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"**
# **I**ntegration **T**est **P**lan

January 19, 2016

Principal Adviser: Prof. **Di Nitto**

Authors:
**Davide Fisicaro** 854043
**Gianmarco Giummarra** 852667
**Salvatore Ferrigno** 850130

# Contents

# 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 if no bugs are detected more components can be added to the testing sequence.

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 Subsystems that interact between each others in order to accomplish business operations;
- **First Level Subsystem**: Each Subsystem composed only of Components;
- **Guest**: a person that has never signed up to the system, so without a personal profile. He can only sign up;
- **Customer**: a person already logged to the system that wants to call a taxi, or a person who is actually using the taxi service;
- **Taxi Driver**: a person who drive a taxi, with his own taxi driver ID provided by the government that hired him;

## 1.4 List of Reference Documents

- RASD
- 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 Components have to be unit-tested. In this way every internal operation such as the ones described with the algorithms and shown in the sequence diagrams in the Design Document, must have passed the white box test;
- 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*.

The list below gives a brief summary of these "atomic" elements of the system:

- 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. For the same reason, in the integration testing for different components, should be possible to build Drivers that allow to test 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 have been divided into three main groups: Guest subsystem (S1), Customer subsystems (S2 to S5) and Taxi Driver subsystems (S6 to S9), depending on which actor is involved in the considered class of operations.

