

Politecnico di Milano
Scuola di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Computer Science and Engineering
A.Y. 2015-2016



Software Engineering 2 Project
“myTaxiService”
Integration Test Plan

January 18, 2016

Principal Adviser: Prof. **Di Nitto**

Authors:

Davide Fisicaro 854043

Gianmarco Giummarra 852667

Salvatore Ferrigno 850130

Contents

1	Introduction	1
1.1	Revision History	1
1.2	Purpose and Scope	1
1.3	List of Definitions, Acronyms and Abbreviations	1
1.4	List of Reference Documents	1
2	Integration Strategy	2
2.1	Entry Criteria	2
2.2	Elements to be Integrated	2
2.3	Integration Testing Strategy	3
2.4	Sequence of Component/Function Integration	3
2.4.1	Software Integration Sequence	3
2.4.2	Subsystem Integration Sequence	6
3	Individual Steps and Test Description	7
3.1	Guest Component subsystem (S1)	7
3.2	Customer Component subsystem	8
3.2.1	Customer Profile Manager (S2)	8
3.2.2	Customer Logout Manager (S3)	8
3.2.3	Immediate Call Manager (S4)	9
3.2.4	Reservation Call Manager (S5)	10
3.3	Taxi Driver Component subsystem	11
3.3.1	Taxi Driver Profile Manager (S6)	11
3.3.2	Incoming Call Manager (S7)	12
3.3.3	Assistance Call Manager (S8)	13
3.3.4	Taxi Driver Logout Manager (S9)	14
4	Tools and Test Equipments Required	15
5	Program Stubs and Test Data Required	16

1 Introduction

1.1 Revision History

1.2 Purpose and Scope

The aim of this document is to provide an integration test plan.

The integration test detects the bugs not discovered during unit tests, focusing the attention on a group of components to be tested.

Two or more components are integrated and analyzed, and when bugs are detected more components can be added to fix the errors.

In more details, interfaces and interactions between all the components of the myTaxiService system will be tested following the approaches that will be described below.

The scope can be found in the RASD, section 1.3.

1.3 List of Definitions, Acronyms and Abbreviations

- **RASD:** Requirement Analysis and Specification Document;
- **Component:** The smallest part of the system, which implements a specific function in the business application; it's the first part to be unit-tested;
- **Subsystem:** Each set of Components, or Lower Level Subsystem that interact each others in order to accomplish business operations;
- **First Level Subsystem:** Each Subsystem which composed only of Components;

1.4 List of Reference Documents

List all reference documents, for instance:

- The RASD
- The Design document

2 Integration Strategy

2.1 Entry Criteria

Some criteria that have to be met before starting with the integration testing are listed below:

- All of the atomic functions have to be unit-tested
- Every integration between components have to be defined
- Every needed interface have to be defined

2.2 Elements to be Integrated

The elements to be integrated are, according to the strategy explained in the section 2.3 of this document, the single components of the system, as described in the Design Document, *Section 2.3*.

To give a brief summary of these “atomic” elements of the system, they are listed below:

- Authentication manager
- Customer profile manager
- Immediate call manager
- Reservation call manager
- Taxi driver profile manager
- Incoming call manager
- Assistance call manager
- Data component
- Payment manager
- Notification manager
- Guest component
- Customer component
- Taxi driver component

2.3 Integration Testing Strategy

The chosen integration testing approach for this Integration Plan is the bottom-up approach. The reason of this choice is that by following this strategy it's possible to begin testing the system from the little components that compose every single part and continue the integration at a higher level for each step. This justifies also the presence of the first entry criterium in the point 2.1 of this document; in the rare case that some functions or some part of code at the moment of the integration testing is not developed yet, it should be possible to build a Stub in order to go on with the testing of the whole single functions at the low level. For the same reason, in the integration testing for different components, should be possible to build Drivers that allow to testing the integration when an interface is not completely developed yet.

2.4 Sequence of Component/Function Integration

2.4.1 Software Integration Sequence

For each subsystem are listed the available operations for the specific context.

In this section are listed the First Level Subsystems, that are basically nine, one for each class of operations. They are below divided into three main groups: Guest subsystem (S1), Customer subsystems (S2 to S5) and Taxi Driver subsystems (S6 to S9), depending on the actor involved in the considered class of operations.

S1. Guest subsystem

In this component the Authentication and Registration operations are listed.

