POLITECNICO
MILANO 1863

Politecnico di Milano
A.A. 2015/2016
Software Engineering 2

Assignment 4: Integration Test Plan

Version 1.0

Alessandro Baldassari (mat. 841561)
Alberto Bendin (mat. 841734)
Francesco Giarola (mat. 840554)

January 12, 2016

Contents

1 Introduction

1.1 Revision History

This is the first version of the document. There are no previous versions.

Revision	Last Edited	Changes
1.0	21/01/2016	Document redaction

1.2 Purpose and Scope

The Test Plan Document (ITPD) describes the plan to accomplish the integration test. This document is supposed to be written before the integration test really happens and takes the architectural description of the software system as a starting point, for this reason it is often redacted in parallel with the Design Document. It explain to the development team what to test, in which sequence, which tools are needed for testing (if any), which stubs/drivers/oracles need to be developed.
The purpose of integration testing is to verify functional, performance, and reliability requirements of individual software modules of the product when they are combined and tested as a group; i.e., units (or groups of units) are exercised through their interfaces. The aim is to test the modules interactions incrementally, with success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components interact correctly, for example across procedure calls or process activations.
This is done after testing individual modules, i.e., unit testing; the overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages up to the complete final system (the testing on the complete system is not part of this integration testing phase).

1.3 List of Definitions and Abbreviations

The following acronyms are used in this document:

- RASD: Requirements Analysis and Specification Document
- DD: Design Document
- DB: DataBase

The following definitions are used in this document:

- Driver: are considered dummy modules which are always distinguished as "calling programs that are handled in bottom up integration testing; they are only used when main programs are under construction.
- Stub: in computer science, test stubs are programs that simulate the behaviors of software components (or modules) that a module undergoing tests depends on. Test stubs are mainly used in incremental testing's top-down approach.
- Oracle: in computing, software testers and software engineers can use an oracle as a mechanism for determining whether a test has passed or failed. The use of oracles involves comparing the output(s) of the system under test, for a given test-case input, to the output(s) that the oracle determines that product should have.

1.4 List of Reference Documents

- Specification document: myTaxiService project
- Requirements Analysis and Specification Document (RASD) for myTaxiService
- Design Document (DD) for myTaxiService

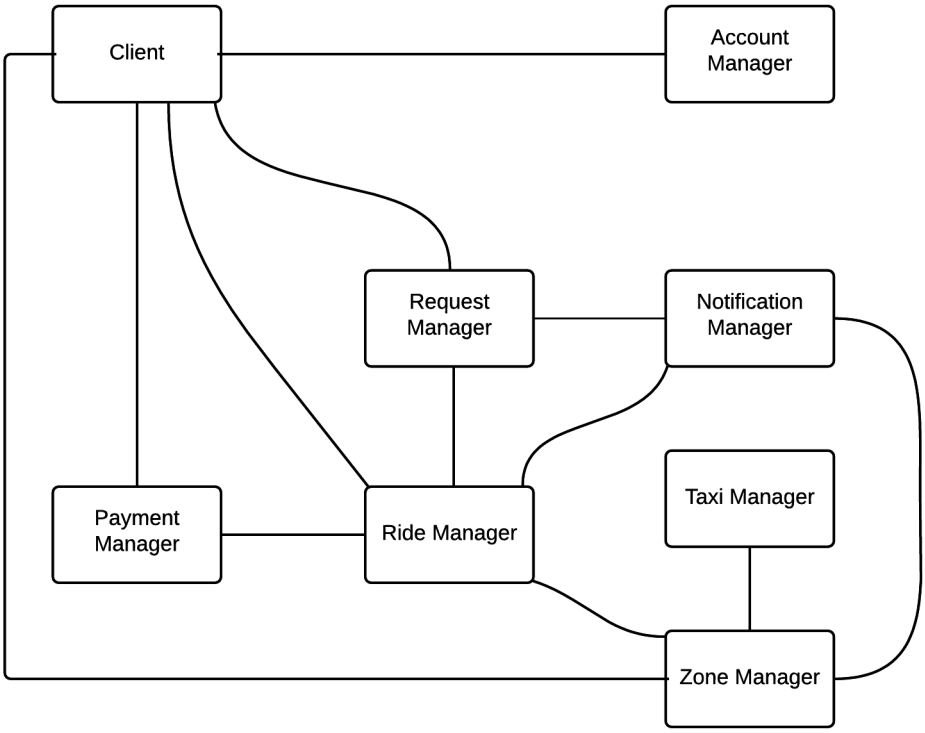
2 Integration Strategy

2.1 Entry Criteria

It is supposed that the unit testing phase has already been completed successfully.

