Software Test Description

Abstract:

This document provides a representation of ARYA Testbench test strategies and test plan.

Keywords:

ARYA, ESB, SOA, Testbench

Approved:

Authors :	Project Manager	Product Owner
Aurélien Tamas-Leloup	Yes	
Yanchao Wang		
Yuanbo Wang		
Alexandre Vey		
Ryan Shipp		

Revision History

Version	Date	Author	Change Description
1.0	12/30/2015	Yuanbo Wang	draft
1.1	01/04/2016	Yuanbo Wang	added test strategies
1.2	01/06/2016	Yuanbo Wang	added test cases for validation
1.3	01/20/2016	Yuanbo Wang	added unit test cases
1.4	01/25/2016	Yuanbo Wang	STD document finished
1.5	01/27/2016	A Tamas-Leloup	review

Software Test Description

Project : ARYA Testbench

CONTENTS

1 Introduction

1.1 Purpose

This document is a Test Plan for the project **ARYA Testbench**. Its purpose is to specify the required activities, and provide the information necessary in order to organize and perform the test activities successfully.

1.2 Scope

The ARYA Testbench project includes three main components: main application, producer application and consumer application.

The component called **main application** communicates with the user, processing his request and displaying the results. It also stores the information about the scenarios and communicates with the nodes using the message broker.

Both **producer** and **consumer application** are distributed. They encapsulate the dummies that will exchange datas through the ESB.

We decided to organize the test phases in 3 main phases : Validation Test, Integration Test and Unit Test. The target of these test phases is to deliver a product with a good quality with all available features.

1.3 Reference documents

document	Version	Date	Remarks
SRS	V1.8	22/11/2015	
STD	V1.3	30/12/2015	

2 Test Strategy

As described above, we will perform tests in 3 phases, which are respectively:

- Unit tests to evaluate the state of all testable functions during development phase.
- Integration to test the interfaces between different modules.
- Validation to test functional and some technical requirements

If we have constraints to run tests, these constraints will be described in test cases. We could develop tools to facilitate tests especially to test the connection between components and ESB, and the communication between the components and message broker.

Integration and Validation phase will run after the internal delivery of software, before the final delivery to customer. These phases will be done in development environment or integration development and they will be run commonly.

In the context of this project Validation phase is the same thing as user acceptance. We will invite a user to try ARYA Testbench and tell if our product is easy enough for new users.

We could start the integration and validation phase when the quality level of unit test is advanced enough. These phases could be postponed or run partially if the test is difficult to launch.

The coverage requirement/tests have to be complete. All defects have to be tracked.

2.1 Test Scenarios

- Unit tests

Unit tests are expected for every incremental part in the development of our product during each Sprint. Unit tests should validate each unitary function and make sure that they work without any unexpected behaviour.

The acceptance criteria for each logical test case should be discussed explicitly during the Sprint meeting, and will be listed in the following test case section. Unit tests are designed and launched using JUnit, and automated by Maven. Unit tests related to web applications will be launched by Selenium.

- Integration tests

Integration tests should validate the communication between every component in the ARYA Testbench. Integration tests will be launched after unit tests, and for system testing every requirement should be respected.

- Validation tests

The validation tests should validate the system with a functional and technical vision. All requirements described in SRS document should be validated.

3 Test Cases for Validation

3.1 Test Strategies and Test Techniques

Every requirement described in the SRS document should be tested, which include four parts: Manage scenarios, View results, Execute a scenario, Node management.

Non-functional requirements and other requirements such as internationalisation and accessibility should also be included in the tests. The main objective of the phase is that test coverage is the highest possible. We decide not to do regression tests.

3.2 Test Environment

For our test plan, specific hardware interfaces will not be considered. The player used to build the system will be able to deal with the different hardware components. The system shall be executed with any PC operating system such as Windows or Linux operating system.

