Politecnico di Milano

A.A. 2015-2016

Software Engineering 2

Test Plan

Version 0.1

Ivana Salerno

Alexis Rougnant

Daniel Vacca

January 21th 2016

Table of content

1.Introduction 3
  1.1.Revision History 3
  1.2.Purpose and Scope. 3
  1.3.List of Definitions and Abbreviations 3
  1.4.List of Reference Documents. 3
2.Integration Strategy 5
  2.1.Entry Criteria 5
  2.2.Elements to be integrated 5
  2.3.Integration Testing Strategy 6
  2.4.Sequence of Component/Function Integration 6
    2.4.1. Software Integration Sequence 6
    2.4.2.Subsystem Integration Sequence 7
3.Individual Steps and Test Description 9
  3.1.Description of integration steps 9
  3.2.Description of test procedures 23
4.Tools and Test Equipment Required 26
5.Program Stubs and Test Data 28
6.Appendix 30
6.1.System integration sequence 30
6.2.Subsystem integration sequence 32

1. Introduction

1.1. Revision History

This is the first version of this document so no revisions are included in this section.

1.2. Purpose and Scope.

This document is intended to present the Integration test plan for the MyTaxiService application, in the context of the project of the course Software Engineering 2.

This test plan gives the directions to validate the functionalities of the system when the components are incrementally joined. For that reason, this document is strongly related to the high level design presented in the Design Document, and thus does not provide specific details neither on the testing of the individual components nor on the whole system testing.

1.3. List of Definitions and Abbreviations

The following are abbreviations are used in the present document (and have not already been presented in any of the reference documents):

- IP: Integration procedure
- IS: Integration step
- ITP: Integration test plan

The following definitions are relevant for this document (and have not already been presented in any of the reference documents):

- Bottom up integration testing: it is an integration testing strategy which consists on performing the integration of the components from the lowermost to the uppermost. At each step, it requires the use of driver components to emulate the upper level parts of the system, where the invocations will come from.
- Driver (component): while testing a component (or a set of them), a driver is the tool used to emulate the part of the system that will perform invocations on the target portion of the testing.
- Stub (component): while testing a component (or a set of them), a stub is the tool used to emulate the part of the system that will receive invocations from the target portion of the testing.
- Top down integration testing: it is an integration testing strategy which consists on performing the integration of the components from the uppermost to the lowermost. At each step, it requires the use of stub components to emulate the lower level parts of the system, where the invocations will be sent to.

1.4. List of Reference Documents.

The following is the list of documents that are related to this Test plan, and that totally define its context:

õ MyTaxiService project AA 2015-2016 description

õ RASD for MyTaxiService, by Ivana Salerno, Alexis Rougnant and Daniel Vacca

õ DD for MyTaxiService, by Ivana Salerno, Alexis Rougnant and Daniel Vacca

õ Integration Test Plan description for project AA 2015-2016

õ Lecture slides on Validation and Verification.

The Integration Test Plan Example was used as a guideline to develop the structure of this document though it is not relevant to understand the domain problem.

2. Integration Strategy

In this section we present the strategy that will be used to guide the test plan. This includes the preconditions to execute it, the specific components that will be tested, the description and justification of the strategy, and a list of the concrete steps to be followed.

2.1. Entry Criteria

Within the context of the software development process, this test plan can only be executed when the following conditions hold:

õ A requirements specification of the domain problem must have been done. In our case, this is presented in the RASD.

õ A high level design specification of the solution must have been performed. This is included in the DD.

õ The source code of the implemented components that will be tested must be available.

õ Code inspection activities are recommended to be executed on the source code of the components, even though it is not entirely mandatory.

õ The components to be tested must have successfully passed the unit testing phase.

2.2. Elements to be integrated

The following list of components to be tested is based on the components presented in the Component view section of the Design Document:

õ MTSPassengerWebView

õ MTSPassengerWebController

õ MTSPassengerMobileView

õ MTSPassengerMobileController

õ MTSPassengerNotificationListener

õ MTSTaxiDriverMobileView

õ MTSTaxiDriverMobileController

õ MTSTaxiDriverMobileNotificationListener

õ MTSModel

õ MTS_DB

õ MTSIntegration

The components WebBrowser, MapsServer, EmailServer and MilanoGovernment are also included in this test plan even though they are external systems. This is because the integration testing is intended to validate the functionalities of the system when the different components are put together.

For the component MTSModel we will also test the following subcomponents:

õ PassengerModel

õ TaxiDriverModel

õ RequestManager

õ ReservationManager

õ SharingEngine

õ QueueManager

õ DataManager

2.3. Integration Testing Strategy

The execution of the test plan will follow a two-phase top-down strategy guided by critical components. As its name suggests, this strategy selects a critical component in the middle of the hierarchy of the system and considers executes two phases of top-down testing:

1) In the first phase we execute the testing rooted in the critical component. The lower half of the system is tested this way.
2) In the second phase, the upper half of the system (which has not been tested yet) is validated, by integrating components only until the point where the critical component is found. That component is now considered as a leaf of the testing process.

It usually happens that the most complex components are the ones which produce the biggest amount and most difficult errors. With this strategy we pretend to found them earlier and reduce in this way the impact on the project.

A bottom-up testing would also have been suitable, but the top-down alternative will produce earlier prototypes that can be used for user requirements validation.

