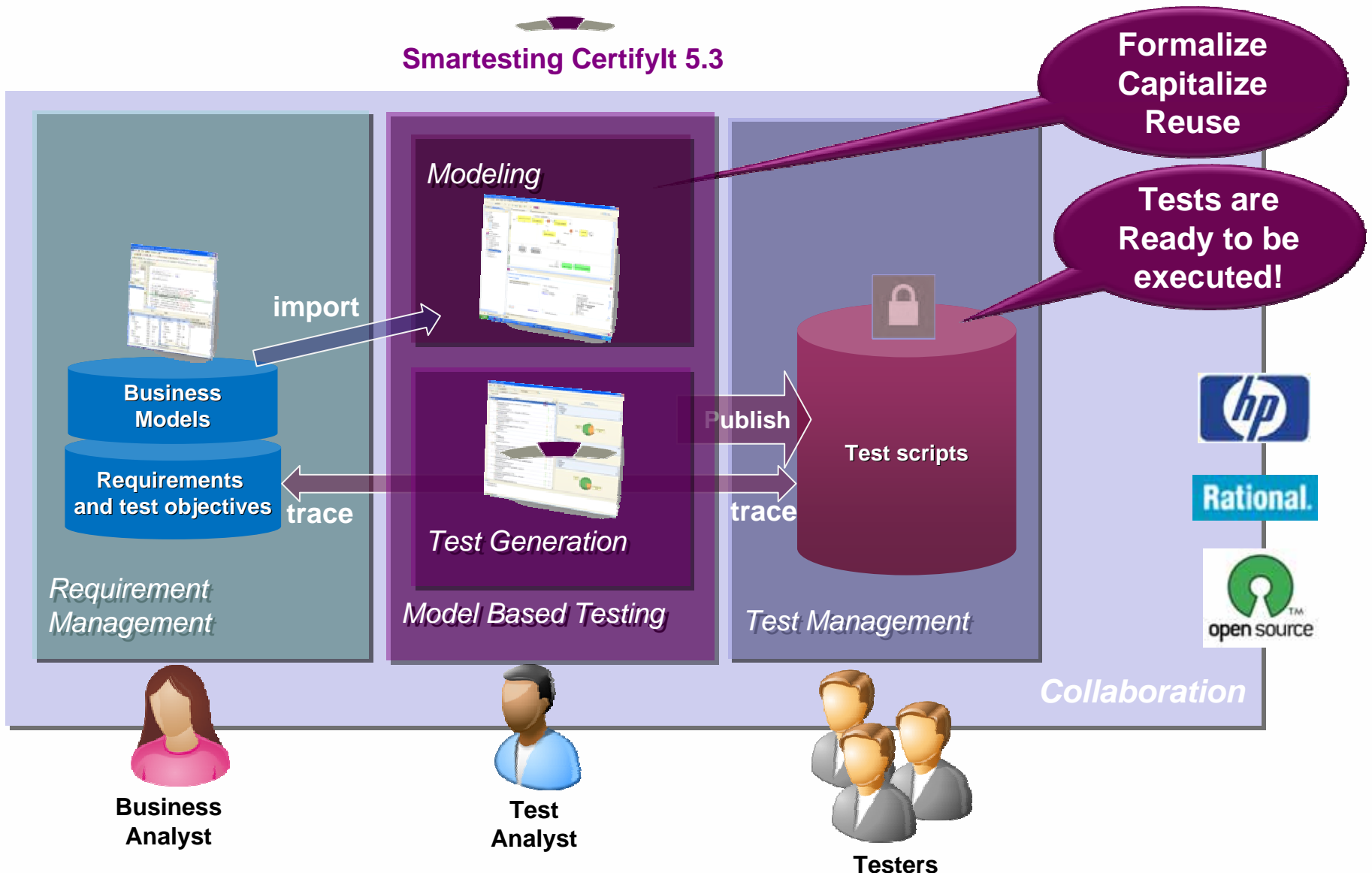
Test Objectives

	A	B	C	D	E
1	Requirement		Priority	Type	Brief desc
10	TIM01 - Enter and Submit Timesheet/Basi...	High	Use Case	Cannot cre	
11	TIM01 - Enter and Submit Timesheet/Basi...	High	Use Case	Cannot us	
12	TIM01 - Enter and Submit Timesheet/Alte...	Medium	Use Case	User can c	
13	TIM01 - Enter and Submit Timesheet/Alte...	Medium	Use Case	Central an	
14	TIM01 - Enter and Submit Timesheet/Alte...	Low	Use Case	Primary ar	
15	TIM01 - Enter and Submit Timesheet/Alte...	Low	Use Case	Billing Acc	
16	TIM01 - Enter and Submit Timesheet/Alte...	Medium	Use Case	User can c	
17	TIM01 - Enter and Submit Timesheet/Alte...	Medium	Use Case	User can c	
18	BP01 - Recruitment Process/Create Job Ve...	High	Business Process	User can a	
