

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](#__RefHeading___Toc861_1707713022)

[1.1. Revision History. 3](#__RefHeading___Toc863_1707713022)

[1.2.Purpose and Scope. 3](#__RefHeading___Toc865_1707713022)

[1.3.List of Definitions and Abbreviations 3](#__RefHeading___Toc867_1707713022)

[1.4. List of Reference Documents. 3](#__RefHeading___Toc869_1707713022)

[1.4.1.The project description 3](#__RefHeading___Toc871_1707713022)

[1.4.2.The RASD 3](#__RefHeading___Toc873_1707713022)

[1.4.3.The Design document 3](#__RefHeading___Toc875_1707713022)

[1.4.4.The documentation of any tool you plan to use for testing 3](#__RefHeading___Toc877_1707713022)

[2.Integration Strategy 3](#__RefHeading___Toc879_1707713022)

[2.1.Entry Criteria. 3](#__RefHeading___Toc881_1707713022)

[2.2.Elements to be Integrated. 3](#__RefHeading___Toc883_1707713022)

[2.3. Integration Testing Strategy. 3](#__RefHeading___Toc885_1707713022)

[2.4.Sequence of Component/Function Integration 4](#__RefHeading___Toc887_1707713022)

[2.4.1.Software Integration Sequence. 4](#__RefHeading___Toc889_1707713022)

[2.4.2.Subsystem Integration Sequence. 4](#__RefHeading___Toc891_1707713022)

[3.Individual Steps and Test Description 4](#__RefHeading___Toc893_1707713022)

[4.Tools and Test Equipment Required 4](#__RefHeading___Toc895_1707713022)

[5.Program Stubs and Test Data 4](#__RefHeading___Toc897_1707713022)

# Introduction

## Revision History

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

## 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.

## 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:
* Driver (component):
* Stub (component):
* Top down integration testing:

## List of Reference Documents.

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

* MyTaxiService project AA 2015-2016 description
* RASD for MyTaxiService, by Ivana Salerno, Alexis Rougnant and Daniel Vacca
* DD for MyTaxiService, by Ivana Salerno, Alexis Rougnant and Daniel Vacca
* Integration Test Plan description for project AA 2015-2016
* **The documentation of any tool you plan to use for testing**

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.

# 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.

## Entry Criteria

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

* A requirements specification of the domain problem must have been done. In our case, this is presented in the RASD.
* A high level design specification of the solution must have been performed. This is included in the DD.
* The source code of the implemented components that will be tested must be available.
* Code inspection activities are recommended to be executed on the source code of the components, even though it is not entirely mandatory.
* The components to be tested must have successfully passed the unit testing phase.

## 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:

* MTSPassengerWebView
* MTSPassengerWebController
* MTSPassengerMobileView
* MTSPassengerMobileController
* MTSPassengerNotificationListener
* MTSTaxiDriverMobileView
* MTSTaxiDriverMobileController
* MTSTaxiDriverMobileNotificationListener
* MTSModel
* MTS\_DB
* 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:

* PassengerModel
* TaxiDriverModel
* RequestManager
* ReservationManager
* SharingEngine
* QueueManager
* DataManager

## 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 intermediate prototypes that can be used for requirements validation.

## 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

### 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 |  |

### 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, MTSNotifier |
| IS10-T1 | MTSModel, MTS\_DB |
| 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 |

# 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).

## Description of integration steps

PREVIOUS STEPS IN ENVIRONMENTAL?

|  |  |
| --- | --- |
| ID | IS1-T1 |
| Components involved | RequestManager → ReservationManager |
| 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 | This test is not following the Top-Down strategy |

|  |  |
| --- | --- |
| 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  I1 |
| 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 | This test is not following the Top-Down strategy |

|  |  |
| --- | --- |
| 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  I1,12 |
| 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 | This test is not following the Top-Down strategy |

|  |  |
| --- | --- |
| 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  I2 |
| 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  I3 |
| 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, 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. |

## 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. MTSModel, MTSIntegration, EmailServer, MapsServer, MilanoGovernment
2. MTSModel, MTS\_DB
3. MTSModel, MTSNotifier, MTSListeners
4. Passenger

|  |  |
| --- | --- |
| Test Procedure ID | TP6-1 |
| 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 | TP6-2 |
| 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. |

1. Taxi driver: MTSDriverAppView, MTSDriverAppCont, MTSModel

# 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.

TOOLS FOR PHP, JAVA, RUBY, ANDROID. MANUAL TESTING??

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.

# 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:**

**Drivers:**

**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:

* Definition of zones
* Unregistered taxi drivers in the Milano Government
* Registered taxi driver
* Registered passengers
* 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).