2.4. Sequence of Component/Function Integration

Once that we have presented the strategy to guide the testing process and the specific parts of the system to be validated, we continue by listing the concrete steps that are to be followed to execute this test plan

2.4.1. Software Integration Sequence

All the components presented in the Component view of the Design Document can be considered as subsystems of the MyTaxiService application. Nevertheless we only provided a deeper view only for MTSModel, so we list integration steps only for its subcomponents.

For applying the two-phase top-down strategy we take the RequestManager as critical component. Initially we set PassengerModel and TaxiDriverModel as drivers, and the rest of the components as stubs. Then we incrementally add the real components to the integration:

Integration Test Step ID	Involved components	
IS1-T1	RequestManger, ReservationManager	Phase 1
IS1-T2	ReservationManager, RequestManager	
IS2-T1	RequestManager, SharingEngine	
IS2-T2	SharingEngine, RequestManager	
IS3-T1	TaxiDriverModel, RequestManager	
IS3-T2	RequestManager, TaxiDriverModel	
IS4-T1	RequestManager , QueueManager	
IS4-T2	ShringEngine , QueueManager	
IS4-T3	TaxiDriverModel , QueueManager	
IS5-T1	PassengerModel , RequestManager	Phase 2
IS6-T1	ReservationManager , DataManager	
IS6-T2	PassengerModel , DataManager	
IS6-T3	TaxiDriverModel, DataManager	

2.4.2. Subsystem Integration Sequence

For the general system we apply again the two-phase top-down strategy with the MTSModel component as the critical one. Note that the external components, which are WebBrowser, MapServer, EmailServer and MilanoGovernment, are not implemented by this team project; however their integration might be tested anyway. The same is true for the components MTSNotifier and MTS_DB which might not completely be implemented by the development team.

Integration Test Step ID	Components involved	
IS7-T1	MTSModel, MTSIntegration	Phase 1
IS8-T1	MTSIntegration, MapsServer	
IS8-T2	MTSIntegration, EmailServer	
IS8-T3	MTSIntegration, MilanoGovernment	
IS9-T1	MTSModel, MTS_DB	
IS10-T1	MTSModel, MTSNotifier	
IS11-T1	MTSNotifier, PassengerNotificationsListener	
IS11-T2	MTSNotifier, TaxiDriverNotificationsListener	
IS12-T1	MTSTaxiDriverMobileView, MTSTaxiDriverMobileController	Phase 2
IS12-T1	MTSTaxiDriverMobileController, MTSModel	
IS13-T1	MTSPassengerMobileView, MTSPassengerMobileController	
IS13-T2	MTSPassengerMobileController, MTSModel	
IS14-T1	WebBrowser, MTSPassengerWebView	
IS14-T2	MTSPassengerWebView, MTSPassengerWebController	
IS14-T3	MTSPassengerWebController, MTSModel	

Note: You can find diagrams illustrating the integration sequence of the system and the subsystem in the appendix

3. Individual Steps and Test Description

In this section we provide the description of the integration steps defined in the section 2.4. Since the purpose of this document is not to describe detailed testing but presenting the general guidelines for it, the definitions of the input and output are given in terms of the sequence diagrams in the Design document. We expect that the specific testing protocols are built by inspecting those diagrams. The protocols defined in the unit testing might also be helpful since the functionalities to be tested are basically the same (here we want re-validate the same functionalities when the components are integrated).

3.1. Description of integration steps

ID	IS1-T1
Components involved	RequestManager → ReservationManage
Environmental conditions	RequestManager, ReservationManager components DataManager stub
Input description	Create typical RequestManager input derived from the interaction that the following sequence diagram exposed between the involved components: - Passenger Cancel Request - Passenger Get Request - Process Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS1-T2
Components involved	ReservationManager → RequestManager
Environmental conditions	RequestManager, ReservationManager components TaxiDriverModel, SharingEngine, QueueManager stubs
Input description	Create typical ReservationManager input derived from the interaction that the following sequence diagram exposed between the involved components: - Process Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS2-T1
Components involved	RequestManager → SharingEngine
Environmental conditions	RequestManager, SharingEngine components QueueManager stub
Input description	Create typical RequestManager input derived from the interaction that the following sequence diagram exposed between the involved components: - Passenger Cancel Request - Passenger Get request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS2-T2
Components involved	SharingEngine → RequestManager
Environmental conditions	RequestManager, SharingEngine components TaxiDriverModel, QueueManager stubs The test IS1 must have succeeded.
Input description	Create typical SharingEngine input derived from the interaction that the following sequence diagram exposed between the involved components : - Process Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS3-T1
Components involved	RequestManager → TaxiDriverModel
Environmental conditions	RequestManager , TaxiDriverModel components DataManager, QueueManager stubs
Input description	Create typical RequestManager input derived from the interaction that the following sequence diagram exposed between the involved components : - Driver Cancel Request - Find Taxi Driver - Passenger Cancel Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS3-T2
Components involved	TaxiDriverModel → RequestManager
Environmental conditions	RequestManager, TaxiDriverModel components QueueManager stub The tests IS1 and IS2 must have succeeded.
Input description	Create typical TaxiDriverModel input derived from the interaction that the following sequence diagram exposed between the involved components : - Driver Answer Request - Driver Cancel Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS4-T1
Components involved	RequestManager → QueueManager
Environmental conditions	RequestManager, QueueManager components
Input description	Create typical RequestManager input derived from the interaction that the following sequence diagram exposed between the involved components : - Driver Answer Request - Driver Cancel Request - Find Taxi Driver - Passenger Cancel Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS4-T2
Components involved	SharingEngine → QueueManager
Environmental conditions	RequestManager, QueueManager components The tests IS2 must have succeeded.
Input description	Create typical SharingEngine input derived from the interaction that the following sequence diagram exposed between the involved components : - Process Request
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS4-T3
Components involved	TaxiDriverModel → QueueManager
Environmental conditions	RequestManager, TaxiDriverModel component The tests IS3 must have succeeded.
Input description	Create typical TaxiDriverModel input derived from the interaction that the following sequence diagram exposed between the involved components : - Driver Set Availability - Driver Update Position
Output description	We expect that the invoked methods generate the output and actions that correspond to the input derived from the sequence diagram.
Observations	