**S1. Guest subsystem**
Operations to be tested:

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

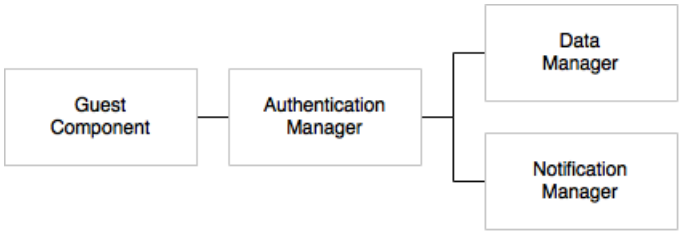


*Figure 2.1*: Integration schema for the Guest Component environment

**S2. Customer Profile Management subsystem**
Operations to be tested:

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

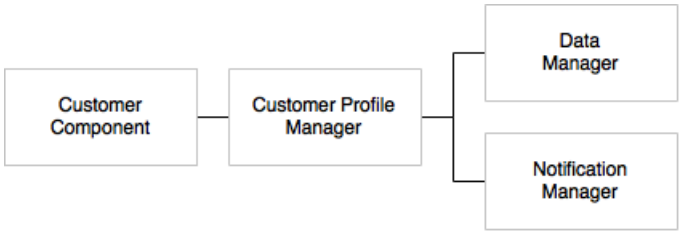


*Figure 2.2*: Integration schema for the Customer Profile management

**S3. Customer Logout Management subsystem**
Operation to be tested:

- Customer Logout

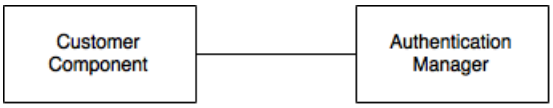


*Figure 2.3*: Integration schema for the Customer Logout management

**S4. Customer Immediate Call Management subsystem**
Operation to be tested:

- Immediate Call requesting

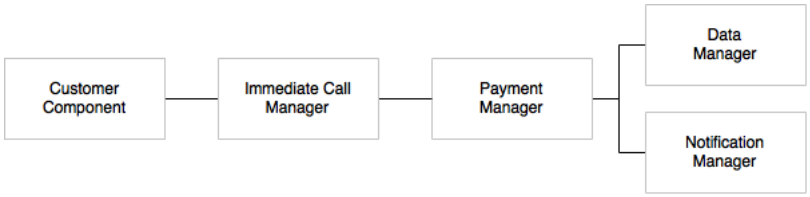


*Figure 2.4*: Integration schema for the Immediate Call by a Customer

**S5. Customer Reservation Call Management subsystem**
Operation to be tested:

- Reservation Call requesting

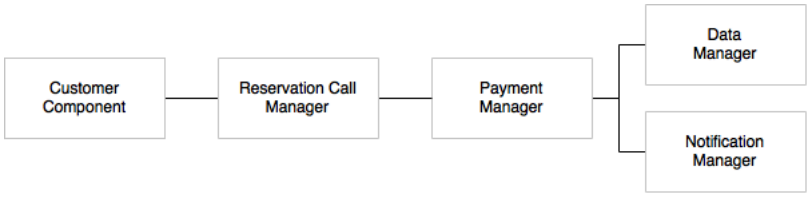


*Figure 2.5*: Integration schema for the Reservation Call by a Customer

**S6. Taxi Driver Profile Management subsystem**
Operations to be tested:

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

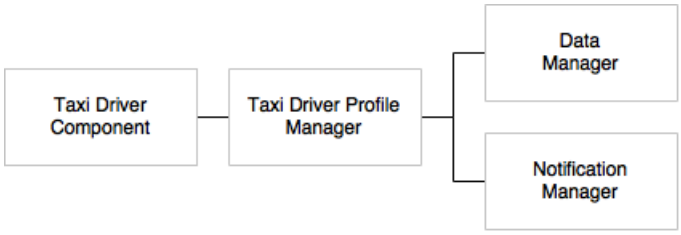


*Figure 2.6*: Integration schema for the Profile management

**S7. Taxi Driver Incoming Call Management subsystem**
Operations to be tested:

- Call accepting
- Call declining

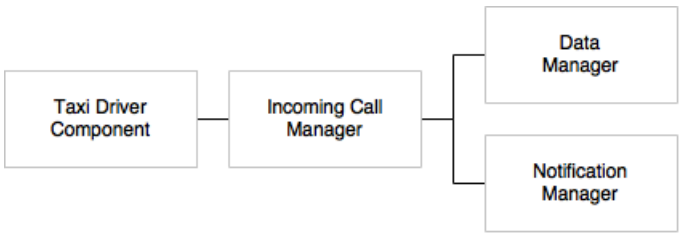


*Figure 2.7*: Integration schema for the Taxi Driver Incoming Call management

**S8. Taxi Driver Assistance Call Management subsystem**
Operations to be tested:

- Accident call
- System failure call
- Generic failure call

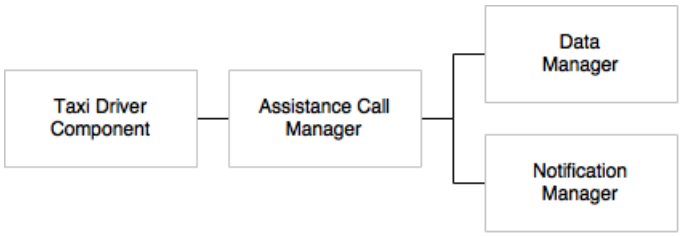


*Figure 2.*8: Integration schema for the Taxi Driver Component Assistance Call management

**S9. Taxi Driver Logout Management subsystem**
Operation to be tested:

- Taxi Driver Logout

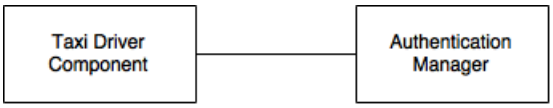


*Figure 2.*9: Integration schema for the Taxi Driver Logout management

### 2.4.2 Subsystem Integration Sequence

In this section are described the sequences for the First Level Subsystem integration. Each one of the following sequences describes the interaction between two First Level Subsystems.

**S10. Immediate Call Management subsystem**

This Second Level Subsystem is composed by the two Subsystems that manage the Immediate Call operations by the point of view of both the Customer and the Taxi Driver.