3.3 Test Cases

Test case identifier	SRS identifier	REQUIREMENT TEXT / DATA	INPUT	EXPECTED OUTPUT

AR-ST-MS-01	AR-SR-MS-01	The system shall include a page that displays all scenarios	user open page	a list of scenarios is displayed to user
AR-ST-MS-02	AR-SR-MS-02	The system shall be able to retrieve all scenarios from the storage	user clicks on the retrieve scenario button	a list of scenarios are is displayed to user
AR-ST-MS-03	AR-SR-MS-03	The system shall allow the user to select a scenario from the list	user clicks on one option of the scenario list	a scenario item is shown as “chosen”
AR-ST-MS-04	AR-SR-MS-04	The system shall provide a page to see one specific scenario and all the related informations	user clicks on the scenario detail page	related information about the scenario is displayed in this page
AR-ST-MS-05	AR-SR-MS-05	The system shall allow the user to initiate a scenario creation sequence	user clicks on create scenario button	a form is presented to create a new scenario, when the page is submitted, a scenario is created
AR-ST-MS-06	AR-SR-MS-06	The system shall check that a provided scenario is correct, if it is then display the list of scenario. If not then display an error page.	user completes the form with incorrect scenario configuration s	the page displays that an error has occured
AR-ST-MS-07	AR-SR-MS-07	After an edition or a creation, the scenario shall be stored/updated by the system	user edited or created a scenario	the page reloads the new scenario configuration
AR-ST-MS-09	AR-SR-MS-09	The system shall allow the user to delete a scenario	user clicks on delete scenario button	the deleted scenario is removed from the page
AR-ST-VR-01	AR-SR-VR-01	The system shall use a graphing/visualisation library to generate graphs for the result data	user display a page	a graph that shows the results is displayed
AR-ST-VR-02	AR-SR-VR-02	The system shall have clearly labeled graphs that are easy for the user to understand	user hovers the cursor on the graph labels	details of the represented graph are displayed
AR-ST-VR-03	AR-SR-VR-03	The system shall display information about the scenario and execution beside the result	user display a page	information about the scenario and its execution is displayed
AR-ST-VR-04	AR-SR-VR-04	The system shall assign a unique timestamp to each execution of a scenario	user hovers the cursor on	details of the execution time are displayed

			the timestamp	
AR-ST-VR-05	AR-SR-VR-04	The system shall have a dropdown menu to choose between results, and be able to display any of them on the graph	user clicks on the dropdown menu	a list of options are available to choose from
AR-ST-ES-01	AR-SR-ES-01	The system shall provide the user with an interface to launch a scenario	user opens the scenario page	a button to launch the scenario is displayed
AR-ST-ES-02	AR-SR-ES-02	The system shall provide configure producers and consumers for a specific scenario	user clicks on configuration button	the form to change configuration parameters is displayed
AR-ST-ES-03	AR-SR-ES-03	The system shall provide the user to execute a scenario	user clicks on execute scenario button	the chosen scenario is executed
AR-ST-ES-04	AR-SR-ES-04	The system shall provide stop to send messages to the ESB during an execution	user clicks on stop or send message to ESB button	a message is sent or stopped from sending to ESB
AR-ST-ES-05	AR-SR-ES-05	The system shall be able ro retrive results at the end of the execution of a scenario	user opens page	the results of the retrieved execution are displayed
AR-ST-ES-06	AR-SR-ES-06	The system shall be able to ave the results at the end of the execution of a scenario	user clicks save results button	the results of the execution are saved
AR-ST-ES-07	AR-SR-ES-07	the system shall provide a view of the resultsto the user at the end of the execution of a scenario	user opens page	the results of the execution are displayed
AR-ST-NM-01	AR-SR-NM-01	The system shall provide the user with an interface to create a new Producer node	user clicks on create new producer button	a page to create a new producer is displayed
AR-ST-NM-02	AR-SR-NM-02	The system shall provide the user with an interface to connect a Producer node	user clicks on connect to producer button	a page to connect to producer is displayed
AR-ST-NM-03	AR-SR-NM-03	The system shall provide the user with an interface to disconnect a Producer node	user clicks on disconnect to producer button	a page to disconnect from producer is displayed
AR-ST-NM-04	AR-SR-NM-04	The system shall provide the user with an interface to create a new Consumer node	user clicks on create new consumer button	a page to create new consumer is displayed
AR-ST-NM-05	AR-SR-NM-05	The system shall provide the user with an interface to connect a Consumer node	user clicks on connect to consumer button	a page to connect to consumer is displayed