ID	IS5-T1
Components involved	PassengerModel, RequestManager
Environmental conditions	The tests IS1 to IS4 must have succeeded. A stub for the DataManager component is used.
Input description	Create typical PassengerModel input derived from the interaction that the following sequence diagram exposed between the involved components: - Passenger Cancel Request - Passenger Get Request - Passenger Receive Request - Process Request
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagrams.
Observations	This test is supposed to validate the integration of PassengerModel with the RequestManager and the already tested part of the system.

ID	IS6-T1
Components involved	ReservationManager, DataManager
Environmental conditions	The tests IS1 to IS5 must have succeeded. A stub for the MTS_DB component is used.
Input description	Create typical ReservationManager input derived from the interaction that the following sequence diagram exposed between the involved components: - Passenger Cancel Request - Process Request
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagrams.
Observations	This test is supposed to validate the integration of ReservationManager with the DataManager and the already tested

	part of the system.

ID	IS6-T2
Components involved	PassengerModel, DataManager
Environmental conditions	The tests IS1 to IS6-T1 must have succeeded. A stub for the MTS_DB component is used.
Input description	Create typical PassenngerModel input derived from the interaction that the following sequence diagram exposed between the involved components: - Passenger Confirm Email - Passenger Create Account - Passenger Edit Account - Passenger Log In
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of PassengerModel with the DataManager and the already tested part of the system.

ID	IS6-T3
Components involved	TaxiDriverModel, DataManager
Environmental conditions	The tests IS1 to IS6-T2 must have succeeded. A stub for the MTS_DB component is used.
Input description	Create typical TaxiDriverModel input derived from the interaction that the following sequence diagram exposed between the involved components: - Driver create account - Driver Edit Account - Driver Login

Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of TaxiDriverModel with the DataManager and the already tested part of the system.

ID	IS7-T1
Components involved	MTSModel, MTSIntegration
Environmental conditions	The tests IS1 to IS6 must have succeeded. Stubs for the MapsServer, EmailServer and MilanoGovernment components are used. Drivers for the MTSControllers components are used.
Input description	Create typical MTSModel input derived from the interaction that the following sequence diagram exposed between the involved components: - Driver create account - Driver edit account - Passenger edit account - Passenger create account - Passenger receive request - Process request
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSModel with the MTSIntegration only.

ID	IS8-T1
Components involved	MTSIntegration, MapsServer
Environmental conditions	The tests IS1 to IS7 must have succeeded. Stubs for the EmailServer and MilanoGovernment components are used. We have actual access to the MapsServer external system.
Input description	Create typical MTSIntegration input derived from the interaction that the following sequence diagram exposed between the involved components: - Passenger receive request - Process request
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSIntegration with the external system MapsServer.

ID	IS8-T2
Components involved	MTSIntegration, EmailServer
Environmental conditions	The tests IS1 to IS8-T1 must have succeeded. A stub for the MilanoGovernment components is used. We have actual access to the EmailServer external system.
Input description	Create typical MTSIntegration input derived from the interaction that the following sequence diagram exposed between the involved components: - Driver create account - Driver edit account - Passenger edit account - Passenger create account
Output description	We expect that the invoked methods generate the outputs and

	actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSIntegration with the external system EmailServer.

ID	IS8-T3
Components involved	MTSIntegration, MilanoGovernment
Environmental conditions	The tests IS1 to IS8-T2 must have succeeded. We have actual access to the MilanoGovernment external system.
Input description	Create typical MTSIntegration input derived from the interaction that the following sequence diagram exposed between the involved components: - Driver create account
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSIntegration with the external system EmailServer.

ID	IS9-T1
Components involved	MTSModel, MTS_DB
Environmental conditions	It is possible to access the database MTS_DB of the system.
Input description	The input will be the typical data for the procedure required by MTSModel.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSModel with the MTS_DB and the already tested part of the system.

ID	IS10-T1
Components involved	MTSModel, MTSNotifier
Environmental conditions	The tests IS1 to IS8 must have succeeded. Stubs for the NotificationListeners components are used. Drivers for the MTSControllers components are used.
Input description	Create typical MTSModel input derived from the interaction that the following sequence diagram exposed between the involved components: - Driver answer request - Driver cancel request - Find taxi driver - Passenger cancel request
Output description	We expect that the invoked methods generate the outputs and actions that correspond to the input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSModel with the MTSNotifier and the already tested part of the system.

ID	IS11-T1
Components involved	MTSNotifier, PassengerNotificationsListener
Environmental conditions	
Input description	The input will be the typical data for the procedure required by MTSNotifier.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSNotifier with the PassengerNotificationsListener and the already tested part of the system.