2.2 Elements to be Integrated

The scheme below show the main components of the system.



Starting from I8 test will be focused on integration of higher level components.

ID	Integration Test	Paragraphs
I1	Login Manager →Account Factory	2.4.1 3.1
I2	Profile Manager →Login Manager	2.4.1
I3	Request Handler →Request Worker	2.4.2
I4	Request Worker →Request Engine	2.4.2
I5	Ride Handler →Ride Worker	2.4.3
I6	Ride Worker →Ride Engine	2.4.3
I7	Zone Engine →Queue Manager	2.4.4
I8	Client →Account Manager	2.4.5
I9	Client →Payment Manager	2.4.5
I10	Client →Request Manager	2.4.5
I11	Client →Ride Manager	2.4.5
I12	Client →Zone Manager	2.4.5
I13	Request Manager →Ride Manager	
I14	Request Manager →Notification Manager	
I15	Ride Manager →Zone Manager	
I16	Ride Manager →Payment Manager	
I17	Ride Manager →Notification Manager	
I18	Zone Manager →Taxi Manager	
I19	Zone Manager →Notification Manager	

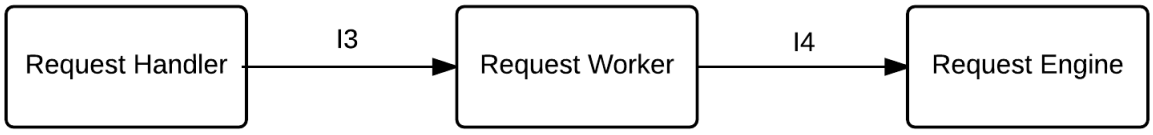
2.3 Integration Testing Strategy

The bottom-up approach will be used to accomplish the analysis. This means that first the interactions inside the sub-systems of the components will be considered and consequently the integrations between the components themselves.

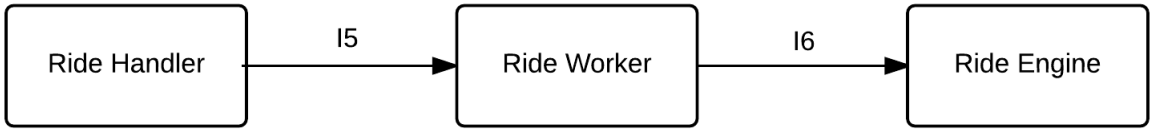
2.4 Sequence of Component/Function Integration

2.4.1 Sequence of integration for Account Manager

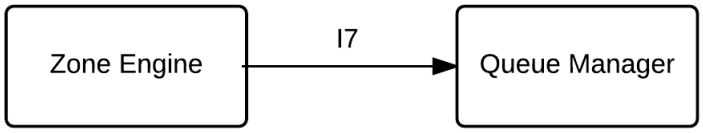
2.4.2 Sequence of integration for Request Manager