19	BP01 - Recruitment Process/Create Job Ve...	High	Business Process	Product in	
20	BP01 - Recruitment Process/View Job ...	High	Business Process	Can navigi	
21	BP01 - Recruitment Process/Apply for Job ...	High	Business Process	Added prc	
22	BP01 - Recruitment Process/View Applica...	High	Business Process	Display th	
23	BP01 - Recruitment Process/Schedule Int...	Medium	Business Process	User can s	
24	BP01 - Recruitment Process/Record Interv...	Medium	Business Process	User can a	
25	BP01 - Recruitment Process/Reject Applic...	Medium	Business Process		
26	BP01 - Recruitment Process/Offer Job/Sei...	High	Business Process		

- 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

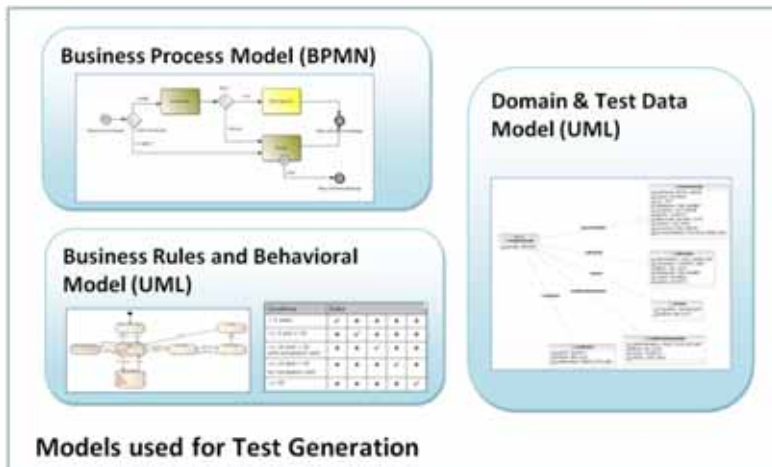
Test Generation *process*

Smartesting Certifylt 5.3



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?