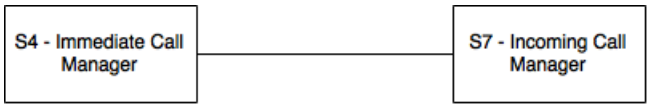


*Figure 2.*10: Integration schema for the Immediate Call management

**S11. Reservation Call Management subsystem**

This Second Level Subsystem represents the composition of the two Subsystems that manage the Immediate Call operations by the point of view of both the Customer and the Taxi Driver.

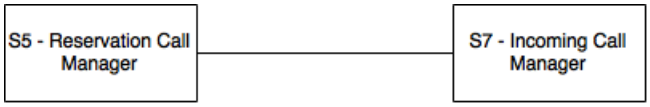


*Figure 2.*11: Integration schema for the Reservation Call management

# 3 Individual Steps and Test Description

In the tables showed in the following sub-sections, the execution precedence 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.

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

## 3.1 Guest Component subsystem (S1)

The integration steps are three: I1.1, I1.2, I1.3 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.

- **I1.1**: 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.
- **I1.2**: 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.
- **I1.3**: 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 |
|---|---|---|---|
| I1.1 | — | Guest Component, Authentication Manager | Data Manager, Notification Manager |
| I1.2 | I1.1 | Authentication Manager, Guest Component, Data Manager | Notification Manager |
| I1.3 | I1.2 | 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: I2.1, I2.2, I2.3 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).*

- **I2.1**: 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.
- **I2.2**: 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.
- **I2.3**: 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 |
|---|---|---|---|
| I2.1 | — | Customer Component, Customer Profile Manager | Data Manager, Notification Manager |
| I2.2 | I2.1 | Customer Profile Manager, Customer Component, Data Manager | Notification Manager |
| I2.3 | I2.2 | 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.

- **I3.1**: 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 |
|---|---|---|---|
| I3.1 | — | Customer Component, Authentication Manager | — |

*Table 3.*3: Integration sequence table for the Customer Profile Management testing

### 3.2.3 Customer Immediate Call Manager (S4)

The integration steps are four: I4.1, I4.2, I4.3, I4.4 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(...)*.

- **I4.1**: 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.
- **I4.2**: 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.
- **I4.3**: 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.
- **I4.4**: 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 |
|---|---|---|---|
| I4.1 | — | Customer Component, Immediate Call Manager | Data Manager, Payment Manager, Notification Manager |
| I4.2 | I4.1 | Immediate Call Manager, Customer Component, Payment Manager | Notification Manager, Data Manager |
| I4.3 | I4.2 | Customer Component, Immediate Call Manager, Payment Manager, Data Manager | Notification Manager |
| I4.4 | I4.3 | 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 Customer Reservation Call Manager (S5)

As for the Immediate Call testing case, the integrations steps are four: I5.1, I5.2, I5.3, I5.4 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(...)*.

- **I5.1**: 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.
- **I5.2**: 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 happens and the payment information for the current Call are correctly forwarded to the external banking system.

- **I5.3**: 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.
- **I5.4**: 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** |
|---|---|---|---|
| I5.1 | — | Customer Component, Reservation Call Manager | Data Manager, Payment Manager, Notification Manager |
| I5.2 | I5.1 | Reservation Call Manager, Customer Component, Payment Manager | Notification Manager, Data Manager |
| I5.3 | I5.2 | Customer Component, Reservation Call Manager, Payment Manager, Data Manager | Notification Manager |
| I5.4 | I5.3 | 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: I6.1, I6.2, I6.3 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).*

- **I6.1**: 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.
- **I6.2**: 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.
- **I6.3**: 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 |
|---|---|---|---|
| I6.1 | — | Taxi Driver Component, Taxi Driver Profile Manager | Data Manager, Notification Manager |
| I6.2 | I6.1 | Taxi Driver Profile Manager, Taxi Driver Component, Data Manager | Notification Manager |
| I6.3 | I6.2 | 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: I7.1, I7.2, I7.3 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().*

