Chapter 1

Integration Strategy

In this chapter the integration strategy will be described. In the section 1.1 the prerequisites for the tests will be presented. The section 1.2 is dedicated to the required components to be integrated in the system to execute some tests. Finally in the sections 1.3 and 1.4 the strategy used to test the integrations will be discussed, paying attention to the order.

1.1 Entry Criteria

The criteria which are required to perform each test are easy. The involved components (and eventually needed program stubs) must be developed and fully tested before executing the integration test.
In addition, for some test, further preconditions may be required. These prerequisites will not be described here, but in the **??**, dedicated to test's description.

1.2 Elements to be Integrated

First of all, no further components need to be integrated to perform integration tests.
Then, to perform the tests described in the following pages, all the components of the system need to be implemented. Furthermore, it is not required that all the components must be implemented before executing the first integration test:

in the subsection 1.4.1 the order of implementation is well-presented focusing both on the subsystems and on the reasons for the required order.

1.3 Integration Testing Strategy

We have decided to test our system's interactions with a bottom-up approach. In the section 1.4 it is possible to see the components' order of implementation defined by us with specific criteria. As soon as two components are available, the tests associated to their interactions have to be applied. Before starting the development of the following part of the system, all the tests should be successful.
Obviously, to perform some tests, we need to use a *non-available* component (it has not been implemented yet): to simulate this components we will use some particular program stubs, if any, and they will be described in detail in the **??**.

1.4 Sequence of Component/Function Integration

In this section we point out the features related to the order of the components' development. Which is our order of implementation? Why do we need to develop the component *A* before *B*? These two questions are fully discussed from now.

1.4.1 Software Integration Sequence

The "subsystems" of *myTaxiService* are two, both important.
The *Client and User Handler* has the specific role to bind the users' requests to the corresponding services on the system by enqueueing the request on the dispatcher. Then it is able to show the results or to perform the following interactions of the involved service.
To handle these operations it has three components, with the names *Authentication*, *ServiceShow* and *ClientInterface*[1]. In the Figure 1.1 the order of implementa-

[1]To have a brief description of each of them read the Design Document.

tion is shown[2].

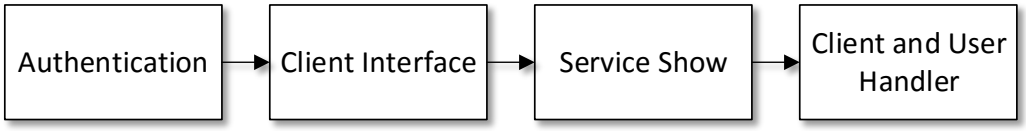

Figure 1.1: Integration Sequence of Client and User Handler

The *Ride Allocator* handles everything that is related to a ride, so the taxy drivers' queues, their assignments and the identification of their positions (by GPS coordinates or an address).
A more precise description is available in the Design Document, as usual, whereas here we are interested only in the implementation order shown in Figure 1.2

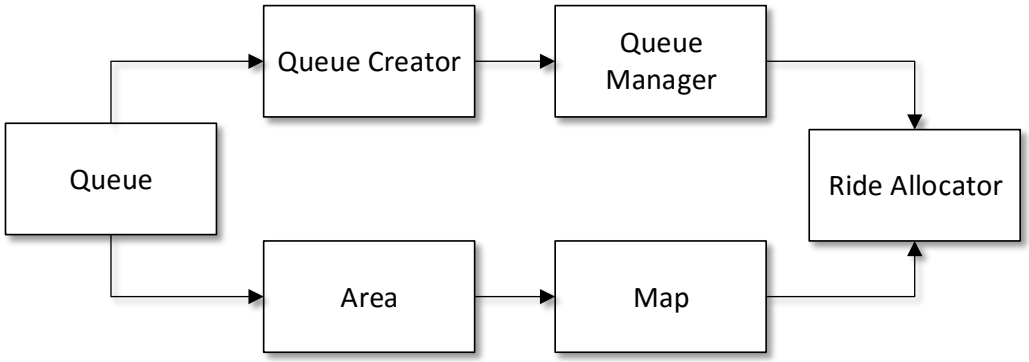

Figure 1.2: Integration Sequence of Ride Allocator

The interactions and the integration inside these two subsystems are not tested in this project phase, but when the single components are implemented. In fact, each component is tested as soon as its first implementation has been completed and until all the tests are successful the development can not be considered done.