- Registration of a new user;
- Customer Authentication in the system, if already logged;
- Taxi Driver authentication;

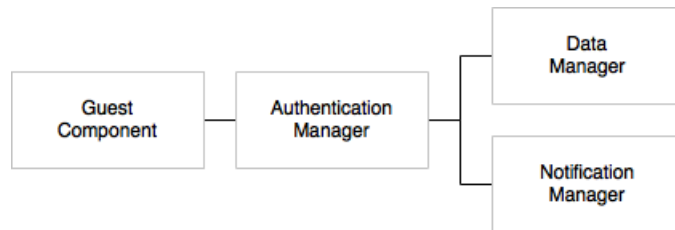


Figure 2.1: Integration schema for the Guest Component environment

S2. Customer Profile Management subsystem

- Email changing
- Password changing
- Password retrieval
- Payment methods changing
- Ride history showing
- Sharing mode activation

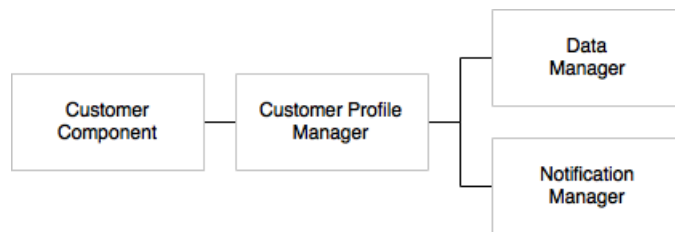


Figure 2.2: Integration schema for the Customer Profile management

S3. Customer Logout Management subsystem

- Customer Logout



Figure 2.3: Integration schema for the Customer Logout management

S4. Customer Immediate Call Management subsystem

- Immediate Call requesting

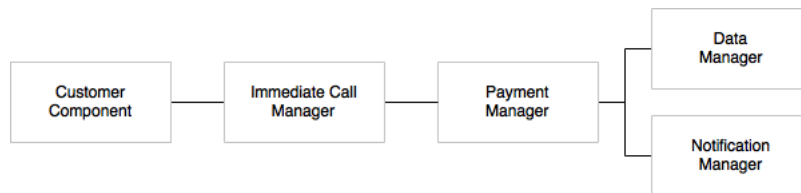


Figure 2.4: Integration schema for the Immediate Call by a Customer

S5. Customer Reservation Call Management subsystem

- Reservation Call requesting

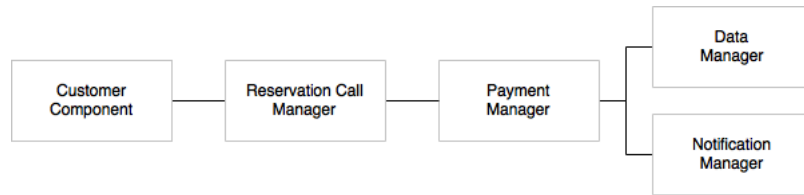


Figure 2.5: Integration schema for the Reservation Call by a Customer

S6. Taxi Driver Profile Management subsystem

- Password changing
- Password retrieval
- Ride history showing
- Given rating showing

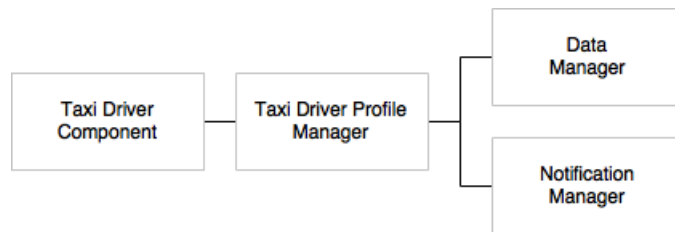


Figure 2.6: Integration schema for the Profile management

S7. Taxi Driver Incoming Call Management subsystem

- Call accepting
- Call declining

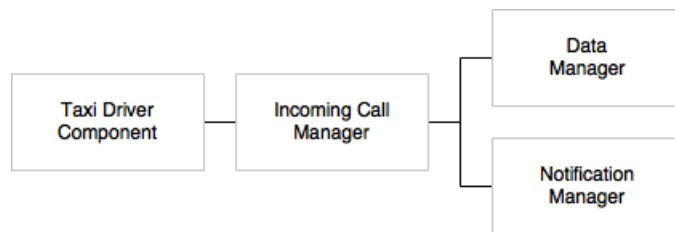


Figure 2.7: Integration schema for the Taxi Driver Incoming Call management

S8. Taxi Driver Assistance Call Management subsystem

- Accident call
- System failure call
- Generic failure call

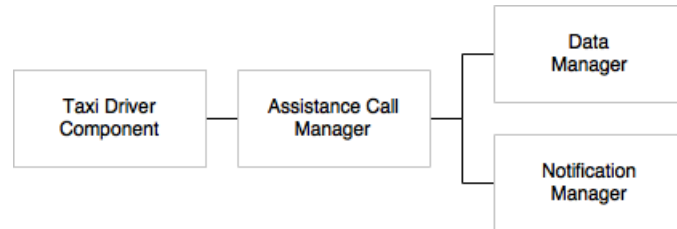


Figure 2.8: Integration schema for the Taxi Driver Component Assistance Call management