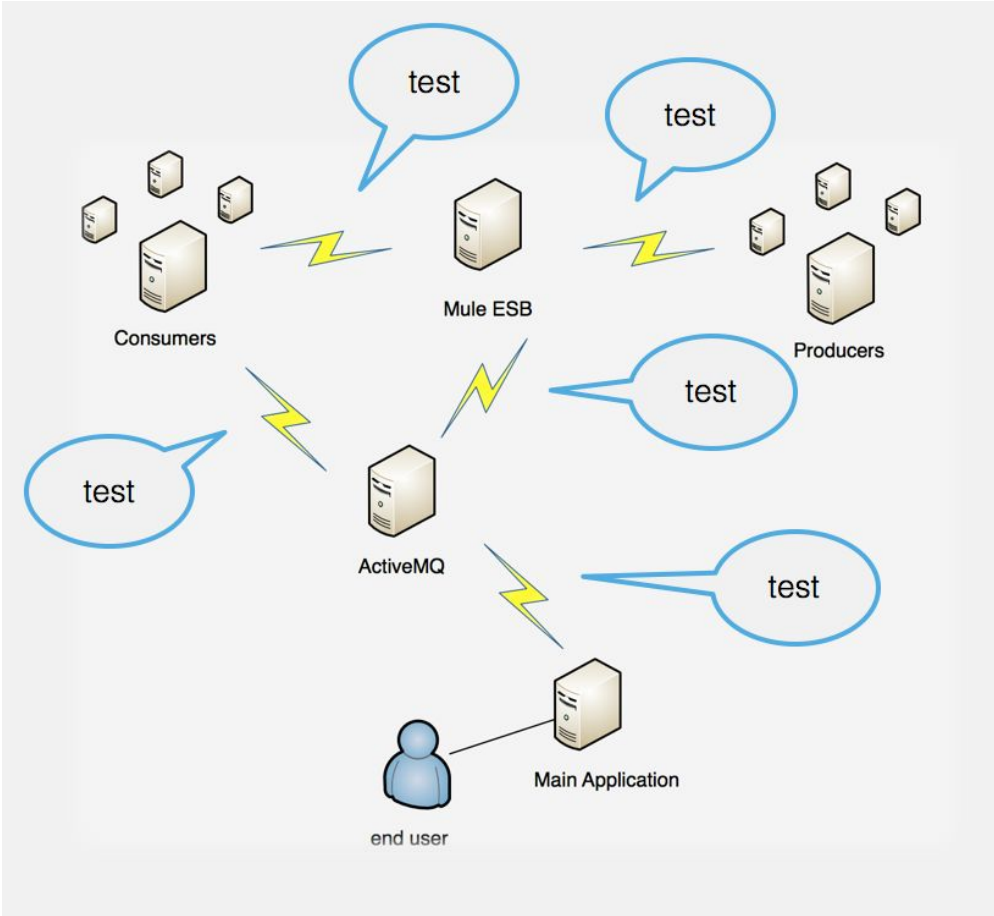
AR-ST-NM-06	AR-SR-NM-06	The system shall provide the user with an interface to disconnect a Consumer node	user clicks on disconnect to consumer button	a page to disconnect from consumer is displayed
ARYA-ST-PR-01	ARYA-PR-01	The system shall not affect the performance of the ESB when watching its behaviour	user starts watching the ESB behaviour	performance statistics of ESB stays the same
ARYA-ST-PR-02	ARYA-PR-02	The system shall not communicate with the ESB during a scenario	-	no communication is observed from the main application as source
AR-ST-OR-01	AR-OR-01	The system shall include internationalization	user chooses to change system language	the pages are displayed in chosen language
AR-ST-OR-02	AR-OR-02	The system shall include accessibility	user of disabilities chooses to display the page	the page is delivered in a way that the user gets full access to the content of the page

4 Test Cases for Integration

4.1 Test Strategies

The main part of these tests will be considered in white-box mode.

- **Process Tests** to check**:**
  - communication between main application and ActiveMQ
  - communication between ActiveMQ and Consumers
  - communication between ActiveMQ and Mule ESB
  - communication between Mule ESB and Producers



4.2 Test Environment

The integration tests should be done at each end of the communication, which can be Linux or Windows operating systems. A communication is considered as "tested and validated" if the tester is able to launch and receive expected results on the two endpoints of this communication.