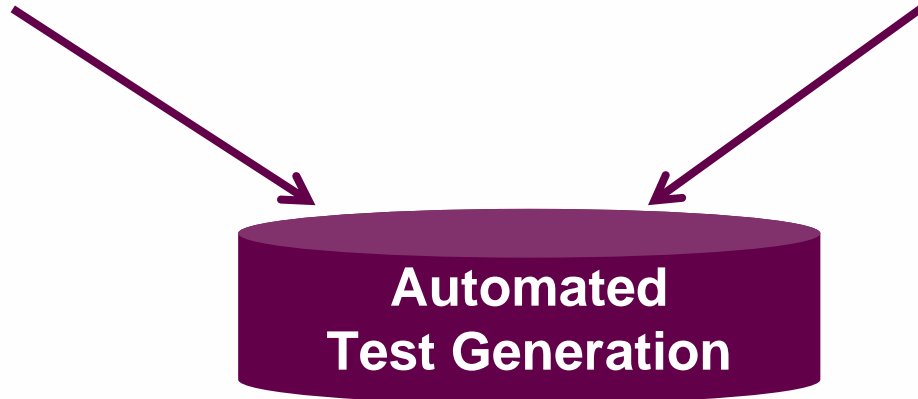
Test Analyst



Testing Strategy

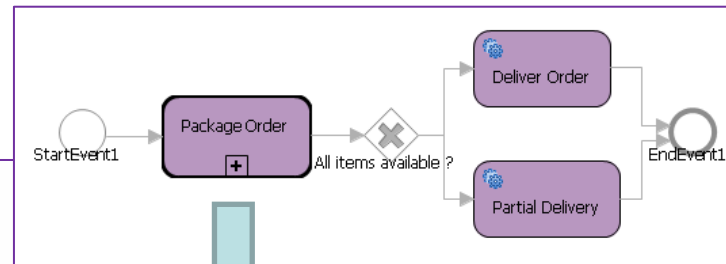
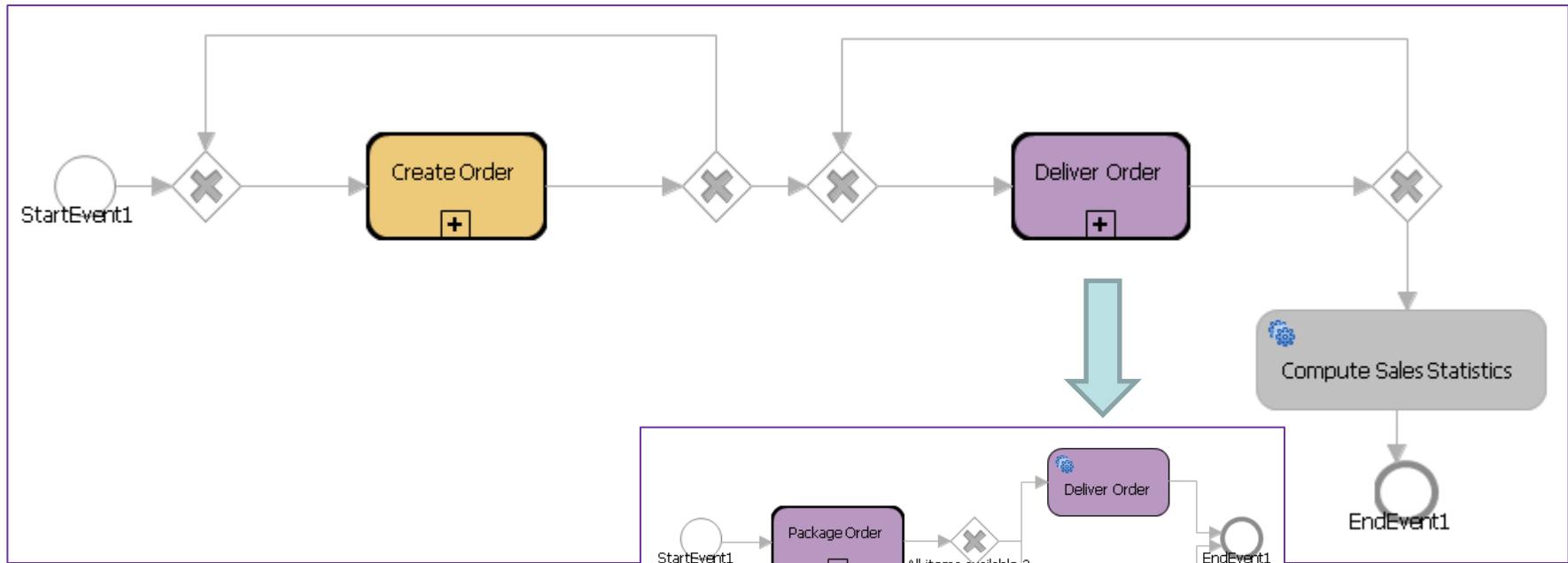
- Model coverage
- Test objectives
- Initial state