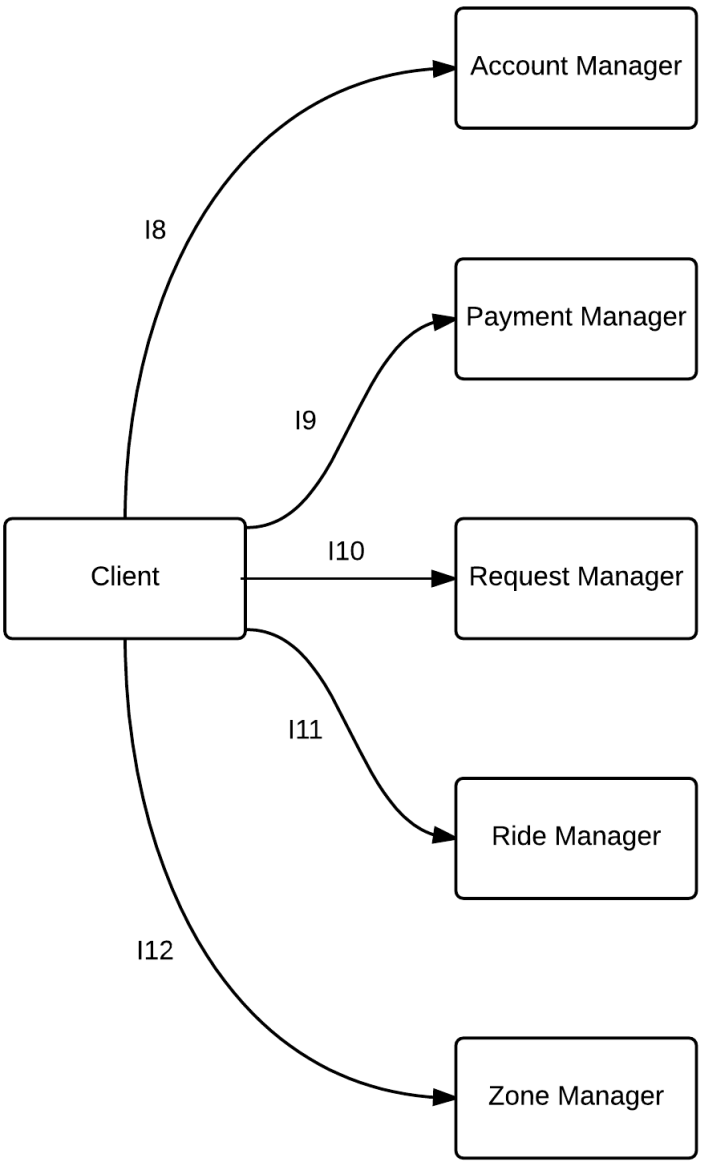
2.4.3 Sequence of integration for Ride Manager



2.4.4 Sequence of integration for Zone Manager



2.4.5 Sequence of integration for Client



3 Individual Steps and Test Description

3.1 Integration tests for Account Manager

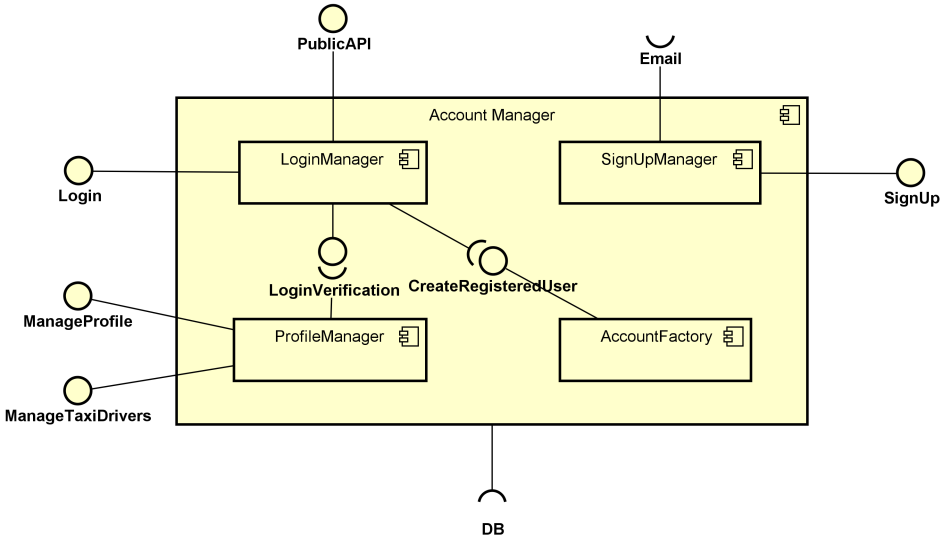


Figure taken from paragraph 2.3.1 of the DD

Test case identifier	I1
Test item(s)	Login Manager →Account Factory
Input specification	Create typical Login Manager input
Output specification	Check if the correct methods are called in the Account Factory
Environmental needs	Client driver

Test case identifier	I2
Test item(s)	Profile Manager →Login Manager
Input specification	Create typical Profile Manager input
Output specification	Check if the correct functions are called in the Login Manager
Environmental needs	Client driver

