Integration Test Plan Document

Politecnico di Milano **A.A. 2015-2016**

Software Engineering 2 Raffaela Mirandola

Salvatore Perillo 853349 Claudio Sesto 841623

Summary

1. Introduction

1.1 Revision History

21/01/2016 Version 1.0

1.2 Purpose and Scope

The purpose of this Integration Test Plan Document is to provide a general guideline for the integration test of the different components.
The software implemented is the one described in the RASD and DD document, the integration test will ensure that all the components of this system will work and cooperate correctly.

1.3 Definitions and Abbreviations

- Central System -- The entire sofware system with all component linked
- Q/R Manager -- The manager responsible of all actions concerning the queue of one zone and the management of the all incoming requests.
- Taxi Mobile App -- The application used by the taxi driver uses to respond to the incoming requests and sets its own state.
- Techinician Station -- the personal computer dedicated to implement new features to add to the main system.

1.4 Reference Documents

- Assignement 4 - Integration Test Plan
- R.A.S.D My Taxi Service
- D.D. My Taxi Service
- Slide about Verification Tools

2. Integration Strategy

2.1 Entry Criteria

Refering to the Design Document, we can divide our system in 7 subsystems:

- Central system
- Q/R Manager
- Taxi mobile App
- GPS
- Databases (static/dynamic)
- Web/Mobile App
- Technician station

Each of this subsystem must work before performing the integration test.
We assume that:

- ➢ The software used in the technician station have been bought and work properly
- ➢ The databases are implemented with an existing technology (MySQL or similar) so work correctly
- ➢ The GPS is provided by a company, we must just test the communication with the taxi driver's smartphone

Instead, for what concerning the entry criteria about the other component, and in particular the web/mobile app, before integration test they must be installed on the Central System and must be online/available for download; the same is for the taxi application.
About the Q/R Manager we also already have implemented the division of the city zones

2.2 Elements to be Integrated

Depending on the section above, the elements that need to be integrated for their correctness are:

- ❑ The GPS (is a external GPS placed on the taxicar) that must be integrated with the application installed on the taxi driver's smartphone.
- ❑ All the Technician Workstations that must be integrated with the central system.
- ❑ The databases (we have two databases, Static and Dynamic as described in the D.D.) that must be integrated with the "Taxi Software" and the Web/Mobile Application to guarantee a correct flow of data between the components and a safe saving.

