POLITECNICO
MILANO 1863

Software Engineering 2: "PowerEnJoy"

Integration Test Plan Document

Version 1.0

Piccirillo Luca - 790380
Zampogna Gian Luca - 863097
Zini Edoardo - 875275

January 15th, 2017

Contents

1 Introduction

1.1 Revision History

Version	Date	Changes
1.0-RC1	15/01/2017	First deadline release.

1.2 Purpose and Scope

This document contains indications about how to proceed during the testing phase and the step by step integration of the PowerEnJoy system components. The expected audience of this document are test developers as well as project managers, who need to coordinate testing with developing activities.

1.3 List of Definitions and Abbreviations

1.3.1 Acronyms, Abbreviations

- GPS = Global Positioning System
- ORAA = Onboard Ride Assistant App
- PEJ = PowerEnJoy
- SBL = Service Business Logic
- SPA = Special Parking Area
- UDMA = User Device Mobile App

1.3.2 Definitions

- **Car:** every vehicle, which respects the requirements, that the system allows the users to use.
- **Onboard Ride Assistant App:** software logic that manage what is shown on the Car screen.
- **Safe Parking Area:** pre-defined areas (i.e. streets) where a user is allowed to park.
- **Service Business Logic:** software logic of the service the user do not directly interact with.
- **Special Parking Area:** pre-defined areas (i.e. streets) where a user is allowed to park and where the batteries of the Cars can be plugged into the power grid.
- **User Device Mobile App:** mobile application the user will interact with on his mobile phone.

1.4 List of Reference Documents

- Project's Assignment document: AA 2016-2017 Software Engineering 2 - Project goal, schedule, and rules.
- Software Engineering 2: "PowerEnJoy" Requirements Analysis and Specification Document.
- Software Engineering 2: "PowerEnJoy" Design Document.
- Xamarin test cloud: `https://www.xamarin.com/test-cloud`
- xUnit: `https://en.wikipedia.org/wiki/XUnit`
- HockeyApp: `https://hockeyapp.net/#s`

2 Integration Strategy

2.1 Entry Criteria

For each component we state which criteria must be satisfied before starting testing. Please note that the criteria do not cover the requirements to successfully complete a test, but only for starting related tests developing stage.
SBL:

- **Booking Handler:** all other SBL components test passed.
- **Zone Manager:** at least one between Vehicle Tracker and Special Areas Assistant is tested.
- **Account Manager:** none.
- **Payment Processor:** none.
- **Vehicle Tracker:** none.
- **Special Areas Assistant:** none.
- **PEJ Context:** none.

User Device Mobile App:

- **Account:** none.
- **View:** all other UDMA components, except View Router, are tested.
- **View Router:** all other UDMA components, except View, are tested.
- **Map Wrapper:** none.
- **Map Explorer:** Map Wrapper is tested.
- **Reserved Car Viewer:** Map Wrapper is tested.

Onboard Ride Assistant App:

- **Status Monitor:** none.
- **Map Wrapper:** none.
- **View:** all other ORAA components, except View Router, are tested.
- **View Router:** all other ORAA components, except View, are tested.

Other components:

- **Client API/Accounting:** none.

- **Client API/Booking:** none.
- **Live Vehicle API:** none.

The remaining components do not need to be tested as they are provided by third parties. They will be used in testing the previously listed components that rely on them:

- **Entity Framework**
- **Payment Processor Provider**
- **Mail Server**
- **Google Maps**
- **Navigator**

2.2 Elements to be Integrated

PEJ System is composed of three major subsystems: Service Business Logic, User Device Mobile App and Onboard Ride Assistant App.
The SBL provides the core functions of the service, and therefore is composed by: Booking Handler, Zone Manager, Account Manager, Payment Processor, Vehicle Tracker, Special Areas Assistant, PEJ Context.
The User Device Mobile App connects to SBL via APIs, Client API/Accounting and Client API/Booking. Its components are: Account, View, View Router, Map Wrapper, Map Explorer, Reserved Car Viewer.
The Onboard Ride Assistant App connects to the SBL via Client API/Booking. Its components are: Status Monitor, Map Wrapper, View, View Router.
In order to fulfil their function the subsystems relies on the following components: Entity Framework, Live Vehicle API, Payment Processor Provider, Mail Server, Google Maps and Navigator.

2.3 Integration Testing Strategy

We planned a bottom-up testing strategy on each subsystem, in order to minimize the need for stubs and to ensure the testing phase can proceed in parallel for all three subsystems, furthermore components become ready to be tested as soon as they are developed without needing for the whole subsystem to be completed. However some stubs are still required if the project manager decides to start testing an higher level component upon only a single lower level component as soon as its development completes, to proceed in a pipelined fashion.
Both Client APIs and Live Vehicle API can be tested immediately using test drivers and stubs, these components are tested alone as first step in order to ensure no communication problem caused by API components will arise while integrating with other subsystems. At the same time the testing on subsystems can starts. The PEJ Context component in SBL is the one all other components of SBL rely on to get the data they need, so its test

will be carried out through testing of other components. Further components that can be initially tested are Map Wrapper in both User Device Mobile App and Onboard Ride Assistant App, since both Navigator and Google Maps are provided by third parties and are already available at the beginning of the test phase. Other components that can be tested immediately are Status Monitor in ORAA and Account in UDMA.
Further details about the rationale of this strategy can be found in the next section of the document, along with all the explanation about how to proceed for the integration testing.

2.4 Sequence of Component Integration

Components outside the subsystems that must be tested are Client API/Accounting, ClientAPI/Booking, Live Vehicle API. These three are tested using test drivers and stubs.

2.4.1 Software Integration Sequence

2.4.1.1 Onboard Ride Assistant App subsystem

The first components to be tested are Map Wrapper and Status Monitor, since the first requires only the IMapProvider interface provided by Navigator, while the second requires a stub for the ClientAPI/Booking too. See *Figure 1*.