[2]According to the Design Document the reader may expect to see all the interfaces of the Client and User Handler. To simplify the representation all the interfaces and the classes have been grouped and inserted into the related component.

This kind of tests are defined in a dedicated document, called *Unit Test Plan Document*, not available now for *myTaxiService*.

1.4.2 Subsystem Integration Sequence

In this section we will present the global interactions between the *myTaxiService*'s core subsystems and we will define all the integration tests and their order.
In the Figure 1.3 we can see the components' implementation order, as we suppose it is better: first of all the DBMS, then, in this order, all related to data checking or handling, to the users, to the security, to the rides and, finally, the dispatcher (the most important component in our system).

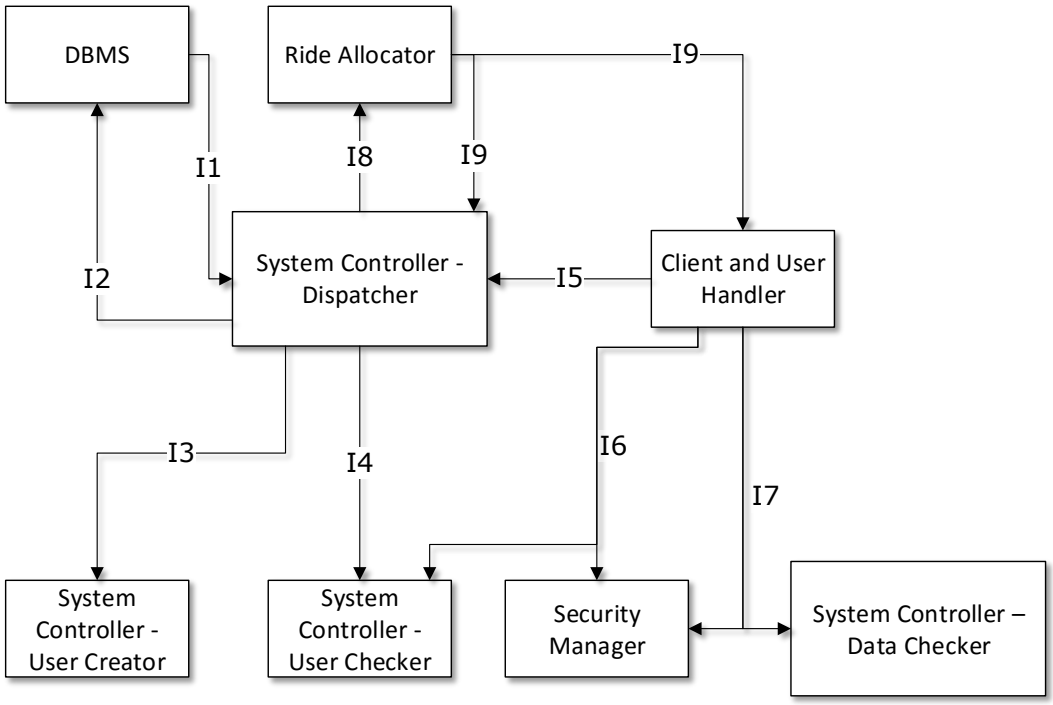

Figure 1.3: Integration Test Order

Integration tests.

ID	Integration test
I1	DBMS → System Controller - Dispatcher
I2	System Controller - Dispatcher → DBMS
I3	System Controller - Dispatcher → User Creator
I4	System Controller - Dispatcher → User Checker
I5	Client and User Handler → System Controller - Dispatcher
I6	Client and User Handler → User Checker , Security Manager
I7	Client and User Handler → System Controller - Data Checker , Security Manager
I8	System Controller - Dispatcher → Ride Allocator
I9	Ride Allocator → System Controller - Dispatcher , Client and User Handler