ID	IS11-T2
Components involved	MTSNotifier, TaxiDriverNotificationsListener
Environmental conditions	
Input description	The input will be the typical data for the procedure required by MTSNotifier.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSNotifier with the TaxiDriverNotificationsListener and the already tested part of the system.

ID	IS12-T1
Components involved	MTSTaxiDriverMobileView, MTSTaxiDriverMobileController
Environmental conditions	IS11-T2 succeeded
Input description	The input will be the typical data for the procedure required by MTSTaxiDriverMobileView. It derived form the interaction that the following sequence diagram exposed between the involved components: - Driver accept request - Driver cancel request - Driver create account - Driver edit account - Driver login - Driver set availability
Output description	The output will be characterized by invoked methods and it match with the given input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSTaxiDriverMobileView with the MTSTaxiDriverMobileController and the already tested part of the system.

ID	IS12-T2
Components involved	MTSTaxiDriverMobileController, MTSModel
Environmental conditions	IS12-T1 succeeded
Input description	The input will be the typical data for the procedure required by MTSTaxiDriverMobileController.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSTaxiDriverMobileController with the MTSModel and the already tested part of the system.

ID	IS13-T1
Components involved	MTSPassengerMobileView, MTSPassengerMobileController
Environmental conditions	IS11-T1 succeeded
Input description	The input will be the typical data for the procedure required by MTSPassengerMobileView. It derives from the interaction that the following sequence diagram exposed between the involved components: - Passenger cancel request - Passenger create account - Passenger edit account - Passenger view requests - Passenger login - Passenger make request
Output description	The output will be characterized by invoked methods and it match with the given input derived from the sequence diagram.
Observations	This test is supposed to validate the integration of MTSPassengerMobileView with the MTSPassengerMobileController and the already tested part of the system.

ID	IS13-T2
Components involved	MTSPassengerMobileController, MTSModel
Environmental conditions	IS13-T1 succeeded
Input description	The input will be the typical data for the procedure required by MTSPassengerMobileController.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSPassengerMobileController with the MTSModel and the already tested part of the system.

ID	IS14-T1
Components involved	WebBrowser, MTSPassengerWebView
Environmental conditions	
Input description	The input will be the typical data for the procedure required by WebBrowser.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of WebBrowser with the MTSPassengerWebView and the already tested part of the system.

ID	IS14-T2
Components involved	MTSPassengerWebView, MTSPassengerWebController
Environmental conditions	IS14-T1 succeeded
Input description	The input will be the typical data for the procedure required by MTSPassengerWebView.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSPassengerWebView with the MTSPassengerWebController and

	the already tested part of the system.

ID	IS14-T3
Components involved	MTSPassengerWebController, MTSModel
Environmental conditions	IS14-T2 succeeded
Input description	The input will be the typical data for the procedure required by MTSPassengerWebController.
Output description	The output will be characterized by invoked methods and it match with the given input.
Observations	This test is supposed to validate the integration of MTSPassengerWebController with the MTSModel and the already tested part of the system.

3.2. Description of test procedures

Once we have described the test steps that make part of the integration plan, we now put them together into procedures accordingly to the functional role that each one of them has in the overall system.

We have defined 6 procedures (one of them split in two):

1) Business logic

Test Procedure ID	TP1-1
Related functionalities	This test procedure verifies whether the MTSModel : - handle TaxiDriver input - handle Passenger input
Chain of steps	Execute IS13-T1 to IS13-T2, after having executed IS1 to IS11.

2) External systems communication

Test Procedure ID	TP2-1
Related functionalities	This procedure is intended to verify the communication with external components through the MTSIntegration component. The procedure will allow to check that request to external components are working well and the output are coherent.
Chain of steps	Execute IS8-T1 to IS8-T2, after having executed IS7.

3) Data management

Test Procedure ID	TP3-1
Related functionalities	This procedure is intended to verify the access to the database.
Chain of steps	Execute IS9, after having executed IS1 to IS7.

4) Notifications

Test Procedure ID	TP4-1
Related functionalities	This procedure is intended to verify the communication with PassengerNotificationsListener and TaxiDriverNotificationsListener through the MTSNotifier component. The steps will allow to check that the listeners are receiving the right information and the communication is established.
Chain of steps	Execute IS11-T1 and IS11-T2, after having executed IS10.

5) Passenger functionalities

Test Procedure ID	TP5-1
Related functionalities	This procedure is intended to verify the functionalities offered to the Passenger through the web application. The steps will allow to verify the behavior of both graphical and logical components.
Chain of steps	Execute IS14-T1 to IS14-T3, after having executed IS1 to IS11.

Test Procedure ID	TP5-2
Related functionalities	This procedure is intended to verify the functionalities offered to the Passenger through the mobile application. The steps will allow to verify the behavior of both graphical and logical components.
Chain of steps	Execute IS13-T1 to IS13-T2, after having executed IS1 to IS11.

6) Taxi driver functionalities

Test Procedure ID	TP6-1
Related functionalities	This procedure is intended to verify the functionalities offered to the Taxi Driver through the mobile application. The steps will allow to verify the behavior of both graphical and logical component.
Chain of steps	Execute IS12-T1 and IS12-T2, after having executed from IS1 to IS11.

4. Tools and Test Equipment Required

In the Design Document we proposed a logical architecture for the MyTaxiService application, which is specific language independent. For that reason we will not provide here a detailed description of the tools that will be used for the integration testing but instead we briefly describe some tools that could be considered as candidates. Such options are based on the specific implementation language.