S9. Taxi Driver Logout Management subsystem

- Taxi Driver Logout



Figure 2.9: Integration schema for the Taxi Driver Logout management

2.4.2 Subsystem Integration Sequence

3 Individual Steps and Test Description

In the tables showed in the following sub-sections, the execution order is described in the *Requirements* column, while the *Tested Components* describes the components relevant to each step, in relation between each others, and the *Needed Drivers* column refers to the drivers used to substitute the involved components that are not object of the related test step.

At every step a driver is substituted with the corresponding component: in this way, when a problem is detected the solution has to be searched in the interaction with the latest added component.

Since that the system has to be cross-platform, each integration test has to be performed in every supported platform, i.e. Android, iOS, Windows Phone.

3.1 Guest Component subsystem (S1)

The integration steps are three: I11, I12, I13 and the tested operations between the components are the ones described in the *authenticationManagement* and *registrationManagement* interfaces.

In the Sequence diagram shown in the *Figure 2.4* of the *Section 2.5.1* of the Design Document can be found a more detailed representation of the components interaction.

- **I11:** the tested interactions are the ones between Guest Component and Authentication Manager, using two drivers for the other components of the subsystem; the test is passed when the Login and Registration operations are correctly performed.
- **I12:** the Data Manager driver is substituted with real component, while a driver still substitutes the Notification Manager component; the test is passed when all the operations involved in the Data storage and retrieval, such as the storage of the data of a new user, are correctly performed.
- **I13:** the whole subsystem is tested without using any driver; the test is passed when all the notifications, for all the supported operations, are dispatched to the correct receiver.

Index	Requirements	Tested Components	Needed Drivers
I11	—	Guest Component, Authentication Manager	Data Manager, Notification Manager
I12	I11	Authentication Manager, Guest Component, Data Manager	Notification Manager
I13	I12	Guest Component, Authentication Manager, Data Manager, Notification Manager	—

Table 3.1: Integration sequence table for the Guest subsystem testing

3.2 Customer Component subsystem

The integration steps are three: I11, I12, I13 and the tested operations between the components are the ones described in the *customerProfileManagement* interface.

In the *Section 2.6.3* of the Design Document can be found a more detailed list of the methods relevant to this component context, i.e. *changeEmail(String oldEmail, String newEmail, String password)*.

- **I21:** the tested interactions are the ones between Customer Component and Customer Profile Manager, using two drivers for the other components of the subsystem; the test is passed when each method is correctly invoked and each parameter is passed in the correct order and form.
- **I22:** the Data Manager driver is substituted with real component, while a driver still substitutes the Notification Manager component; the test is passed when all the operations involved in the Data storage, delete, modify and retrieval are correctly performed.
- **I23:** the whole subsystem is tested without using any driver; the test is passed when all the notifications, for all the supported operations, are dispatched to the correct receiver.

3.2.1 Customer Profile Manager (S2)

Index	Requirements	Tested Components	Needed Drivers
I21	—	Customer Component, Customer Profile Manager	Data Manager, Notification Manager
I22	I21	Customer Profile Manager, Customer Component, Data Manager	Notification Manager
I23	I22	Customer Component, Customer Profile Manager, Data Manager, Notification Manager	—

Table 3.2: Integration sequence table for the Customer Profile Management testing

3.2.2 Customer Logout Manager (S3)

The integration test for the Logout manager in the Customer context checks if the logout operation ends correctly, managing and terminating the current session in the correct way.

The method that will be tested in this case is the *logout()* method shown in the *Section 2.6.1* of the Design Document.

- **I24:** the tested interaction is the one between the Customer Component and the Authentication Manager; the test is passed when the logout operation terminates without not expected behaviours.

Index	Requirements	Tested Components	Needed Drivers
I24	—	Customer Component, Authentication Manager	—

Table 3.3: Integration sequence table for the Customer Profile Management testing

3.2.3 Immediate Call Manager (S4)

The integration steps are four: I31, I32, I33, I34 and the tested operations between the components are the ones described in the *immediateCallManagement* interface.

In the *Section 2.6.5* of the Design Document can be found a more detailed list of the methods relevant to this component context, i.e. *sendImmediateRequest(...)*.