3.2 Integration tests for Request Manager

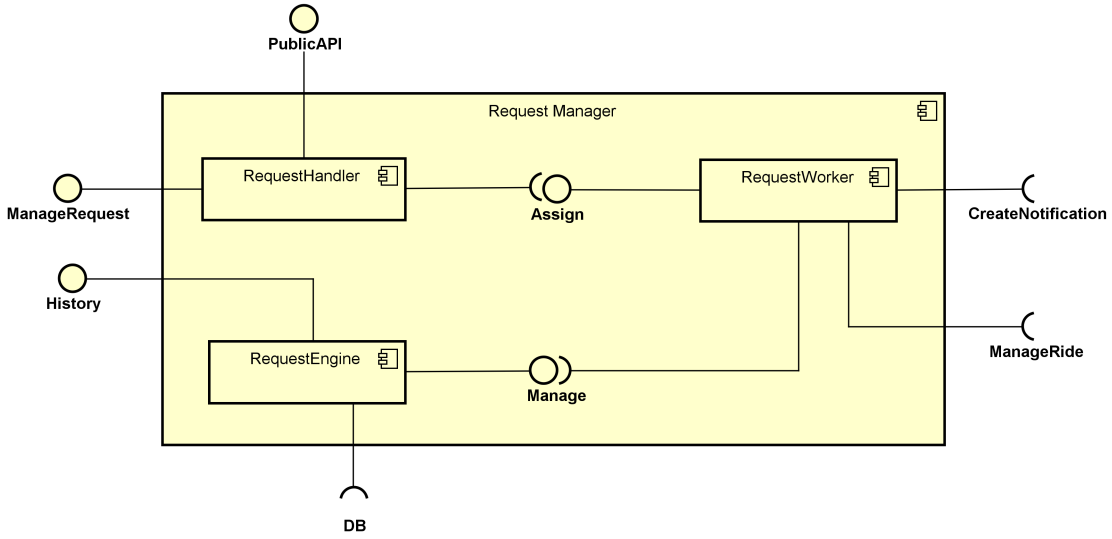


Figure taken from paragraph 2.3.2 of the DD

Test case identifier	I3
Test item(s)	Request Handler →Request Worker
Input specification	Create typical Request Handler input
Output specification	Check if the correct functions are called in the Request Worker
Environmental needs	Client (Passenger) driver to simulate a typical communication input (creation or modification of request/reservation), the stubs for Ride Manager and Notification Manager to satisfy the components inter-dependencies

Test case identifier	I4
Test item(s)	Request Worker →Request Engine
Input specification	Create typical Request Worker input
Output specification	Check if the correct functions are called in the Request Engine
Environmental needs	I3 succeeded, the stub for the DB to satisfy the component inter-dependencies