Language	**Tool**	**Tool Description**
PHP	*PHPUnit*	Efficient tool for unit testing and integration testing. Widespread, so has a lot of documentation. Supported in lots of IDE
	Laravel	PHP framework that is using PHPUnit as a backend and make easier the tests writing. Easy to use. Has a good documentation and an active community of users.
Android	*AndroidJUnitRunner*	JUnit 4-compatible test runner for Android
	Espresso	UI testing framework Suitable for functional UI testing within an app.
	UI Automator	UI testing framework Suitable for cross-app functional UI testing across system and installed apps.
	Monkey	This tool runs on your device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. Monkey tool can be used to stress-test applications that you are developing, in a random yet repeatable manner.
	Monkeyrunner	This testing system provides an API for writing programs that control an Android device from outside of Android code.
Java EE	*Arquillian*	It can be used to test all the components which are deployed inside an application container. The selection of Aqruillian is based in the following main benefits: • Portability: the designed tests can be used for various application containers • Usability: the tests can be executed either from the IDE or directly from the build tool • Extensibility: the test can be supported in already existing or implemented testing

		frameworks Arquillian is also capable of covering the details of a real execution environment, which also leads to refine and enrich the testing environment. This will allow to obtain “more real” testing. Such advantages are reached thanks to the following features: • Custom management of the container • Suitable centralized files to describe the test and its dependencies • Executing the test inside or against real monitored containers (which makes debugging easier) • Organized classpath controlling • Client-server debugging More information and details of this tools can be found on the official website[1].

Note that these are only suggestions. When further implementation decisions are made, different testing tools can be selected. However, such selection must consider the environmental conditions that have been described for the testing process in this document, as well as the derived input and output for each test step.

[1] http://arquillian.org/

5. Program Stubs and Test Data

Some environmental conditions for the test steps have already been mentioned in past sections. Here we list the conditions that specifically concern the drivers, stubs, test data and possibly additional components that must be available to carry out this integration test plan.

Stubs:

According to what has already been exposed in the past section, this integration test will be supported by stubs for the following components:

- DataManager
- TaxiDriverModel
- SharingEngine
- QueueManager
- MTS_DB
- MapsServer
- EmailServer
- MilanoGovernment,
- PassengerNotificationListener
- TaxiDriverNotificationListener

Drivers:

According to what has already been exposed in the past section, this integration test will require the development of drivers for the following components:

- MTSTaxiDriverMobileController
- MTSPassengerWebController
- MTSPassengerMobileController

Test data:

There should be enough instances of the entities to cover all the test cases and steps. Again, the data that was used in the unit testing can be reused for this purpose. We specially expect to have data of:

õ Definition of zones

õ Unregistered taxi drivers in the Milano Government

õ Registered taxi driver

õ Registered passengers

õ Normal, sharing and reservation requests

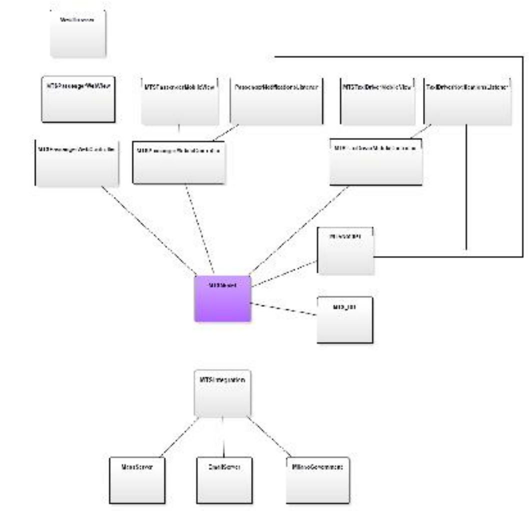
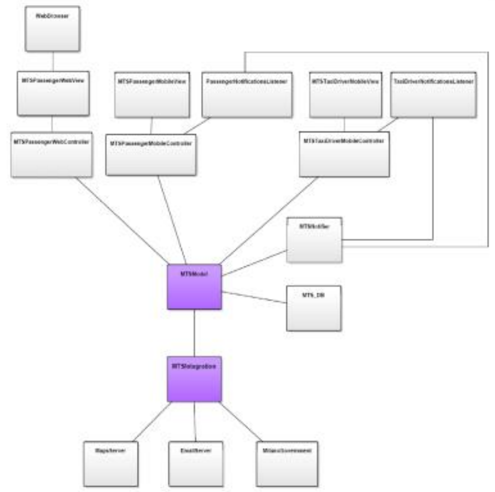
Additional components:

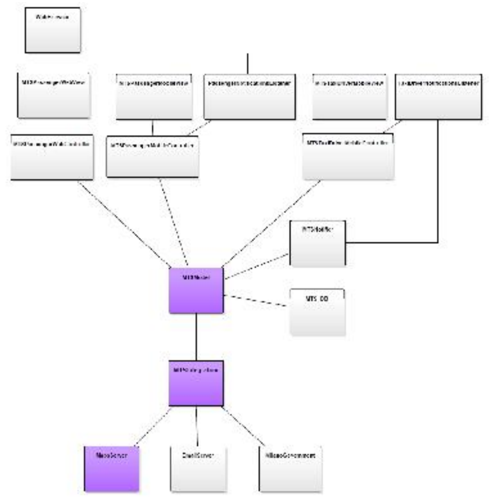
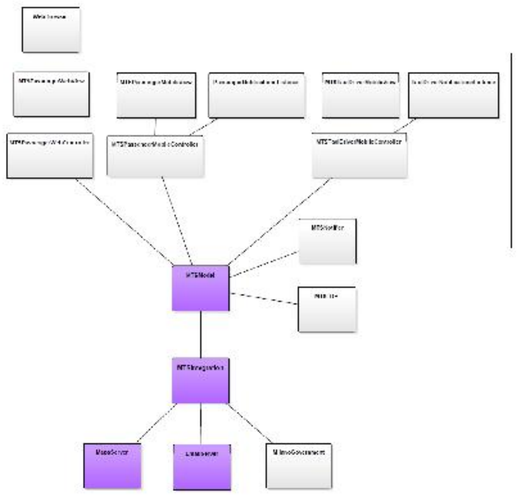
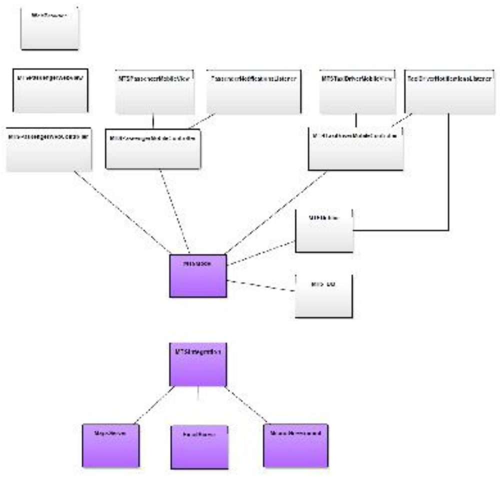
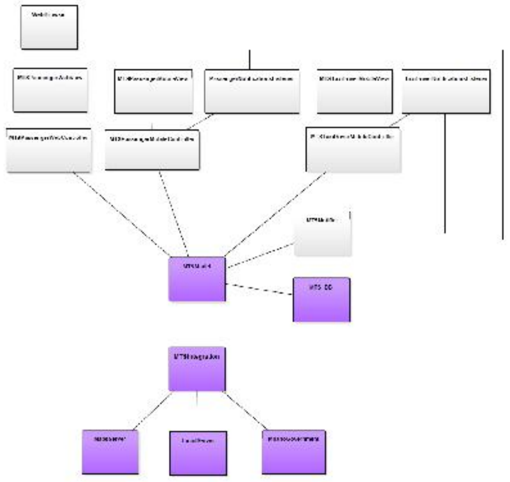
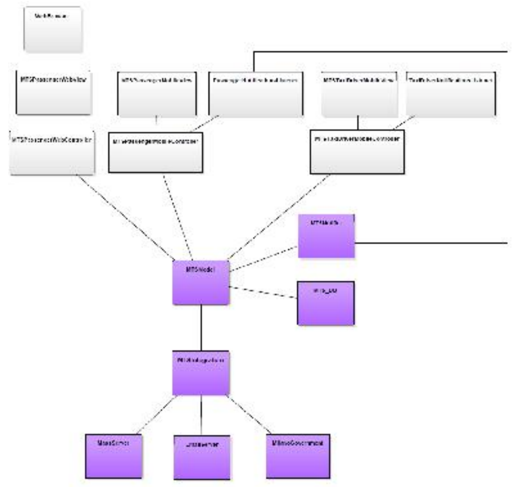
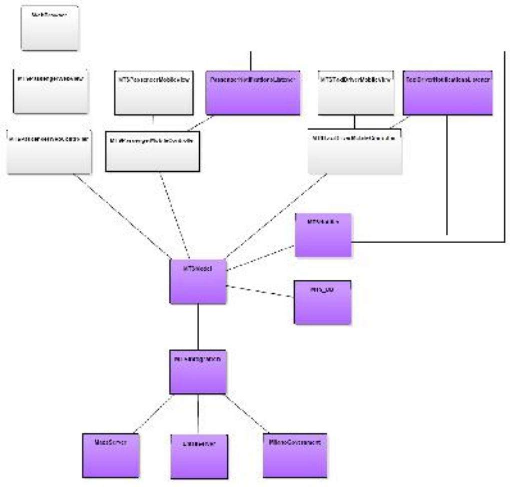
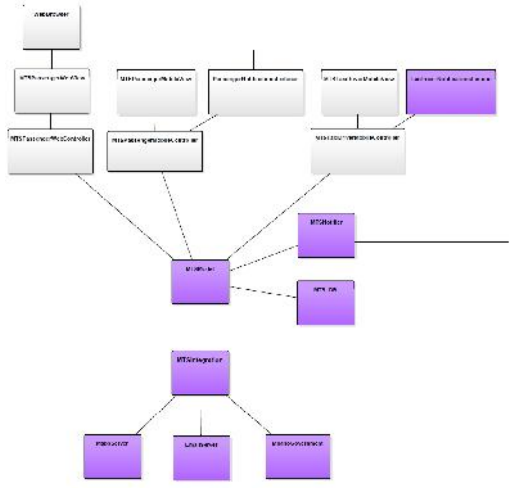
The present proposal of the architecture includes interaction with some external systems (MilaanoGovernment, EmailServer and MapsServer). We expect to have access to them in each one of the steps which they are involved in (see section 3.1). The same is true for the WebBrowser.

