ARYA Testbench Team

ARYA Testbench
Quality Assurance Plan

Authors : **Aurélien Tamas-Leloup**
-Ryan Shipp-Yuanbo Wang-Yanchao Wang-Alexandre Vey
Version 1.0

ARYA Testbench Team

Revision History

Date	Version	Description	Author
11/09/15	0.1	Added a few introduction details	A. Tamas-Leloup
11/15/15	0.2	Minor corrections	R. Shipp
11/17/15	0.3	Added Software Quality Attributes, Acronyms, definition and tips for next parts	A. Tamas-Leloup
12/10/15	0.4	Added the releases and SVT schedule	Yu. Wang
01/15/16	0.5	Added Acronyms, definition, Added Test Coverage matrix Added Test metrics	Ya. Wang
01/17/16	0.6	Added Topologies	Yu Wang
01/18/16	0.7	Added the risks	A.Tamas-Leloup
01/19/16	0.8	Review	Ya. WANG
01/20/16	0.9	Review	R. Shipp

Table of Contents

ARYA Testbench Team

Quality Assurance Plan

1. Introduction

1.1 Purpose

This document explains the Software Quality Assurance Plan of ARYA Testbench. It describes the procedures and control methods that will be used to reach the desired level of quality for the product.

1.2 Scope

This Quality Assurance Plan will be associated with the product called ARYA Testbench. This product is a Java web application whose goal is to run qualification scenarios on different ESBs. It will be developed for Thales as part of the SWIM project.

1.3 Definitions, Acronyms, and Abbreviations

TERM/ACRONYM	DEFINITION
ESB	Enterprise Service Bus
SRS	Software Requirements Specification
UR	User Requirements
SDD	Software Design Document
STD	Software Test Document
QAP	Quality Assurance Plan
UC	Use Case
DP	Design Pattern
SOA	Software Oriented Architecture
SOAP	Software Object Access Protocol
KPI	Key Performance Indicator
SWIM	System Wide Information Management
SVT	Software Verification and Test
Producer	A service linked to the ESB which provides responses according to the request it receives
Consumer	An application which sends requests via ESB to producers

2. Quality Objectives

In order to summarize the different quality objectives for our project, we provided the following table. It goes over each software quality attribute and indicates the applicability of the attribute, its importance and a short explanation.

Quality Attribute	Applicable	Importance (from 1 to 5)	Explanation
Usability	yes	5	We will provide an user-friendly design to make the software very easy to use
Learnability	yes	4	Usability will probably enable learnability.
Modularity	yes	5	Our design will divide the system in the smallest part possible
Efficiency	yes	5	Some parts of the system need to respect performance aspects
Safety	no		
Security	no		
Reliability	yes	3	The product will be tested throughout the construction phase. We will also provide a test plan as well as a STD.
Resilience	no		
Robustness	yes	2	Input will be checked but robustness will not be our main focus
Understandability	yes	4	We will set objectives about our code/comment ratio
Testability	yes	4	We will set code coverage objectives
Adaptability	no		
Complexity	no		
Portability	no		
Re-usability	yes	3	Documentation will be provided with the software.

3. Management

3.1 Organization

The Quality Assurance Manager will be responsible for the quality of the product. Any issues endangering the quality shall be reported to him.

3.2 Roles and Responsibilities

Team Members	**Roles**
Aurélien Tamas-Leloup	Project Manager
Ryan Shipp	Quality Assurance Manager
Yanchao Wang	Requirement Manager
Yuanbo Wang	Test lead
Alexandre Vey	Design Lead

4. Documentation

The essential documentation of our project will include:

- Software Requirements Specification (SRS)
- Software Design Document (SDD)
- Software Test Document (STD)
- Product Breakdown Structure (PBS)

5. Main Features Included in ARYA Testbench V1.0

The following is the release contents as detailed in the ARYA Testbench v1.0 Plan DCP. System tests will cover them functionally through various test methods as described in our STD document.

MAIN FUNCTIONS	DESCRIPTION	ACTORS
Manage Scenarios	Users carry out management work on the scenarios	Users
View Results	Users see the results of the scenarios executions	Users

Execute a scenario	Users selects one scenario and launch it	ESB, Users
Configure nodes	Users configure nodes, and topology of the test target	Users

The list of requirements and their details are included in System Requirement Specification(SRS) document.

6. Release and SVT Schedules

The SVT dates are listed in the schedule below, which shows the start and end date for each sprint. Four sprints are expected for ARYA project, and milestones for each sprint are specified.