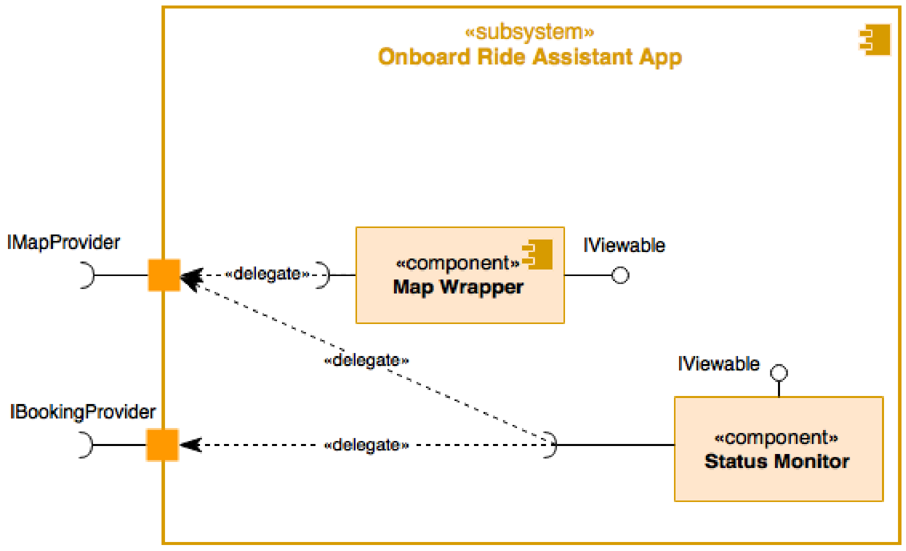

Figure 1: ORAA Phase One

Last components of the Onboard Ride Assistant App to be tested are View and View Router, whose testing will be carried out ensuring that all possible navigation paths established in the UX Diagram contained in the Design Document are possible, and the View always provide the right information with respect to UX Diagram and the mockups in RASD. See *Figure 2*.

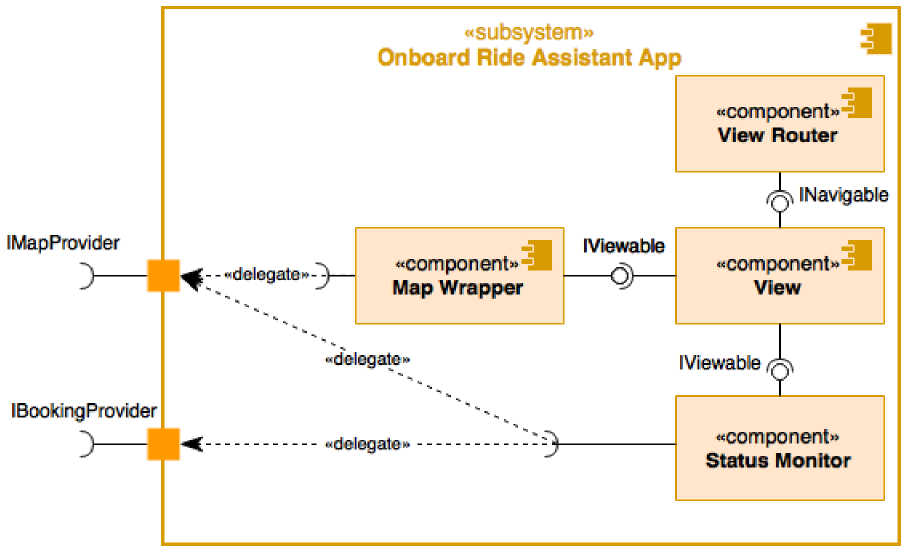

Figure 2: ORAA Phase Two

2.4.1.2 Service Business Logic subsystem

Four components can be tested at the same time: Special Areas Assistant and Vehicle Tracker, which require a test driver since Zone Manager is not tested, and also a stub of Live Vehicle API for Vehicle Tracker; Payment Processor and Account Manager, which require a driver since Booking Handler is not tested. Furthermore Account Manager will use a test driver of ClientAPI/Accounting, while the IMAiler interface can be used for mail related functions. See *Figure 3*.