- **I31:** the tested interaction is the one between the Customer Component and the Immediate Call Manager, using three drivers for the other components of the subsystem; the test is passed if the request is sent using the right parameters, according to the business logic described in the Functional Requirements of the RASD, *Section 3.1*. The different types of Immediate Call operation have to be tested, such as Private or Shared Immediate Call.
- **I32:** the Payment Manager Component substitutes the related driver used in the previous testing sequence; the test is passed when the verification and validation of the payment information happens correctly and without unexpected behaviours and that the payment information for the current Call are correctly forwarded to the external banking system.
- **I33:** the Data Manager Component is used instead of the related driver used in the previous testing sequence; the test is passed when all the operations involved in the Data storage, delete, modify and retrieval are correctly performed.
- **I34:** finally, the Notification Manager Component substitutes the last driver used in the previous testing sequence, in order to test the whole subsystem; the test is passed when all the notifications, for all the supported operations, are dispatched to the correct receiver.

This current subsystem testing case needs some drivers that substitutes components of an external subsystem, successively tested and integrated with this one:

- Incoming Call Manager
- Taxi Driver Manager

Index	Requirements	Tested Components	Needed Drivers
I31	—	Customer Component, Immediate Call Manager	Data Manager, Payment Manager, Notification Manager
I32	I31	Immediate Call Manager, Customer Component, Payment Manager	Notification Manager, Data Manager
I33	I32	Customer Component, Immediate Call Manager, Payment Manager, Data Manager	Notification Manager
I34	I33	Customer Component, Immediate Call Manager, Payment Manager, Data Manager, Notification Manager	—

Table 3.4: Integration sequence table for the Immediate Call Management testing

3.2.4 Reservation Call Manager (S5)

As for the Immediate Call testing case, the integrations steps are four: I41, I42, I43, I44 and the tested operations between the components are the ones described in the *reservationCallManagement* interface.

In the *Section 2.6.6* of the Design Document can be found a more detailed list of the methods relevant to this component context, i.e. *sendReservationRequest(...)*.

- **I41:** the tested interaction is the one between the Customer Component and the Reservation Call Manager, using three drivers for the other components of the subsystem; the test is passed if the request is sent using the right parameters, according to the business logic described in the Functional Requirements of the RASD, *Section 3.1*. The different types of Reservation Call operation have to be tested, such as Private or Shared Reservation Call.
- **I42:** the Payment Manager Component substitutes the related driver used in the previous testing sequence; in the same way for the Immediate Call Manager testing in the previous section, the test is passed when the payment information are correctly sent and received and no unexpected behaviours happen and the payment information for the current Call are correctly forwarded to the external banking system.

- **I43:** the Data Manager Component is used instead of the related driver used in the previous testing sequence; the test is passed when all the operations involved in the Data storage and retrieval are correctly performed.
- **I44:** finally, the Notification Manager Component substitutes the last driver used in the previous testing sequence; the test is passed when all the notifications, for all the supported operations, are dispatched to the correct receiver.

This current subsystem testing case needs some drivers that substitutes components of an external subsystem, successively tested and integrated with this one:

- Incoming Call Manager
- Taxi Driver Manager

Index	Requirements	Tested Components	Needed Drivers
I41	—	Customer Component, Reservation Call Manager	Data Manager, Payment Manager, Notification Manager
I42	I41	Reservation Call Manager, Customer Component, Payment Manager	Notification Manager, Data Manager
I43	I42	Customer Component, Reservation Call Manager, Payment Manager, Data Manager	Notification Manager
I44	I43	Customer Component, Reservation Call Manager, Payment Manager, Data Manager, Notification Manager	—

Table 3.5: Integration sequence table for the Reservation Call Management testing

3.3 Taxi Driver Component subsystem

3.3.1 Taxi Driver Profile Manager (S6)

The integration steps are three: I51, I52, I53 and the tested operations between the components are the ones described in the *taxiDriverProfileManagement* interface.

The integration testing happens in a similar way to the Customer subsystem Profile Manager.

In the *Section 2.6.4* of the Design Document can be found a more detailed list of the methods relevant to this component context, i.e. *retrievePassword(String email)*.

- **I51:** the tested interactions are the ones between Taxi Driver Component and Taxi Driver Profile Manager, using two drivers for the other components of the subsystem; the test is passed when each method is correctly invoked and each parameter is passed in the correct order and form.
- **I52:** the Data Manager driver is substituted with the real component, while a driver still substitutes the Notification Manager component; the test is passed when all the operations involved in the Data storage, delete, modify and retrieval are correctly performed.
- **I53:** the whole subsystem is tested without using any driver; the test is passed when all the notifications, for all the supported operations, are dispatched to the correct receiver.