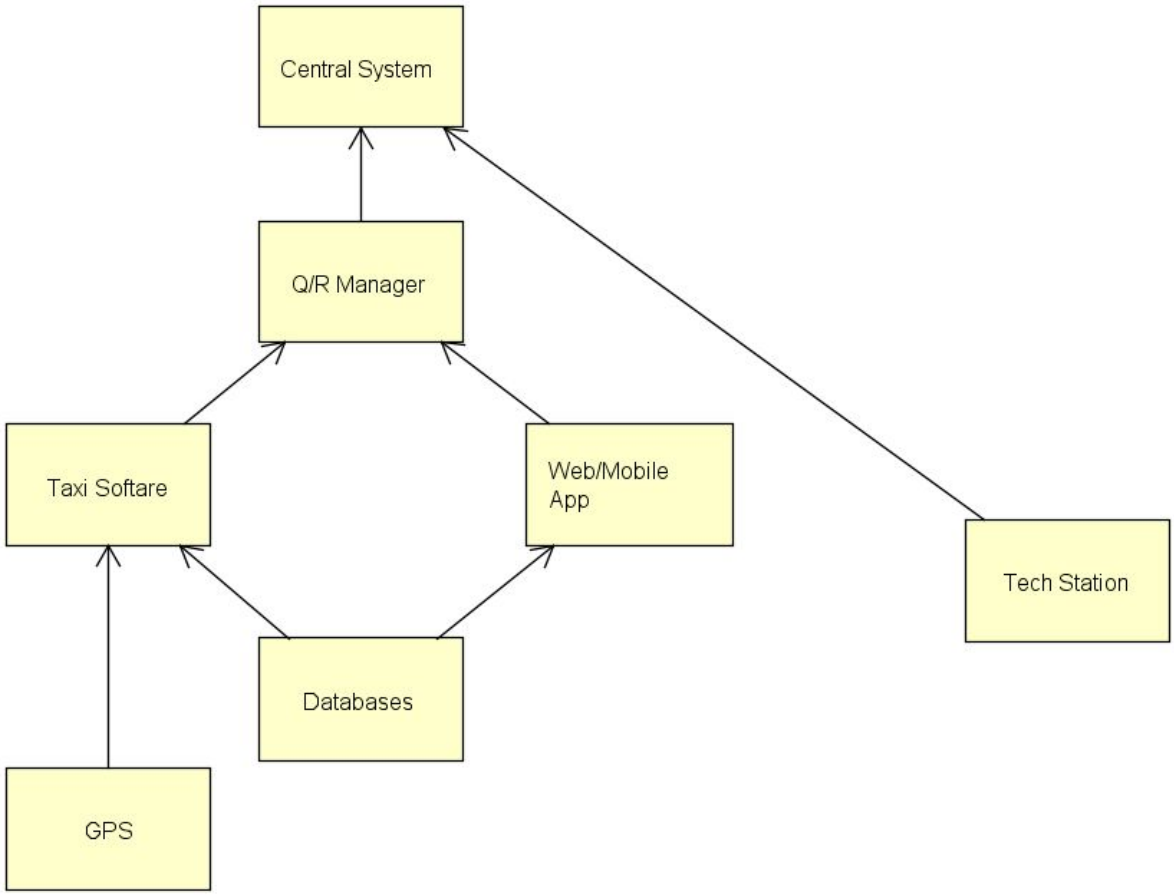
2.3 Integration Testing Strategy

We 've decided to use the bottom up testing strategy: in this way the integration of the top component will be simpler once the lower ones are integrate.Our strategy is not a standard bottom up ,because in some point we need to create a stub of a "higher" component to understand if the "lower" one works correctly.