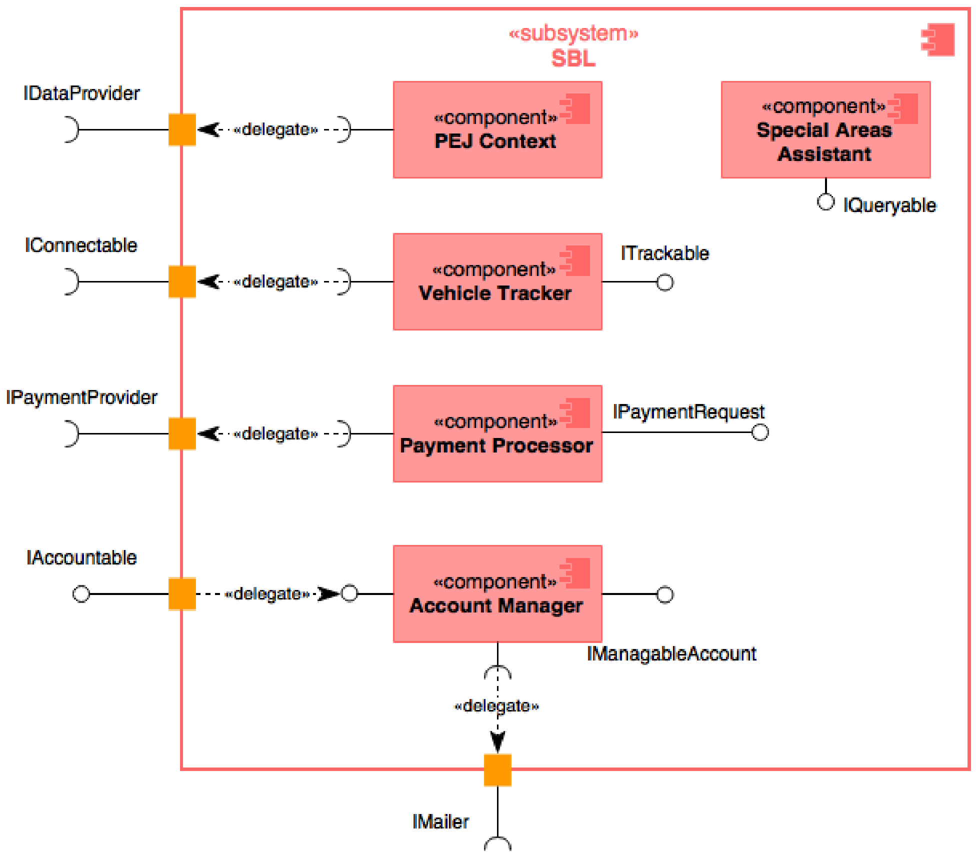

Figure 3: SBL Phase One

Zone Manager is being tested; except for Booking Handler, all components are now tested; this way Zone Manager doesn't need any stub and can be tested using just one test driver instead of the actual Booking Handler. See *Figure 4*.

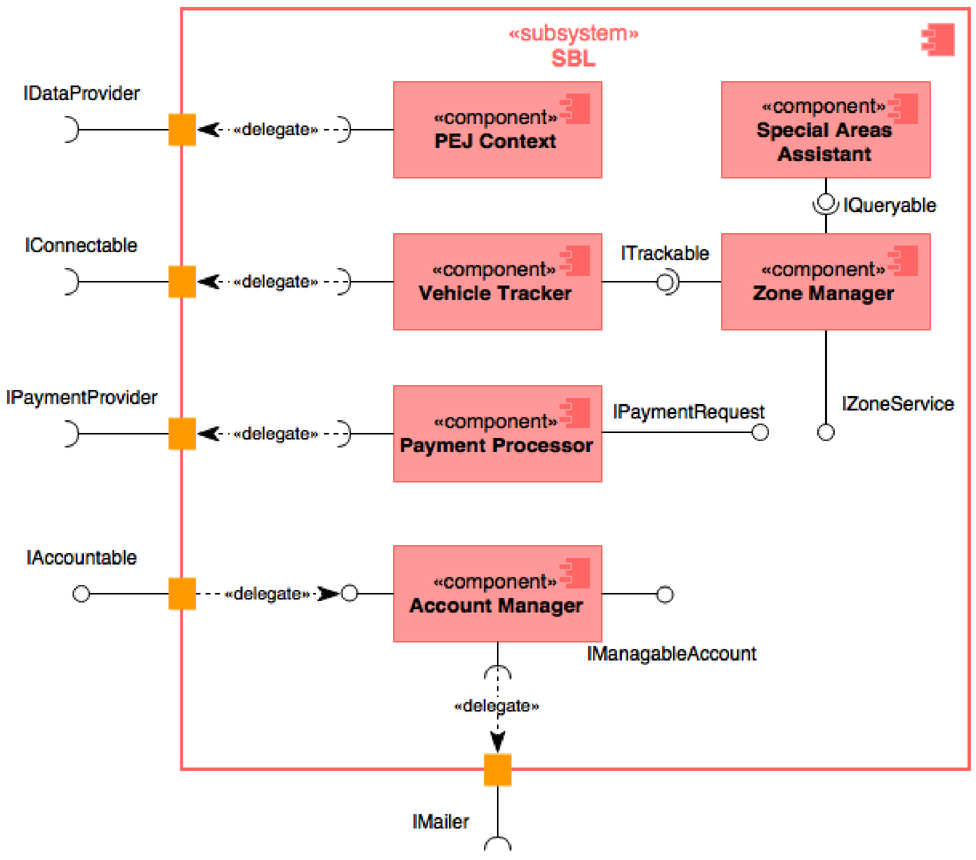

Figure 4: SBL Phase Two

All other components are tested, so now Booking Manager can be tested without the need for any stub or driver. See *Figure 5*.