Automated Test Generation

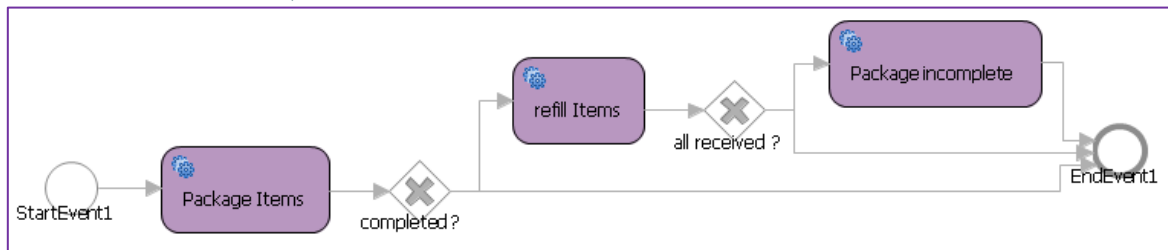


Models used for test generation

1- Business process models using BPMN



A business process with sub-processes in BPMN



Models used for test generation

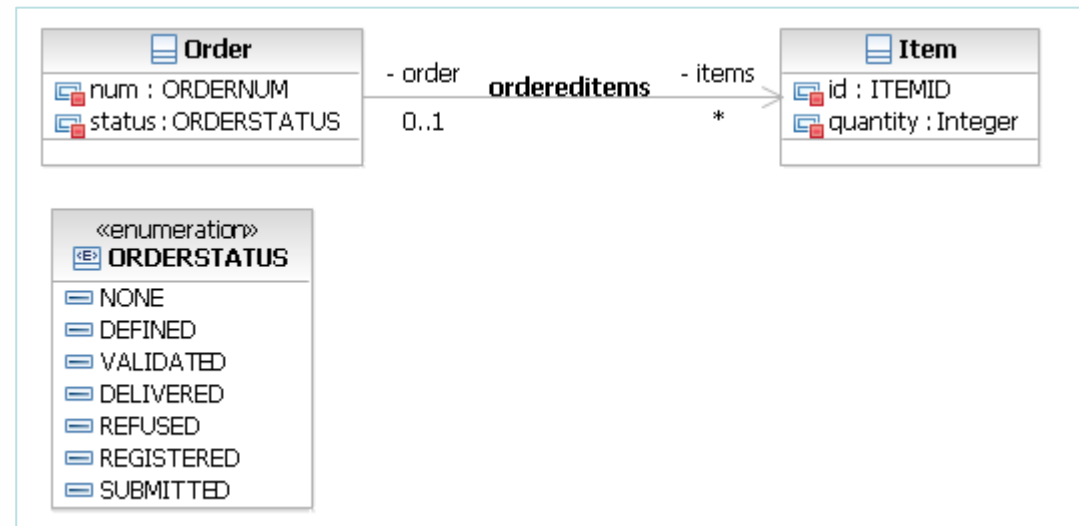
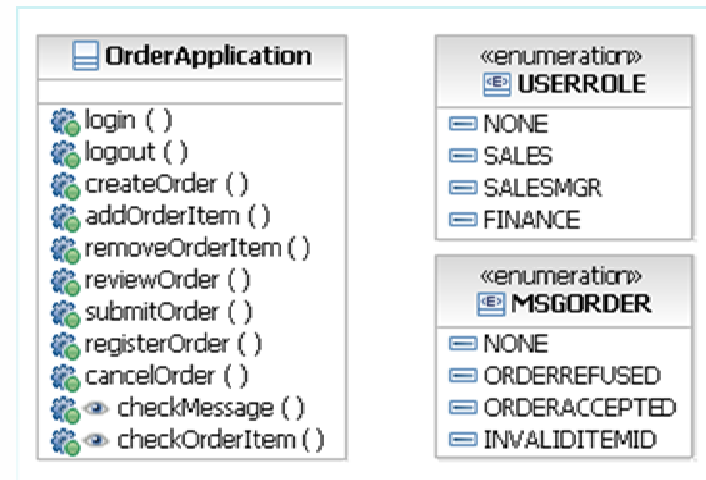
2. Modeling Actions/Observations and Logical Data