Key Project Event	Date	Milestone
Sprint 1 start	04/12/2015	Basic UI, Configurable dummies, Launch scenario, Start working on the broker
Sprint 1 end	18/12/2015	
Sprint 2 start	19/12/2015	Full beautifull UI, Launch scenario, Create Scenario
Sprint 2 end	06/01/2016	
Sprint 3 start	07/01/2016	Retrieve results, Display results
Sprint 3 end	18/01/2016	
Sprint 4 start	19/01/2016	Configuration Management Deployment
Sprint 4 end	28/01/2016	

7. Metrics

1. JUnit Test

The number of opened issues is the first metric that we will take into account in order to make sure the whole team is always on track. The failed unit tests should be quickly reported and, if necessary, the related issues should be reopened and retested in JIRA. Code coverage is the second metric that will be tracked. At least 80% of code written should be covered by unit tests (JUnit) .

2. Coding Style

a. Comment lines

We need at least 10% comment lines

b. Code convention

The code conventions of the language should be respected.

c. Function length

Each function in the code should not be longer than 100 lines.

d. Method parameter number

The average number of parameter for functions should be less than 5. There should be not be functions with more than 7 parameters.

e. Nested if-else statement

There should not be nested if-else statements with a depth greater than 4.

f. Nested loop

There should not be nested loops with a depth greater than 3.

Get an A for code quality according to the default rules defined by SonarQube.

3. Defect Management:

Defect management is another important metric for us. A defect chart shall be provided to show the state of each defect and its impact on the evolution of the project. A curve shall show the defect number resolved during each sprint.

8. Evaluation and Test

Entry Criteria	1. Sprint backlog has been written by the scrum master and approved by the whole team 2. A JIRA task has been defined for each user story and assigned to the members of the team 3. Each member of the team should be able to make the lastest project version to work. 4. Planned test cases are approved.
Exit Criteria	1. All bugs & defects have been resolved 2. All features of the current sprint have been implemented 3. All features of the current sprint have been tested by a different developer

	4. All members of the team have attended a Sprint review meeting

Test Coverage Matrix

Test Type→ / *Coverage Area*	*UNIT*	*SVT*		
		SVT End-to-End	*SVT Performance*	*SVT Load/Stress*
Database Access Object				
Result	X	X	X	X
Scenario	X	X	X	X
Configuration	X	X	X	
App Controller				
Scenario Validation and Execution	X	X	X	X
Scenario Validation and Storage	X	X	X	X
Result Access	X	X	X	X
Result Collecting	X	X	X	X
Frontend Web				
Scenario Creation		X	X	X
Scenario List		X	X	X
Scenario Execution		X	X	X
Result Presentation		X	X	X
ESB Configuration		X		
Consumer				
Scenario Reception and Execution	X	X	X	X
Result Generation and send back	X	X	X	X
Producer				
Accept Request and handling	X	X	X	X
Parameters handling	X	X	X	X
Other Coverage Areas				

8.1 Test Strategy and Objectives

The details of test strategies and objectives are described in Software Test Description document:

https://docs.google.com/document/d/1CFz5yyk_TStAz-1SibwWVM_y94ejwwoWwtrkZ9GvqQA/edit

8.2 Test Topologies

8.2.1 Configuration 1 : Hard coded dummies communicating without Mule

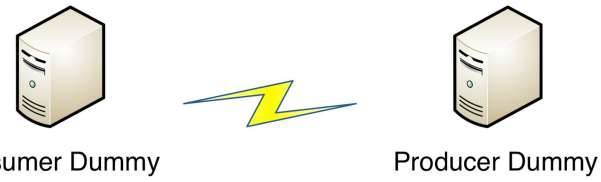
8.2.2 Configuration 2 : Hard coded dummies with Mule in between