2.4 Sequence of Component/Function Integration

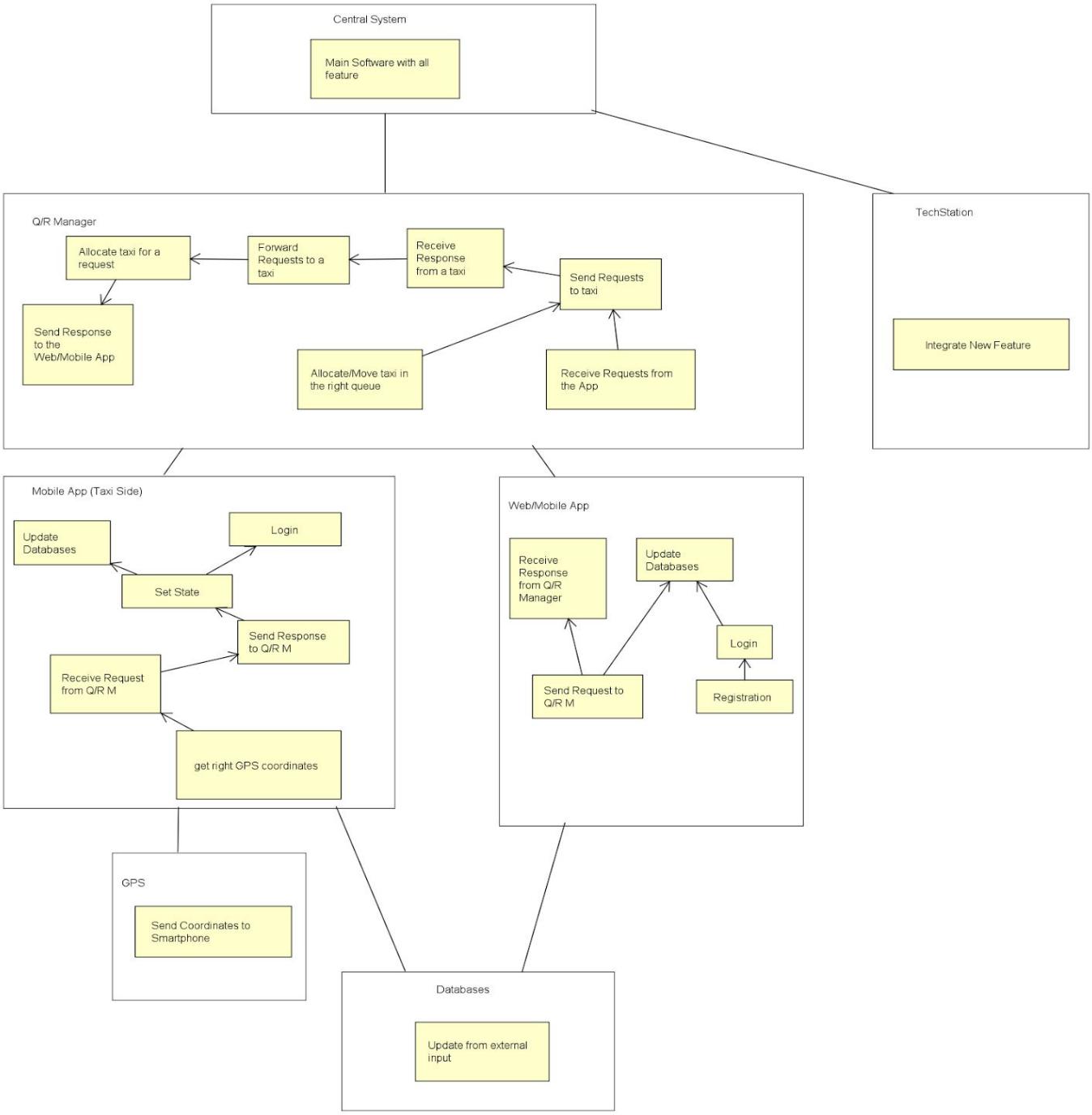
2.4.1 Sofware Integration Sequence

Due the choice we made in the previous paragraph, the testing part will start from the lower levels, that are also the smaller components: each component has also some internal functions that must be tested in order to guarantee the correctness of the component.
This detail is shown in the next subparagraph.



2.4.2 Subsystem Integration Sequence

In the following graph you can see the components with their main functions that has to be tested: the arrows shows the order in which the test has to be performed. Obviously, a function of a component cannot be tested before all the previous functions of that component and all the functions and components of lower levels have been tested are integrated.



3. Individual Steps And Test Description

Component	*Function*	*Input condition*	*Output condition*
Database	Update from external input	Create typical input for a database.	Any tuple insertion/update must be saved.
Web/Mobile App	Registration	Create typical input for a registration.	Check that correct methods are called in the Database
	Login	Create typical input for a login.	Check that correct methods are called in the Database
	Send request to Q/R manager	Create input of a simple request.	Check that correct methods are called in the Q/R Manager
	Update databases	Create registration input	Check that correct methods are called in the Database
	Receive response from Q/R Manager	Create Q/R manager response.	Check that correct methods are called in the Q/R Manager
Mobile App (taxi side)	Get right GPS Coordinates	Create Typical input of a GPS.	Check that internal methods and database methods are correctlycalled
	Receive request from Q/R Manager	Create input of a simple request	Check that correct methods are called in the Q/R Manager
	Send response to Q/R Manager	Create Q/R manager response	Check that correct methods are called in the Q/R Manager
	Set state	Create Boolean input	Check that internal correct methods are called
	Login	Create typical login input	Check that correct methods are called in the Database
	Update databases	Create typical data that can be sent by the Mobile App (Taxi side)	Check that correct methods are called in the Database
Technician Station.	Integrate new features	Create a trivial simple feature	Check that the central system uses the new features

Q/R Manager	Allocate/Move right taxi in the queue	Create typical coordinates intput	Check that correct methods are called in the Q/R Manager
	Receive request from web/mobile app	Create a example of a request	Check that correct methods are called in the Q/R Manager
	Send request to a taxi	Create typical request input	Check that correct methods are called in the Q/R Manager
	Receive response from mobile app (taxi side)	Create typical response input	Check that correct methods are called in the Q/R Manager
	Forward request to a taxi	Create typical request input	Check that correct methods are called in the mobile app (taxi side)
	Allocate taxi for a request	Create positive input request	Check that correct methods are called in the Q/R Manager
	Send response to web/mobile app	Create typical response input	Check that correct methods are called in the mobile app
Central system	Main software with all features	Create Q/R manager input	Check that correct methods are called