⇒ UML Class

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

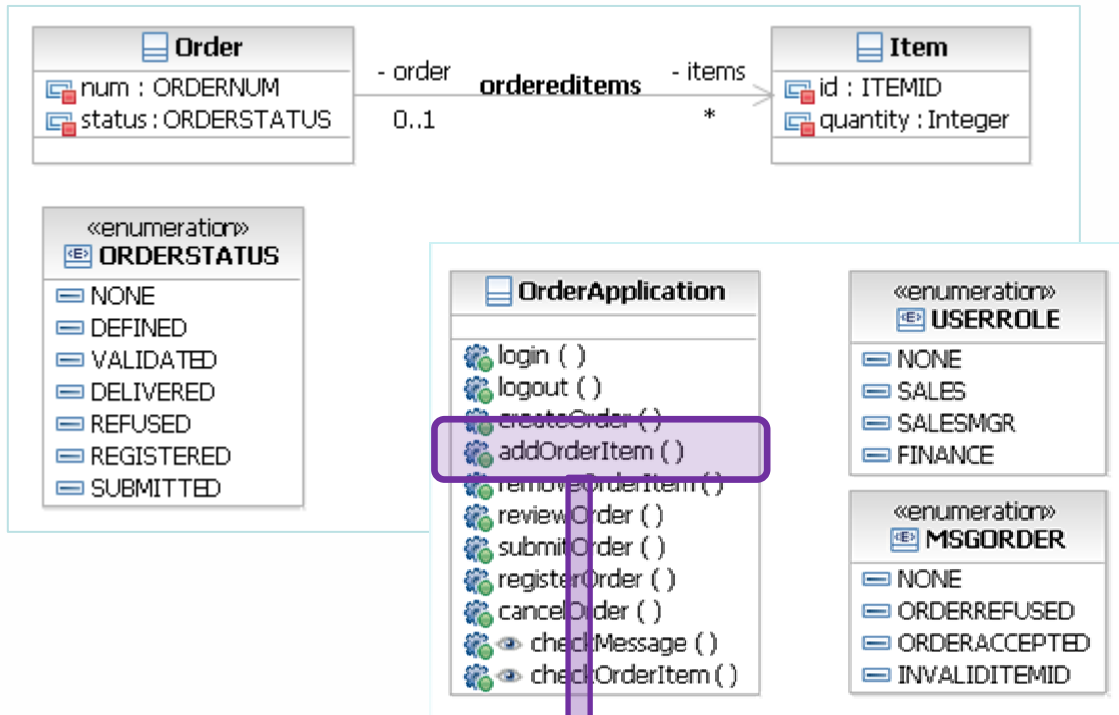
⇒ UML Object

- An instance of a class



Models used for test generation

3. Behavioral modeling



A precise description of the requirements and business rules defines the **expected behavior**