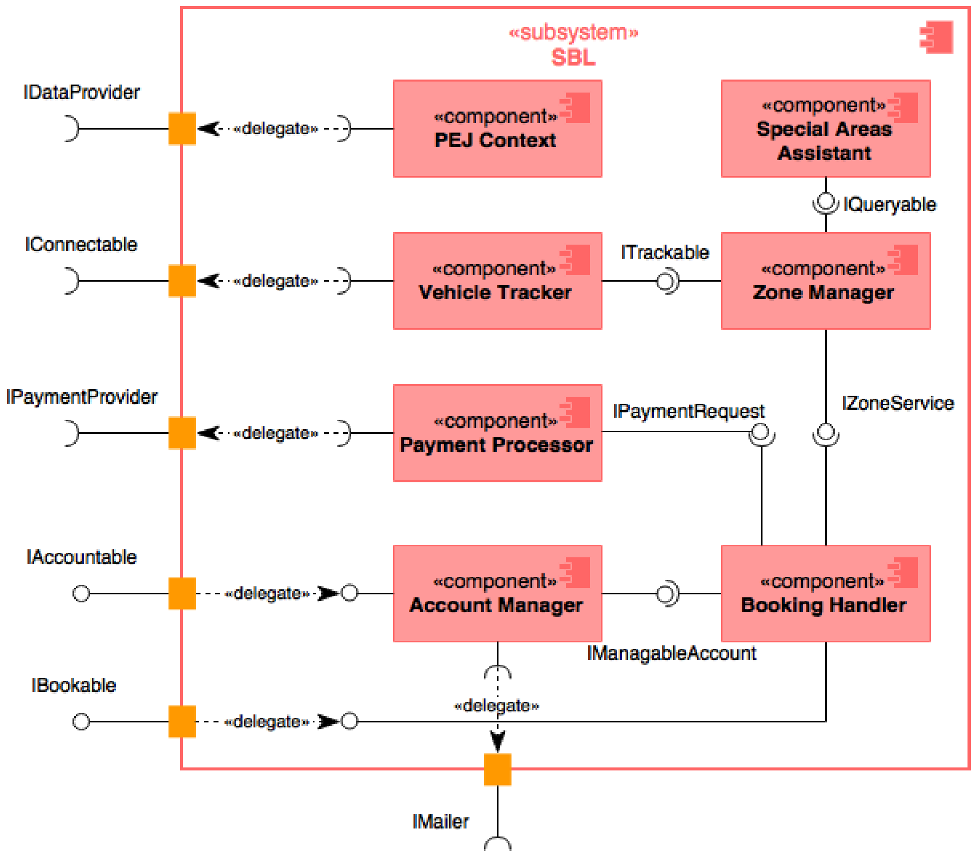

Figure 5: SBL Phase Three

2.4.1.3 User Device Mobile App subsystem

First components to the tested are Map Wrapper and Account. The first relies on Google Maps to get the data it needs, a test driver is used for testing; while Account requires also a stub in addition to the test drive. See *Figure 6*.

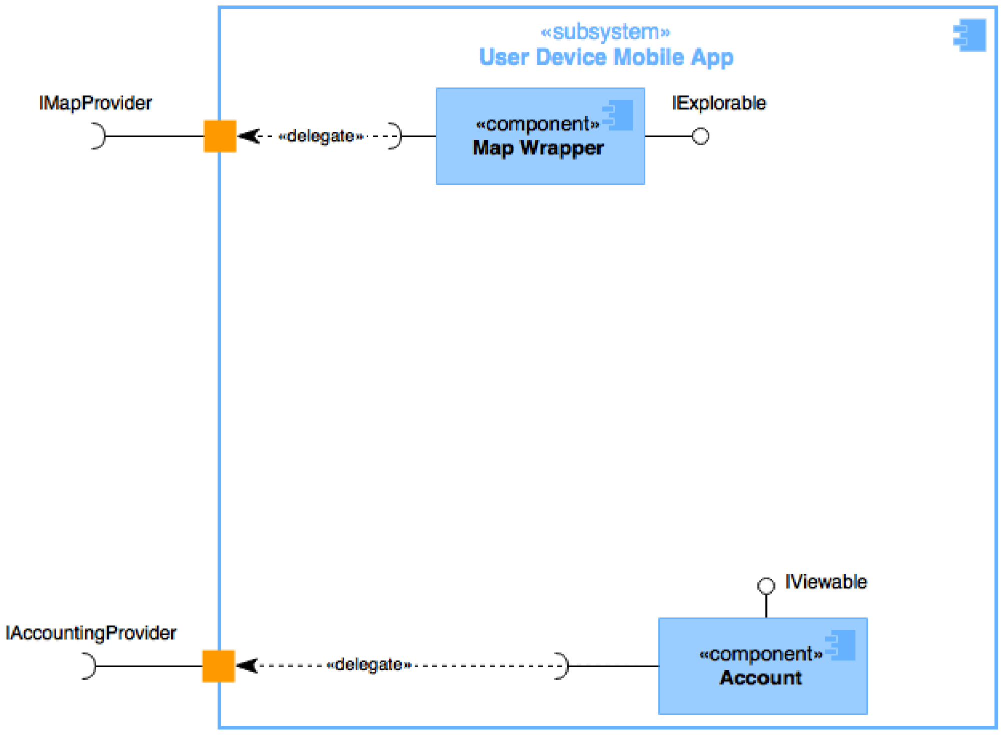

Figure 6: UDMA Phase One

Next are Map Explorer and Reserved Car Viewer. These components are tested using the actual Map Wrapper component and the data provided by a stub of CLientAPI/-Booking. See *Figure 7*.