- **I7.1**: 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.
- **I7.2**: 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.
- **I7.3**: 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 |
|---|---|---|---|
| I7.1 | — | Incoming Call Manager, Notification Manager | Data Manager, Taxi Driver Manager |
| I7.2 | I7.1 | Incoming Call Manager, Notification Manager, Taxi Driver Component | Data Manager |
| I7.3 | I7.2 | 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: I8.1, I8.2, I8.3 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).*

- **I8.1**: 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.
- **I8.2**: 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.
- **I8.3**: 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 |
|---|---|---|---|
| I8.1 | — | Taxi Driver Component, Assistance Call Manager | Data Manager, Notification Manager |
| I8.2 | I8.1 | Assistance Call Manager, Taxi Driver Component, Notification Manager | Data Manager |
| I8.3 | I8.2 | 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.

- **I9.4**: 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 |
|---|---|---|---|
| I9.1 | — | Taxi Driver Component, Assistance Call Manager | Data Manager, Notification Manager |

*Table 3.*9: Integration sequence table for the Taxi Driver Logout Management testing

### 3.3.5 Immediate Call Manager (S10)

This integration test is a Second Level Subsystem test which integrates the two First Level Subsystem S4 and S7.

The preconditions, of course, includes that both the S4 and S7 integration tests are passed.

The choice to integrate this two subsystems is made because in the S4 and S7 tests some external drivers, respectively included in S7 and S4, were needed in order to accomplish the integration test; with this step, integrating the two subsystems, it's possible to reproduce and test a real situation without using any external driver.

The test is passed if all the operations related to the Immediate Call, both shared or private, involve in the correct way both the subsystems, correctly managing the interactions between the two.

| Index | Requirements | Tested Subsystems | Needed Drivers |
|---|---|---|---|
| I10.1 | S4, S7 | S4, S7 | — |

*Table 3.10*: Integration sequence table for the Immediate Call Management testing

### 3.3.6 Reservation Call Manager (S11)

This integration test is a Second Level Subsystem test which integrates the two First Level Subsystem S5 and S7.

The preconditions, as for the S10 integration test, includes that both the S5 and S7 integration tests are passed.

The choice to integrate this two subsystems is made because in the S5 and S7 tests some external drivers, respectively included in S7 and S5, were needed in order to accomplish the integration test; with this step, integrating the two subsystems, it's possible to reproduce and test a real situation without using any external driver.

The test is passed if all the operations related to the Reservation Call, both shared or private, involve in the correct way both the subsystems, correctly managing the interactions between the two.

| Index | Requirements | Tested Subsystems | Needed Drivers |
|---|---|---|---|
| I11.1 | S5, S7 | S5, S7 | — |

*Table 3.11*: Integration sequence table for the Reservation Call Management testing

## 3.4 Conclusions

The selected approach allows to test the system starting from the bottom level, in order to reach, finally, the upper level interaction testing. In this case the last tested level is represented by the subsystems S10 and S11, and no other subsystems interact each others in order to accomplish the business operations.

### 3.4.1 Performance test

For each sub-system a Performance test can be executed, in order to have an idea of what could be the supported number of users involved in the system. In particular, this tests have to be done for all the subsystems, and will allow to have an idea of the components of the real architecture to be built, taking in account the estimated number of possible customers and taxi drivers of the city which the system has to be developed for.

### 3.4.2 System test

By the way, finally a complete system test could be executed, simulating real-life situations, for example the scenarios identified in the *Section 4* of the RASD.

# 4 Tools and Test Equipments Required

- **Arquillian**: the main tool that have to be used for help the integration testing. This tool is useful in JEE because it aims to test functionalities between different components, and is not internal-component oriented. Allows the writing of integration tests for business objects that are executed inside a container or that interacts with the container as clients;[1]
- **Apache JMeter**: to be used for the performance test;

# 5 Program Stubs and Test Data Required

No Stubs are needed for this integration test plan, due to the selected approach.

[1] Reference to: `https://docs.jboss.org/arquillian/reference/1.0.0.Alpha1/en-US/html_single/`