8.2.3 Configuration 3 : Configurable dummies with Mule in between

8.2.4 Configuration 4 : Configurable dummies connected to the message broker

8.2.4.1 Diagram

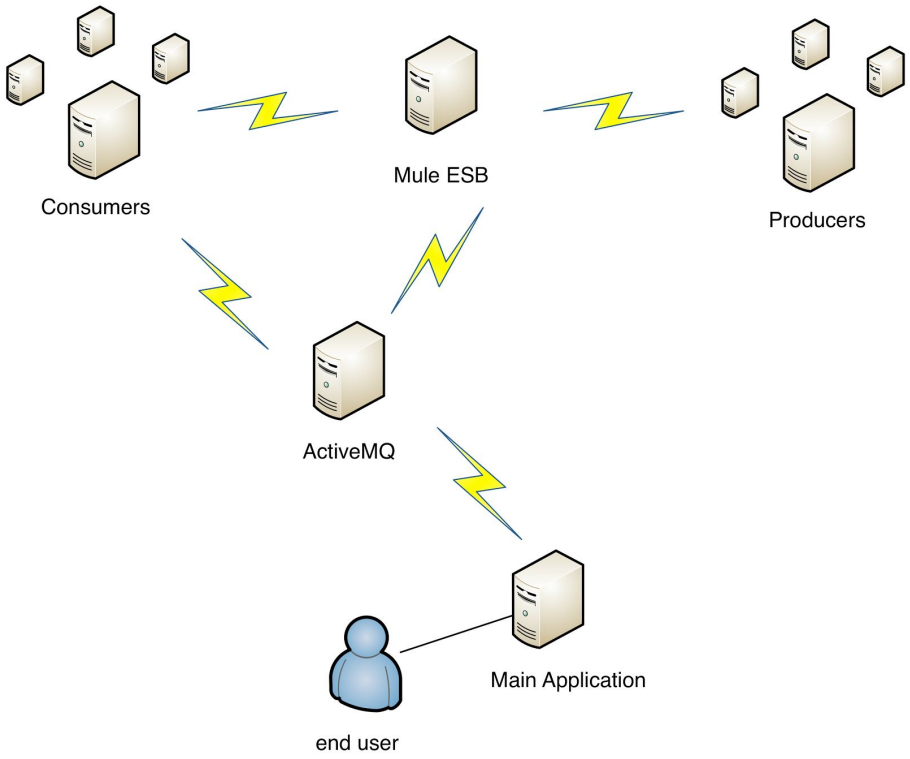
Configuration 1 : Hard coded dummies communicating without Mule



Configuration 2 : Hard coded dummies with Mule in between

Configuration 3 : Configurable dummies with Mule in between

Configuration 4 : Configurable dummies connected to the message broker



8.2.4.2 Machine Details

Machine Name	HW architecture	Number of processors	Processor speed	Memory	Disk space (free /used)	OS version	Kernel version	Key Software	Role of machine in the test configuration
Producer	Proxmox VM	1	3699.998 MHz	512MB	4GB	Ubuntu 15.04	Linux 2.6.32-39-pve	Java	To test
Consumer	Proxmox VM	1	3699.998 MHz	512MB	4GB	Ubuntu 15.04	Linux 2.6.32-39-pve	Java	To test
MainApp	Proxmox VM	1	3699.998 MHz	512MB	4GB	Ubuntu 15.04	Linux 2.6.32-39-pve	Java, Apache TomEE	To test
ActiveMQ	Proxmox VM	1	3699.998 MHz	512MB	10GB	Ubuntu 15.04	Linux 2.6.32-39-pve	ActiveMQ	To test
MuleESB	Proxmox VM	1	3699.998 MHz	1GB	4GB	Ubuntu 14.04.3 LTS	Linux 2.6.32-39-pve	MuleESB	To test

8.3 SVT Milestone Testing

All the information related to the testing is included in the Software Test Dossier

8.4 Test Management

8.4.1 *Defect Management*

We will report defects & bugs using JIRA. Any bugs will be added to the current sprint and shall be resolved before the end of the sprint.

8.4.2 *Test Tracking and Reporting Procedures*

The test tracking will be done using Jira. We used a custom workflow that makes sure every new feature is tested as it is completed, and before its issue is closed on Jira. Any issue shall be validated by another developer before being closed.

9. Tools, Techniques, and Methodologies

In terms of tools, we have decided to use SonarQube to have a continuous inspection of code quality. SonarQube is a software that offers reports on code duplication, code coverage, complexity, and unit tests, and can inform the user about potential bugs in the code.

We decided to apply a code review strategy; the reviewer should be someone different from who wrote it and has no knowledge of the context of the code. He is to see if the code written meets the standards that we set as quality code, and if the output is the same as we expected.

10. Configuration Management

To manage the different configurations of the software, we will use a git repository, hosted in INSA. We decided that each and every commit message should be precise enough to understand what has changed. Any issue or conflict regarding the git repository shall be reported to the Quality Manager.

11. Dependencies

This section is not applicable for this project.

12. Initial Issues, Risks and Mitigation Plans

Issue description and impact on SVT (a situation, problem, or an activity that has happened or is happening which impacts upon the approved Project Plan)	**Owner** (person or group with responsibility to resolve the issue)	**Plans to address**
Ryan will leave France before the end of the project	Ryan Shipp	We made sure that he can access Stash & JIRA from outside of the INSA. We also planned skype meetings.

Risk Description (a potential event or future situation that might adversely affect a project)	**Probability of Risk occurrence (H, M, L)**	**Description of Impact**	**Risk Mitigation Plans** (actions that will be taken to reduce the likelihood of the occurrence of the identified risk)
Wrong Time Estimation	H	Some key requirements may not be completed when the project ends	Time estimation will be done by the whole team rather than by one specific developer.
Slow Learning Process	H	Some key requirements may not be completed when the project ends	Technology choices will be made at the start of the project to start the learning phase as soon as possible.
Lack of testing	M	The project will not work as expected	We will provide a test plan.
Failure to identify failure	H	The client is not fully satisfied with the solution	We validate our design & system requirements with two presentations.