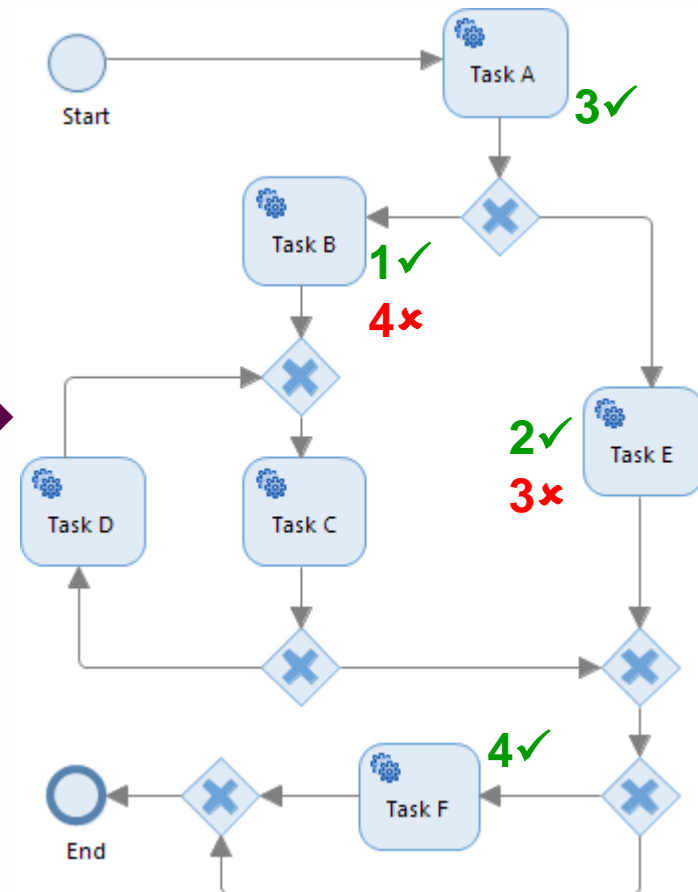
```
---@REQ: SALES/ADD_ORDER_ITEM

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. `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

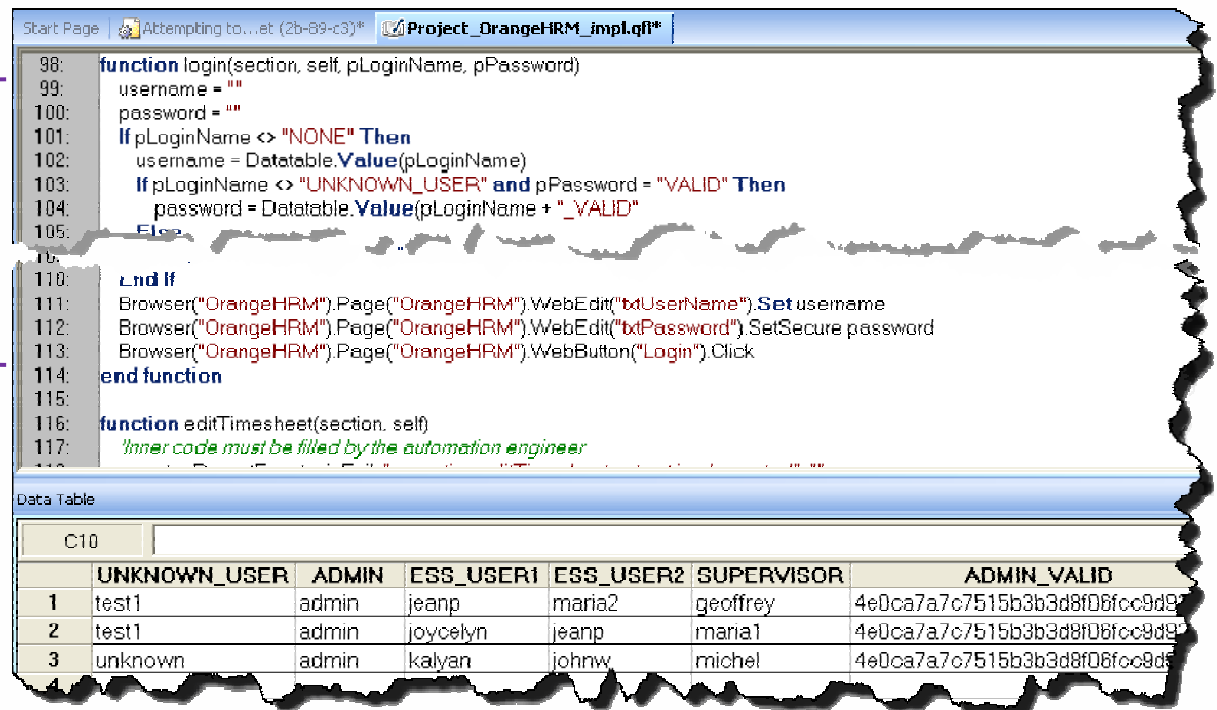


Login Name :

Password :

- ◀ 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