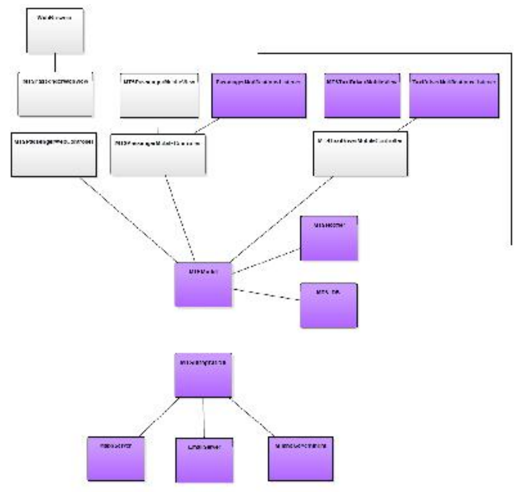
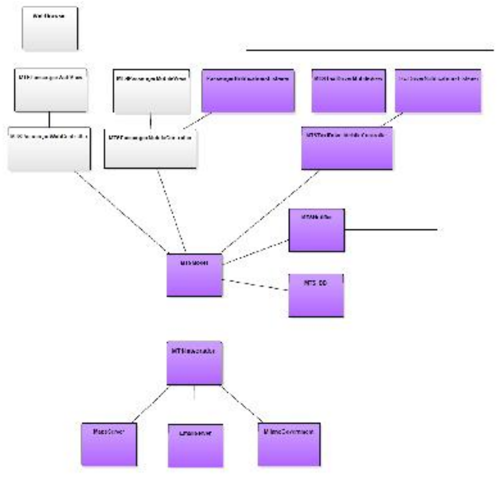
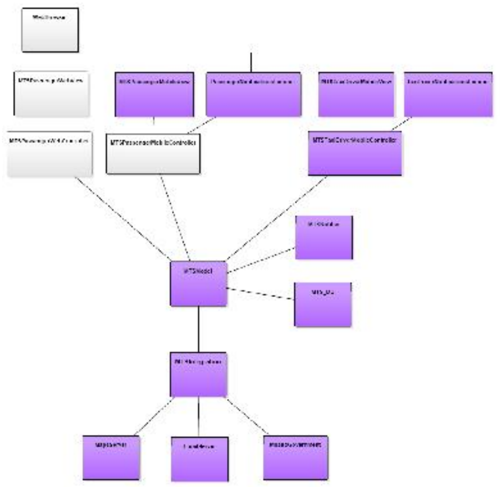
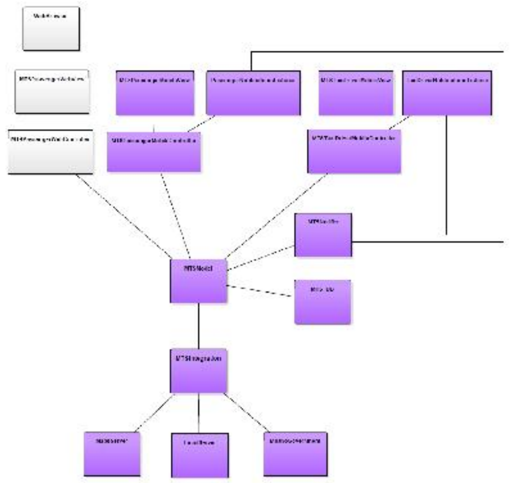
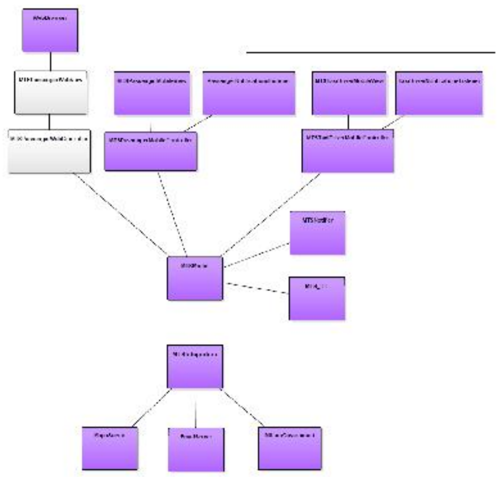
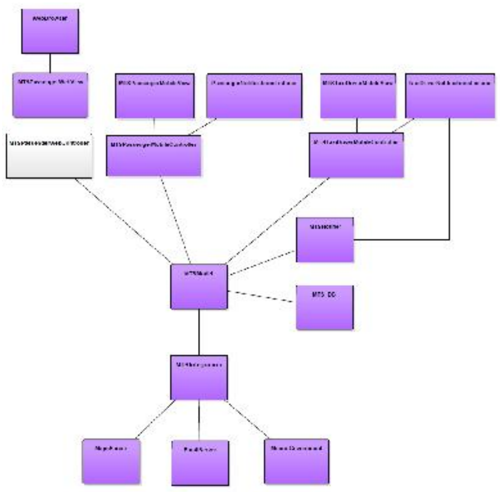
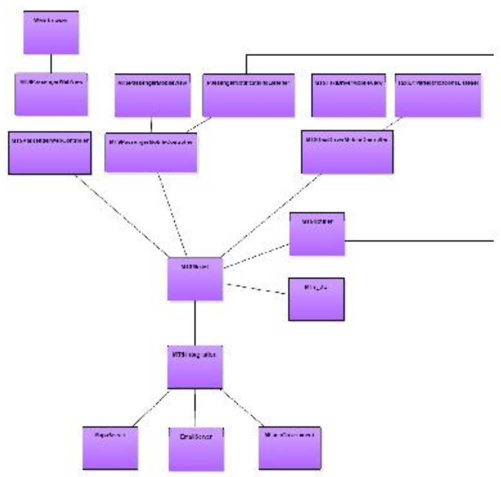
In most cases, the notification components and persistence systems are not developed from scratch for these type of applications; instead, some already implemented alternatives are acquired and adapted as desired. In the case that such happens for MTS_DB and MTSNotifier, we expect that those components are ready to be used by the beginning of this integration plan (i. e. they have been completely adapted).