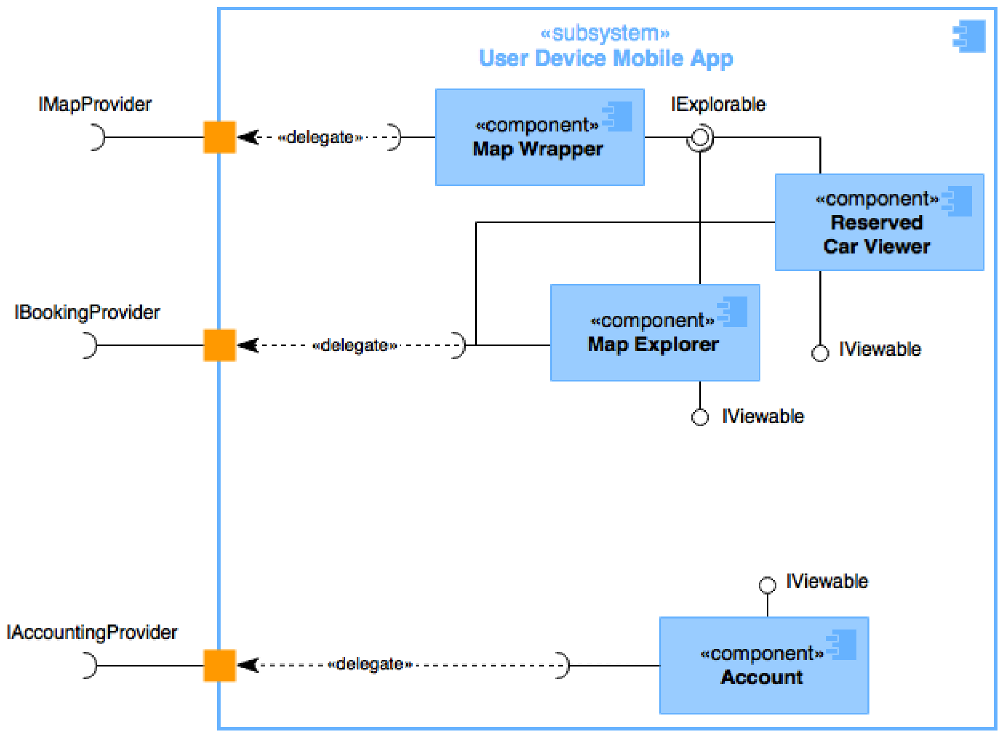

Figure 7: UDMA phase two

Lastly View and View Router, whose testing will be carried out ensuring that all possible navigation paths established in the UX Diagram contained in the Design Document are possible, and the View always provide the right information with respect to UX Diagram and the mockups in RASD. See *Figure 8*.

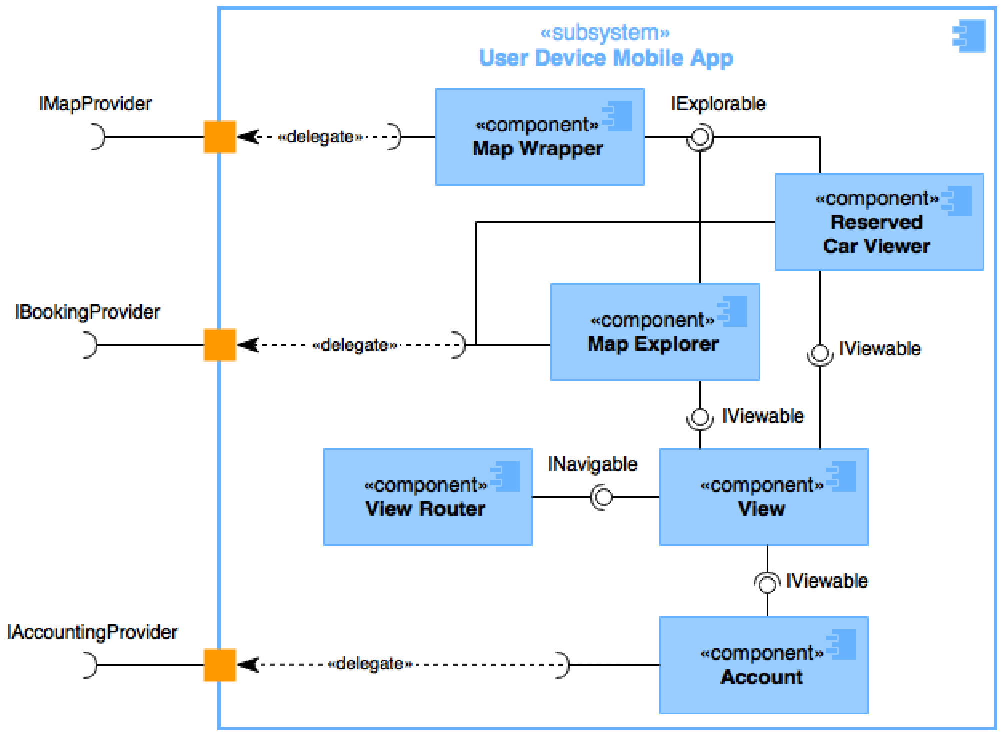

Figure 8: UDMA Phase Three

2.4.2 Subsystem Integration Sequence

After the single subsystems have been tested it is possible to integrate them. It is likely that, due to its complexity, SBL will be the last subsystem to be fully tested, which means that by the time it is tested the other two subsystems will be already tested. Therefore the integration sequence can start either from ORAA or from UDMA by integrating them with ClientAPIs, both ClientAPI/Booking and ClientAPI/Accounting where needed. After that, SBL is integrated with Live Vehicle API and ClientAPIs. The final result can be seen in *Figure 9* with all third parties components already added.