Purposes of the tests

- Databases
  - Update from external input
    - Verify that the databases are updated whenever there is an input or a modification of a tuple.
- Web/Mobile App
  - Registration
    - Check that data must be stored in the static database.
  - Login
    - After a user have been registered, check if is possible to enter in the personal page with its data.
  - Send Request to Q/R Manager
    - After the insertion of a data for one request, check if this data are send correctly to the Q/R Manager.
    - Verify that the communication from Web/Mobile App to the Q/R Manager works correctly.
  - Update Databases
    - Check if the data of a request are correctly sent to the databases.
  - Receive Response from Q/R Manager
    - Verify that the communication from Q/R Manager to the Web/Mobile App works correctly.
- Mobile App (Taxi Side)
  - get right GPS coordinates
    - Check that the coordinates are consistent with the place in which you are.
    - Verify that the communication from GPS to the Mobile App (Taxi Side) works correctly.
  - Receive Request from Q/R Manager
    - Check if a “Request“ type input can be received and handled correctly.
    - Verify that the communication from Q/R manager to the Mobile App (Taxi Side) works correctly.
  - Send Response to Q/R Manager
    - Check that a “Response” type data can be sent to the Q/R Manager.
    - Verify that the communication from Mobile App (Taxi Side) to the Q/R Manager works correctly.
  - Set State
    - Check that the status of the taxi is consistent with the choice.
  - Login
    - Same as “Login” in the mobile app section.
  - Update Databases
    - Same as “Update Databases of mobile app section.

- Tech Station
  - Integrate new feature
    - The new feature must be visible by the system and can be used by users.
- Q/R Manager
  - Allocate/Move taxi in the right queue
    - Check if a istance of a taxi can be moved correctly from one queue to another following the current value of the coordinate.
  - Receive Request from the Web/Mobile App
    - Check that a “Request” data type can be received and handled correclty.
    - Verify that the communication from Web/Mobile App to the Q/R Manager works correctly.
  - Send Requests to a taxi
    - After a request is received by the Q/R Manager, this one is sent to the correct taxi.
    - Verify that the communication from Q/R Manager to the Mobile App (taxi side) works correctly.
  - Receive Response from a taxi
    - Verify that the communication from Mobile App (taxi side) to the Q/R Manager works correctly.
  - Forward Requests to a taxi
    - After a request is declined from a taxi driver, the manager must be able to forward the request to the others following the right algorithm.
  - Allocate taxi for a request
    - If the request is accepted, this taxi will not receive any request until the state is set again to “Free”.
  - Send Response to the Web/ Mobile App
    - Verify that the communication from Q/R Manager to the Web/ Mobile App works correctly.
- Central System
  - Main Software with all feature
    - After all the other steps are checked , there is the need to check that all software works.

4. Tools and Test Equipment Required

4.1 Manual Testing

For most of the tests we have to perform we can use manual testing by inserting, for example, data pretending to be the real one: it's simple to do that because those are just strings, apart from the GPS coordinates.

4.2 Mockito

Mockito is used in those tests where we need to create a stub that send/receive automatically the right type of data that the situation requires.
One example of this situation is:
Test the point "Receive request from Q/R Manager " in the Taxi Software section.A stub that send a "request" data is needed.

4.3 JMeter

JMeter must be used to check if the Q/R Manager can handle the high load of traffic due to the dynamics of the system ,like taxi status setting and all request coming from the web/Mobile App.

5. Program Stubs and Test Data Required

There is a need of stub in this cases:

1. Mobile App (Taxi Side) --> Receive Request from Q/R Manager
   A stub of Q/R Manager that only send the correct type of request is created.
2. Mobile App (Taxi Side) --> Send Response from Q/R Manager
   A stub of Q/R Manager that only receive the response is created.
3. Web/Mobile App → Send Request to Q/R Manager
   A stub of Q/R Manager that only receive the correct type of request is created.
4. Web/Mobile App →Receive Response from Q/R Manager
   A stub of Q/R Manager that only send the correct type of response is created.

As mentioned before, in order to test any component, all the components at the "lower" levels must be already tested and integrated.
Some example:

- to test some function of the Q/R Manager we need to have the correct data of the tests of the two App(Mobile App (Taxi side) , Web/Mobile App)
- to test some function of the two App we need to test the correct use of the databases.

6. Hours of Work

Salvatore Perillo 15h

Claudio Sesto 15h