```
98: function login(section, self, pLoginName, pPassword)
99:   username = ""
100:   password = ""
101:   If pLoginName <> "NONE" Then
102:     username = Datable.Value(pLoginName)
103:     If pLoginName <> "UNKNOWN_USER" and pPassword = "VALID" Then
104:       password = Datable.Value(pLoginName + "_VALID")
105:     Else
106:       password = ""
107:     End If
108:   End If
109:   Browser("OrangeHRM").Page("OrangeHRM").WebEdit("txtUserName").Set username
110:   Browser("OrangeHRM").Page("OrangeHRM").WebEdit("txtPassword").SetSecure password
111:   Browser("OrangeHRM").Page("OrangeHRM").WebButton("Login").Click
112: end function
113:
114: function editTimesheet(section, self)
115:   'inner code must be filled by the automation engineer
116: end function
```

Data Table

C10	UNKNOWN_USER	ADMIN	ESS_USER1	ESS_USER2	SUPERVISOR	ADMIN_VALID
1	test1	admin	jeanp	maria2	geoffrey	4e0ca7a7c7515b3b3d8f08fcc9d9
2	test1	admin	joycelyn	jeanp	maria1	4e0ca7a7c7515b3b3d8f08fcc9d9
3	unknown	admin	kalyan	johnw	michel	4e0ca7a7c7515b3b3d8f08fcc9d9

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

The screenshot displays a test management system interface. On the left, a tree view shows the test suite structure under 'OrderTestSuite', including test cases like 'addOrderItem (b2-fb-6d)', 'cancelOrder (b2-1e-22)', 'registerOrder (b2-a3-1c)', and 'removeOrderItem (b2-62-3c)'. The main area shows a table with test case details:

Description	Expected Result
Login to the application with <<<SalesRepresentativeUser>>>	
Menu / Order / Create New Order	The order just created will be defined by #ORDER_002
On order # ORDER_002, add 1 items of type Restricted	
Menu / Order / Submit Open order # ORDER_002 from the list and submit	ORDER: ORDER_002 ITEM: Restricted Quantity: 1
MenuLogout	
Login to the application with <<<SalesManagerUser>>>	
Menu / Order / Review	Order # ORDER_002 should be in the list
Locate order # ORDER_002 in the list	The displayed message should be: "Order has been correctly entered in the system"
MenuLogout	
Login to the application with <<<FinanceUser>>>	

• In robot language for automation, when needed

```
Details Design Steps * Test Script * Attachments * Req Coverage * Linked Defects

Action1
1: login "BODY", "OrderApp", "SALES"
2:
3: createOrder "BODY", "OrderApp"
4:
5: getCreatedOrderNumber "BODY", "OrderApp", "NUM2"
6:
7: addOrderItem "BODY", "OrderApp", "NUM2", "RESTRICTEDITEM", "1"
8:
9: submitOrder "BODY", "OrderApp", "NUM2"
10:
11: checkOrderItem "BODY", "OrderApp", "ItemInstance2", "NUM2", "RESTRICTEDITEM", "1"
12:
13: logout "BODY", "OrderApp"
14:
15: login "BODY", "OrderApp", "SALES"
16:
17: displayMgrReviewList "BODY", "OrderApp"
18:
19: checkMgrReviewList "BODY", "OrderApp", "Order2", "NUM2"
20:
21: reviewOrder "BODY", "OrderApp", "NUM2"
22:
23: checkMessage "BODY", "OrderApp", "ORDERACCEPTED"
24:
25: logout "BODY", "OrderApp"
26:
27: login "BODY", "OrderApp", "FINANCE"
28:
29: registerOrder "BODY", "OrderApp", "NUM2"
30:
```



Table of Contents

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

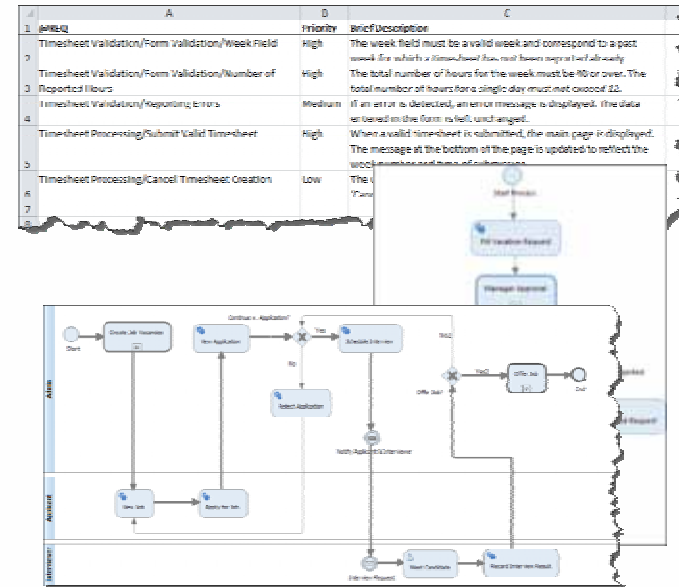
From Requirements to Test: Summary

➔ 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



twitter.com/smartesting