4.3 Test Cases for process tests

TEST CASE IDENTIFIER	EXPLICATION TEXT	INPUT	EXPECTED OUTPUT
AR-ST-IT-01	Communication bewteen main application and ActiveMQ, at main application side	Store a message in the message broker	The message is retrieved on main application side
AR-ST-IT-02	Communication bewteen main application and ActiveMQ, at ActiveMQ side	Send a message from main application to ActiveMQ	The message is detected in the queue of ActiveMQ
AR-ST-IT-03	Communication bewteen Consumers and ActiveMQ, at ActiveMQ side	Send a message from the consumers to ActiveMQ	The message is detected in the queue of ActiveMQ
AR-ST-IT-04	Communication bewteen Consumers and ActiveMQ, at Consumers side	Store a message in the queue of ActiveMQ	The message is retrieved by consumers
AR-ST-IT-05	Communication bewteen Mule ESB and ActiveMQ, at ActiveMQ side	Send a message containing proxy list from the Mule ESB to ActiveMQ	The message is detected in the queue of ActiveMQ
AR-ST-IT-06	Communication bewteen Mule ESB and ActiveMQ, at Mule ESB side	Store a message in the queue of ActiveMQ	The message is retrieved by Mule ESB
AR-ST-IT-07	Communication bewteen Mule ESB and Producers, at Mule	Send a SOAP request to call the web service on the producers	The expected result is retrieved from the external web service

	ESB side		

5 Test Cases for Unit tests

5.1 Test Strategies and Test Techniques

During the unit testing phase, we have done a preliminary study on the applicability of unit test for each method. For the methods that are eligible to launch unit tests, another member other than the one who implemented the method is in charge of its test. In this case, every class should pass the unit test.

At the end of the unit testing phase, the test coverage should reach at least 80%, which is the expected value in our project. The process will be tested as a white box. Some tests could be run in debug mode.

5.2 Test Environment

The system shall be executed on any PC operating system such as Windows or Linux operating system, JUnit and Mockito are used as testing tools.

5.3 Test Cases for Producer

Test Case Number	Test Subject	Applicability	Input Value(s)	Expected Output Value(s)
AR-ST-UT-01	WebServiceImplementation.java	OK		message size is as expected

5.4 Test Cases for Consumer

Test Case Number	Test Subject	Input Value(s)	Expected Output Value(s)
AR-ST-UT-02	XMLGregorianDateConverter.java	a valid date	a date in Gregorian format
AR-ST-UT-03	ScenarioConverter.java	a scenario	a serialized string of the scenario
AR-ST-UT-04	ObjectFactory.java	-	an object created
AR-ST-UT-05	ResultConverter.java	the result	a serialized string of the result
AR-ST-UT-06	EventComparator.java	two events	the value of comparaison
AR-ST-UT-07	PointToPointComConsumer.java	a topic name	a creation that is connected to the queue
AR-ST-UT-08	MessagingGateway.java	a topic	a connection to the queue

5.5 Test Cases for Main Application

Test Case Number	Test Subject	Input Value(s)	Expected Output Value(s)
AR-ST-UT-09	ScenarioDAO.java	a scenario object	the fields of the scenario
AR-ST-UT-10	ResultDAO.java	a result object	the fields of the result
AR-ST-UT-11	ConfigDAO.java	a config object	the name of fields in the config object
AR-ST-UT-12	AppController.java	a scenario	the scenario is saved
AR-ST-UT-13	AppController.java	a scenario	the scenario is validated with configuration
AR-ST-UT-14	AppController.java	a scenario	the scenario is validated with correct timing
AR-ST-UT-15	AppController.java	a scenario	the scenario is validated with a single event

5.6 Test Cases for Mule Deployer

Test Case Number	Test Subject	Input Value(s)	Expected Output Value(s)
AR-ST-UT-16	MuleMain.java	a config object	a proxy list object
AR-ST-UT-17	ShellManager.java	a config object	the applications generated in mule folder

5.7 Traceability

Every test case is traceable and owns a link to one or several requirements described in SRS document. The traceability of test cases is shown in DOORS.