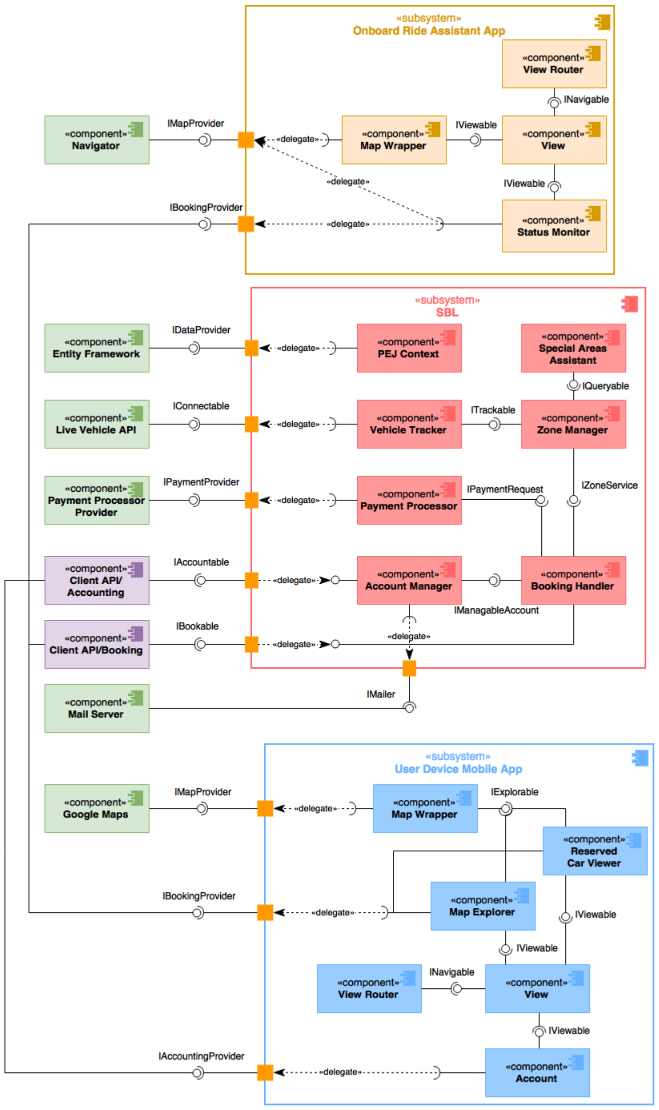

Figure 9: Integrated Subsystems

3 Individual Steps and Test Description

Here follows a exhaustive description of each component feature testing. For each feature a test suite is provided (foolowing *Component.Feature* naming convention). All tests in a test suite come at the same point in the component integration so they share the same test items and environmental needs. Input and Output specifications are respectively given in the shape of pre- and post- conditions that the single test (Tn, where n is a number relative to that test suite) requires to be matched in order to be marked as successfully passed. For better readability test items are listed with arrows that resembles interface symbols used in component diagrams, so the reader shouldn't need to continuously go back to diagrams in order to follow this section.

3.1 Service Business Logic

3.1.1 Account Manager

Test Case Identifier	AccountManager.Creation
Test Item(s)	IAccountable → Account Manager → IMailer, PEJ Context
Input Specification	**T1, T2:** valid registration form. **T3, T4:** not valid registration form.
Output Specification	**T1:** PEJ Context contains a new tuple containing the data of the registration form. **T2:** send email request sended via IMAiler interface to email address contained in registration form. **T3:** PEJ Context remains unchanged. **T4:** no email send request is made via IMailer interface.
Environmental Needs	Mail Server stub

Test Case Identifier	AccountManager.Authentication
Test Item(s)	IAccountable → Account Manager → PEJ Context
Input Specification	**T1:** valid login form. **T2:** not valid login form.
Output Specification	**T1:** a valid authentication token exists and user data are sent via IAccountable. **T2:** no new authentication token is created and an error code is sent via IAccountable.
Environmental Needs	None

Test Case Identifier	AccountManager.ProfileManagement
Test Item(s)	IAccountable → Account Manager → IMailer, PEJ Context
Input Specification	**T1:** valid modification request of driving licence. **T2:** valid modification request of personal information. **T3:** valid modification request of credit card information. **T4:** not valid modification request of driving licence. **T5:** not valid modification request of personal information. **T6:** not valid modification request of credit card information.
Output Specification	**T1, T2, T3:** data are modified and a notification is sent via IMailer. **T4, T5, T6:** data are not modified and an error code is sent via IAccountable.
Environmental Needs	Mail Server stub

Test Case Identifier	AccountManager.DetailsRetrieval
Test Item(s)	IManageableAccount → Account Manager → PEJ Context
Input Specification	**T1:** existing user ID. **T2:** not existing user ID.
Output Specification	**T1:** user ID is found and information are sent via IManageableAccount. **T2:** no user ID is found and an error is sent via IManageableAccount.
Environmental Needs	None

3.1.2 Payment Processor

Test Case Identifier	PaymentProcessor.Billing
Test Item(s)	IPaymentRequest → Payment Processor → PEJ Context
Input Specification	**T1:** ride satisfies a given discount policy. **T2:** ride satisfies several given discount policies. **T3:** ride does not satisfy any existing discount policy.
Output Specification	**T1:** a new pending payment is added in PEJ Context to the user ID contained in data and the appropriate discount is applied. **T2:** a new pending payment is added in PEJ Context to the user ID contained in data and all appropriate discounts are applied. **T3:** a new pending payment is added in PEJ Context to the user ID contained in data and no discount is applied.
Environmental Needs	None

Test Case Identifier	PaymentProcessor.RequestProcessing
Test Item(s)	Payment Processor → IPaymentProvider, PEJ Context
Input Specification	**T1:** at least one pending payment exists for a given user.
Output Specification	**T1:** a payment request is sent via IPaymentProvider and there is no pending payment for the billed user ID.
Environmental Needs	Payment Processor Provider stub

3.1.3 Vehicle Tracker

Test Case Identifier	VehicleTracker.Monitoring
Test Item(s)	ITrackable → Vehicle Tracker → IConnectable, PEJ Context
Input Specification	**T1:** a new vehicle status update has been received. **T2:** status data available for a given vehicle.
Output Specification	**T1:** vehicle status in PEJ Context is updated accordingly to the new status. **T2:** vehicle status read from PEJ Context is sent via ITrackable.
Environmental Needs	Live Vehicle API stub

Test Case Identifier	VehicleTracker.Controlling
Test Item(s)	ITrackable → Vehicle Tracker → IConnectable, PEJ Context
Input Specification	**T1:** an action compatible with current given vehicle status is requested. **T2:** a status incompatible action is requested for a given vehicle.
Output Specification	**T1:** action command is sent via IConnectable to the same vehicle. **T2:** no command is sent to any vehicle.
Environmental Needs	Live Vehicle API stub

3.1.4 Apecial areas Assistant

Test Case Identifier	SpecialAreasAssistant.Filtering
Test Item(s)	IQueryable → Special Areas Assistant → PEJ Context
Input Specification	**T1:** given position is valid and exist at least one available special parking area. **T2:** given position is not valid.
Output Specification	**T1:** a non empty list of available special parking areas within a given distance from the given location is returned. **T2:** an empty list is returned.
Environmental Needs	None