6. Appendix

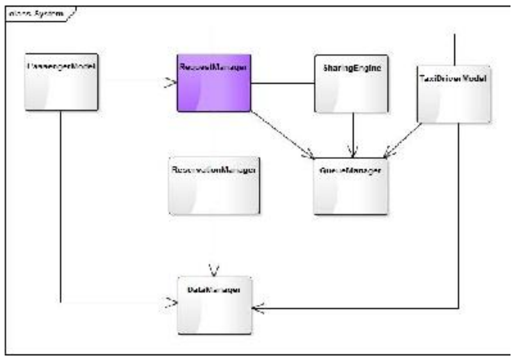
6.1. System integration sequence

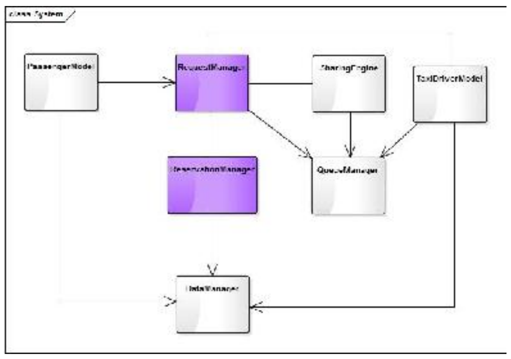



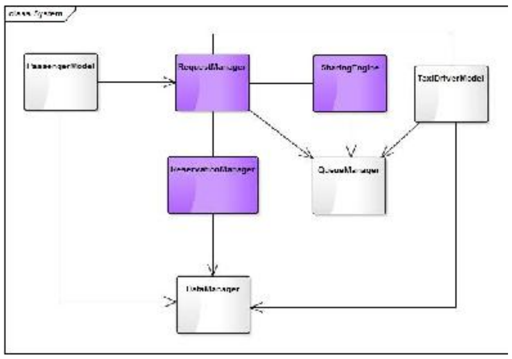








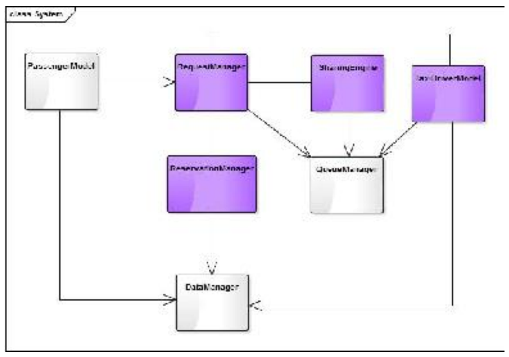


MTS_DB

MTSPassengerMobileController
MTSTaxiDriverMobileController
MTSModel

MTSModel
MTSIntegration


MTS_DB

MTSIntegration
Email Server

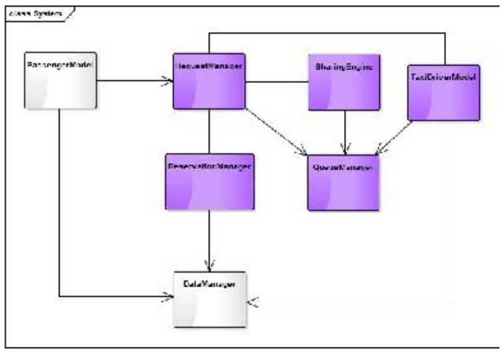
6.2. Subsystem integration sequence

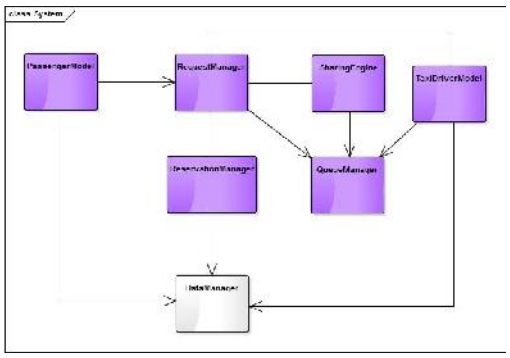

class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager

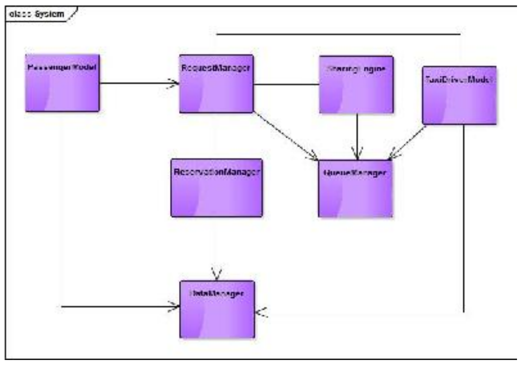

class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager


class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager


class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager


class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager


class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager


class System
PassengerModel
RequestManager
SharingEngine
TaxiDriverModel
ReservationManager
QueueManager
DataManager