Index	Requirements	Tested Components	Needed Drivers
I51	—	Taxi Driver Component, Taxi Driver Profile Manager	Data Manager, Notification Manager
I52	I51	Taxi Driver Profile Manager, Taxi Driver Component, Data Manager	Notification Manager
I53	I52	Taxi Driver Component, Taxi Driver Profile Manager, Data Manager, Notification Manager	—

Table 3.6: Integration sequence table for the Taxi Driver Profile Management testing

3.3.2 Incoming Call Manager (S7)

The integration steps are three: I61, I62, I63 and the tested operations between the components are the ones described in the *incomingCallManagement* interface.

In the *Section 2.6.7* of the Design Document can be found a more detailed list of the methods relevant to this component context, i.e. *acceptCall()* and *declineCall()*.

- **I61:** the tested interactions are the ones between Incoming Call Manager and Notification Manager, using two internal drivers for the other components of the subsystem; the test is passed when the interactions between the two components are correctly performed, without any unexpected behaviours.

- **I62:** the Taxi Driver Component driver is substituted with the real component, while the last internal driver still substitutes the Data Manager component; the test is passed when all the types of call that a Taxi Driver could receive are correctly notified to the relative Component.
- **I63:** the whole subsystem is tested without using any internal driver; the test is passed when all the data storage, delete, modify and retrieval operations are correctly performed.

This current subsystem testing case needs some drivers that substitutes components of an external subsystem, successively tested and integrated with this one:

- Immediate Call Manager
- Customer Manager

Index	Requirements	Tested Components	Needed Drivers
I61	—	Incoming Call Manager, Notification Manager	Data Manager, Taxi Driver Manager
I62	I61	Incoming Call Manager, Notification Manager, Taxi Driver Component	Data Manager
I63	I62	Incoming Call Manager, Notification Manager, Data Manager, Taxi Driver Component	—

Table 3.7: Integration sequence table for the Incoming Call Management testing

3.3.3 Assistance Call Manager (S8)

The integration steps are three: I71, I72, I73 and the tested operations between the components are the ones described in the *taxiDriverAssistanceCallManagement* interface.

In the *Section 2.6.8* of the Design Document can be found a more detailed list of the methods relevant to this component context, i.e. *signalAccident(String comment)*, *signalGenericFault(String comment)*.

- **I71:** the tested interactions are the ones between Taxi Driver Component and Assistance Call Manager, using two drivers for the other components of the subsystem; the test is passed when each type of assistance call are correctly sent by the Taxi Driver Component and received and managed by the Assistance Call Manager, without any unexpected behaviours.

- **I72:** the Notification Manager driver is substituted with the real component, while a driver still substitutes the Data Manager component; the test is passed when the Notification Manager correctly receives the request from the Assistance Call Manager and sends the Notification to the Taxi Driver, after the request management.
- **I73:** the whole subsystem is tested without using any driver; the test is passed when all the data storage, delete, modify and retrieval operations are correctly performed.

Index	Requirements	Tested Components	Needed Drivers
I71	—	Taxi Driver Component, Assistance Call Manager	Data Manager, Notification Manager
I72	I71	Assistance Call Manager, Taxi Driver Component, Notification Manager	Data Manager
I73	I72	Taxi Driver Component, Assistance Call Manager, Notification Manager, Data Manager	—

Table 3.8: Integration sequence table for the Assistance Call Management testing

3.3.4 Taxi Driver Logout Manager (S9)

The integration test for the Logout manager in the Taxi Driver context checks if the logout operation ends correctly, managing and terminating the current session in the correct way.

The method that will be tested in this case is the *logout()* method shown in the Section 2.6.1 of the Design Document.

- **I74:** the tested interaction is the one between the Taxi Driver Component and the Authentication Manager; in a similar way to the Customer context, the test is passed when the logout operation terminates without not expected behaviours.

Index	Requirements	Tested Components	Needed Drivers
I71	—	Taxi Driver Component, Assistance Call Manager	Data Manager, Notification Manager

Table 3.9: Integration sequence table for the Taxi Driver Logout Management testing

4 Tools and Test Equipments Required

Identify all tools and test equipment needed to accomplish the integration. Refer to the tools presented during the lectures. Explain why and how you are going to use them. Note that you may also use manual testing for some part. Consider manual testing as one of the possible tools you have available.

5 Program Stubs and Test Data Required

Based on the testing strategy and test design, identify any program stubs or special test data required for each integration step.