3.1.5 Zone Manager

Test Case Identifier	ZoneManager.AvailableVehiclerBrowsing
Test Item(s)	IZoneService → Zone Manager → PEJ Context
Input Specification	**T1:** given position is valid. **T2:** given postition is not valid.
Output Specification	**T1:** list of available vehicles is sent back via IZoneService. **T2:** empty list is sent back via IZoneService.
Environmental Needs	Special Areas Assistant component, Vehicle Tracker stub

Test Case Identifier	ZoneManager.VehicleStatusUpdating
Test Item(s)	IZoneService → Zone Manager → ITrackable
Input Specification	**T1:** a vehicle status update where its position is within a Special Parking Area and its plug is connected to the power grid is received. **T2:** vehicle status update where its position is within Safe Areas but not within a Special Parking Area. **T3:** vehicle status update where its position is neither within Safe Areas nor within a Special Parking Area.
Output Specification	**T1:** vehicle status is sent via IZoneService with Special Parking Area flag equals to true. **T2:** vehicle status is sent via IZoneService with Special Parking Area flag equals to false and Safe Area flag equals to true. **T3:** vehicle status is sent via IZoneService with Safe Area flag equals to false.
Environmental Needs	Vehicle Tracker component, Special Areas Assistant stub

Test Case Identifier	ZoneManager.VehicleActionHandling
Test Item(s)	IZoneService → Zone Manager → ITrackable
Input Specification	**T1:** unlock car ID whose corresponding vehicle belongs to the zone managed by that instance of Zone Manager. **T2:** lock car ID whose corresponding vehicle belongs to the zone managed by that instance of Zone Manager.
Output Specification	**T1:** unlock car request is routed to ITrackable. **T2:** lock car request is routed to ITrackable.
Environmental Needs	Vehicle Tracker component

3.1.6 Booking Handler

Test Case Identifier	BookingHandler.Creation
Test Item(s)	IBookable $\rightarrow$ Booking Handler $\rightarrow$ IManageableAccount, IZoneService
Input Specification	**T1:** chosen vehicle is available for booking and the user has no existing booking. **T2:** chosen veicle is not available for booking.
Output Specification	**T1:** a new booking instance is created and a notification is sent via IManageableAccount and IZoneService. **T2:** an error message is sent back via IBookable.
Environmental Needs	Account Manager component, Zone Manager component

Test Case Identifier	BookingHandler.Cancellation
Test Item(s)	IBookable $\rightarrow$ Booking Handler $\rightarrow$ IManageableAccount, IZoneService
Input Specification	**T1:** given user has an existing vehicle booking.
Output Specification	**T1:** booking instance for that user and that vehicle is destroyed, vehicle availability is refreshed and a notification is sent via IManageableAccount and IZoneService.
Environmental Needs	Account Manager component, Zone Manager component

Test Case Identifier	BookingHandler.Expiration
Test Item(s)	IBookable $\rightarrow$ Booking Handler $\rightarrow$ IManageableAccount, IZoneService, IPaymentRequest
Input Specification	**T1:** given user has an existing vehicle booking for which expiration timer has triggered.
Output Specification	**T1:** a bill request is sent via IPaymentRequest and BookingHandler.Cancellation is triggered.
Environmental Needs	Account Manager component, Zone Manager component, Paymant Processor component

Test Case Identifier	BookingHandler.UserVehicleUnlocking
Test Item(s)	IBookable → Booking Handler → IZoneService
Input Specification	**T1:** a request from a given user is received to unlock the vehicle from a given position within unlocking range. **T2:** a request from a given user is received to unlock the vehicle from a given position outside unlocking range. **T3:** a request from a given user is received to unlock the vehicle but no vehicle booking exists for that user.
Output Specification	**T1:** unlock request for user's booked vehicle is sent via IZoneService. **T2, T3:** no vehicle unlocking performed.
Environmental Needs	Zone Manager component

Test Case Identifier	BookingHandler.RideStatusUpdating
Test Item(s)	IBookable → Booking Handler → IZoneService
Input Specification	**T1:** given vehicle is currently in a state of ride in progress.
Output Specification	**T1:** request for car status and position info is sent via IZoneService.
Environmental Needs	Zone Manager component

Test Case Identifier	BookingHandler.AutomaticVehicleLocking
Test Item(s)	IBookable → Booking Handler → IZoneService
Input Specification	**T1:** given vehicle is currently in a state of ride in progress.
Output Specification	**T1:** locking request for the vehicle related to this booking is sent via IZoneService.
Environmental Needs	Zone Manager component

3.2 User Device Mobile App

From now on, Input and Output specifications are no longer given in the shape of pre- and post- conditions. This is done since applications components are mostly view-related and user-interaction-related.

3.2.1 Map Wrapper

Test Case Identifier	MapWrapper.MapLooking
Test Item(s)	IExplorable → Map Wrapper → IMapProvider
Input Specification	**T1:** GPS coordinates. **T2:** valid address. **T3:** not valid address.
Output Specification	**T1, T2:** scrollable and zoomable map frame of the place specified by given coordinates is returned via IExplorable. **T3:** map frame is not updated.
Environmental Needs	Google Maps

3.2.2 Account

Test Case Identifier	Account.Creation
Test Item(s)	View → Account → IAccountingProvider
Input Specification	**T1:** user filled in registration form.
Output Specification	**T1:** registration form data sent via IAccountingProvider.
Environmental Needs	ClientAPI/Accounting stub

Test Case Identifier	Account.Authentication
Test Item(s)	View → Account → IAccountingProvider
Input Specification	**T1:** login data.
Output Specification	**T1:** login data are received via IAccountingProvider.
Environmental Needs	ClientAPI/Accounting stub

Test Case Identifier	Account.Recap
Test Item(s)	View → Account
Input Specification	**T1:** cached user data.
Output Specification	**T1:** no exception occurs.
Environmental Needs	None

Test Case Identifier	Account.DataUpdate
Test Item(s)	View → Account → IAccountingProvider
Input Specification	**T1:** user filled in form containing updated data.
Output Specification	**T1:** form containing updated data is sent via IAccountingProvider.
Environmental Needs	ClientAPI/Accounting stub.