3.3 Integration tests for Ride Manager

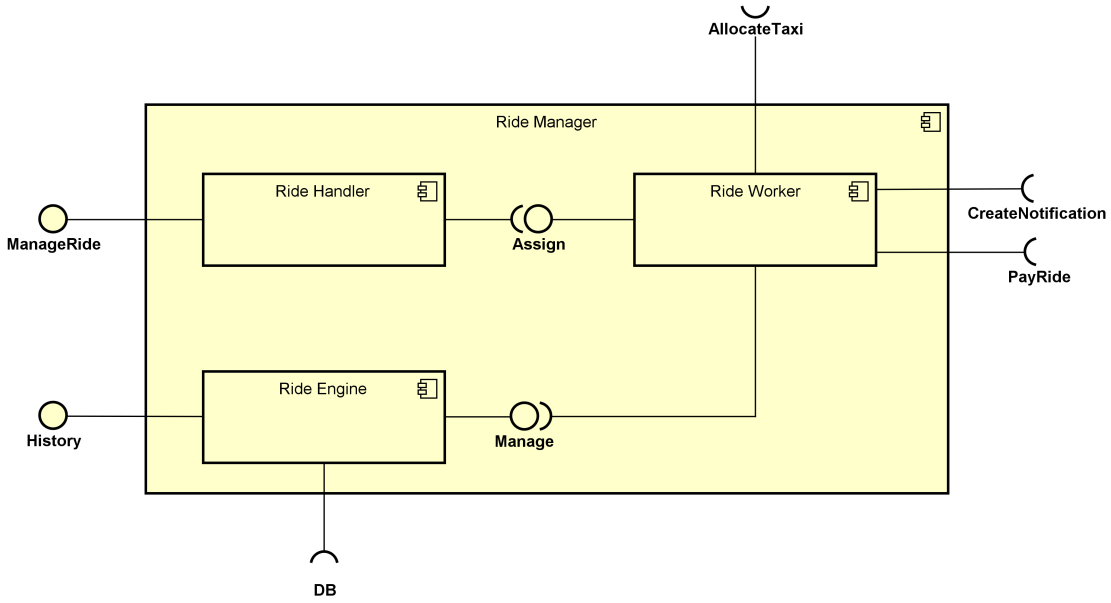


Figure taken from paragraph 2.3.3 of the DD

Test case identifier	I5
Test item(s)	Ride Handler →Ride Worker
Input specification	Create typical Ride Handler input
Output specification	Check if the correct functions are called in the Ride Worker
Environmental needs	Request Manager driver to simulate a typical communication input (creation or modification of ride), the stubs for Payment Manager, Zone Manager and Notification Manager to satisfy the components inter-dependencies

Test case identifier	I6
Test item(s)	Ride Worker →Ride Engine
Input specification	Create typical Ride Worker input
Output specification	Check if the correct functions are called in the Ride Engine
Environmental needs	I5 succeeded, the stub for the DB to satisfy the component inter-dependencies

3.4 Integration tests for Zone Manager

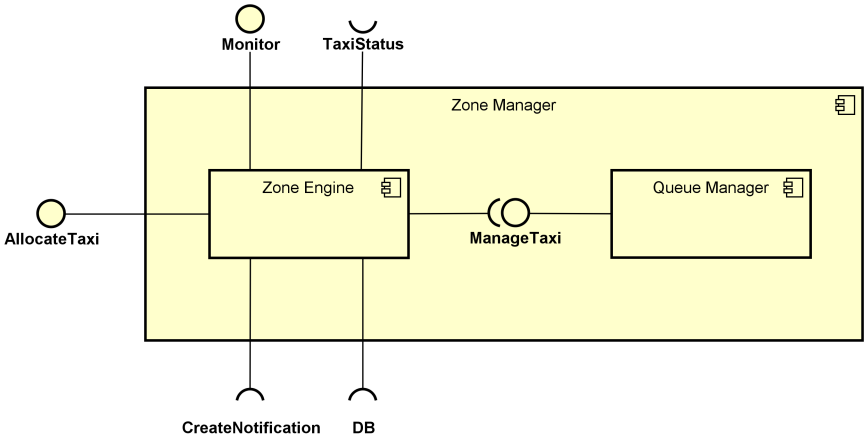


Figure taken from paragraph 2.3.6 of the DD

Test case identifier	I7
Test item(s)	Zone Engine →Queue Manager
Input specification	Create typical Zone Engine input
Output specification	Check if the correct functions are called in the Queue Manager
Environmental needs	Ride Manager driver to simulate a typical communication input (allocation of taxis), the stubs for Taxi Manager, DB and Notification Manager to satisfy the components inter-dependencies

3.5 Integration tests for Client and Account Manager

Test case identifier	I8
Test item(s)	Client →Account Manager
Input specification	Create typical Client input
Output specification	Check if the correct functions are called in the Account Manager
Environmental needs	I1, I2 succeeded, the stubs for DB and Notification Manager to satisfy the components inter-dependencies

3.6 Integration tests for Client and Payment Manager

Test case identifier	I9
Test item(s)	Client →Payment Manager
Input specification	Create typical Client input
Output specification	Check if the correct functions are called in the Payment Manager
Environmental needs	The stub for Public Payment Service to satisfy the components inter-dependencies

tp sequence come ale per i 4 macro blocchi account, request,ride, zone.
magari incollare i 4 sequence diagrams e per ogniuno fare la tp

4 Tools and Test Equipment Required

Mockito, Arquillian, JMeter

5 Program Stubs and Test Data Required

6 References

Material from Wikipedia

- Integration testing: https://en.wikipedia.org/wiki/Integration_testing
- Oracles: https://en.wikipedia.org/wiki/Oracle_(software_testing)

7 Appendix

7.1 Software and tools used

- TeXstudio 2.10.6 (http://www.texstudio.org/) to redact and format this document.
- Astah Professional 7.0 (http://astah.net/editions/professional)

7.2 Hours of work

The time spent to redact this document:

- Baldassari Alessandro: 20 hours.
- Bendin Alberto: 20 hours.
- Giarola Francesco: 20 hours.