Test Case Identifier	Account.SignOut
Test Item(s)	View → Account
Input Specification	**T1:** none.
Output Specification	**T1:** login data deleted.
Environmental Needs	None

3.2.3 Map Explorer

Test Case Identifier	MapExplorer.ShowMap
Test Item(s)	IViewable → MapExplorer → IExplorable, IBooking-Provider
Input Specification	**T1:** map frame and available Cars' positions.
Output Specification	**T1:** map including Cars' positions is shown.
Environmental Needs	Map Wrapper component, ClientAPI/Booking stub

3.2.4 Reserved Car Viewer

Test Case Identifier	ReservedCarView.ShowMap
Test Item(s)	IViewable → Reserved Car Viewer → IExplorable, IBook-ingProvider
Input Specification	**T1:** map frame and reserved Car position.
Output Specification	**T1:** map including reserved Car position is shown.
Environmental Needs	Map Wrapper component, ClientAPI/Booking stub.

3.2.5 View

Test Case Identifier	View.ModelBinding
Test Item(s)	INavigable → View → IViewable
Input Specification	**T1:** ViewModel.
Output Specification	**T1:** A view is visualized with successful binding.
Environmental Needs	View Router driver, Map Explorer component, Reserved Car View component

3.2.6 View Router

Test Case Identifier	ViewRouter.Navigation
Test Item(s)	View Router → INavigable
Input Specification	**T1:** valid and consistent viewmodel. **T2:** unsupported or inconsistent viewmodel.
Output Specification	**T1:** a new instance of a view capable of binding the input viewmodel and performing navigation to that view is created. **T2:** an error pop-up message is displayed.
Environmental Needs	View stub

3.3 Onboard Ride Assistant App

3.3.1 Map Wrapper

Test Case Identifier	MapWrapper.MapLooking
Test Item(s)	IExplorable → Map Wrapper → IMapProvider
Input Specification	**T1:** GPS coordinates. **T2:** valid address. **T3:** not valid address.
Output Specification	**T1, T2:** scrollable and zoomable map frame of the place specified by given coordinates is returned via IExplorable. **T3:** map frame is not updated.
Environmental Needs	Navigator

3.3.2 Status Monitor

Test Case Identifier	StatusMonitor.SearchAddress
Test Item(s)	IViewable → Status Monitor → IMapProvider, IBooking-Provider
Input Specification	**T1:** user provided address. **T2:** Navigator recognized address. **T3:** Navigator not recognized address. **T4:** server response to safe area check.
Output Specification	**T1:** the address is sent via IMapProvider to Navigator. **T2:** the address is sent via IBookingProvider to ClientAPI/Booking. **T3:** no exception occurs. **T3:** server response is shown.
Environmental Needs	ClientAPI/Booking stub, Navigator

Test Case Identifier	StatusMonitor.SpaListing
Test Item(s)	IViewable → Status Monitor → IBookingProvider
Input Specification	**T1:** address.
Output Specification	**T1:** SPA list is returned via IViewable.
Environmental Needs	ClientAPI/Booking stub

Test Case Identifier	StatusMonitor.MoneySavingDestination
Test Item(s)	IViewable → Status Monitor → IBookingProvider
Input Specification	**T1:** address.
Output Specification	**T1:** the closest money saving destination to the given address is set as destination.
Environmental Needs	ClientAPI/Booking stub, Navigator

Test Case Identifier	StatusMonitor.ShowDiscounts
Test Item(s)	IViewable → Status Monitor → IBookingProvider
Input Specification	**T1:** none.
Output Specification	**T1:** discounts are returned and sent via IViewable.
Environmental Needs	ClientAPI/Booking stub

Test Case Identifier	StatusMonitor.StatusUpdate
Test Item(s)	IViewable → Status Monitor → IBookingProvider
Input Specification	**T1:** viewmodel.
Output Specification	**T1:** viewmodel of the actual view is update.
Environmental Needs	ClientAPI/Booking stub

3.3.3 View

Test Case Identifier	View.ModelBinding
Test Item(s)	INavigable → View → IViewable
Input Specification	**T1:** viewmodel.
Output Specification	**T1:** a view is visualized with successful binding.
Environmental Needs	View Router driver, Map Wrapper component, Status Monitor component

3.3.4 View Router

Test Case Identifier	ViewRouter.Navigation
Test Item(s)	View Router → INavigable
Input Specification	**T1:** valid and consistent viewmodel. **T2:** unsupported or inconsistent viewmodel.
Output Specification	**T1:** a new instance of a view capable of binding the input viewmodel and performing navigation to that view is created. **T2:** an error pop-up message is displayed.
Environmental Needs	View stub

4 Tools and Test Equipment Required

4.1 xUnit

Most of test that are listed in previous section of this document should (as they're designed to) be implemented by a xUnit compatible test framework depending on the application framework chosen to develop the single subsystem.

4.2 Mocking tool

Depending on chosen testing framework, the adoption of an appropriate mocking provider toolkit is strongly recommended to ease and automatize test doubles generation for not yet implemented components.

4.3 Xamarin Test Cloud

Xamarin Test Cloud will provide a set of features to run automated tests upon many different emulated user interaction cases and to profile client application performances across different types of supported devices.

4.4 HockeyApp

HockeyApp is a platform that provides a set of tools intended to assist the development team during a first stage of alpha and beta distribution such as crash reporting and other useful functions like user metrics and feedback. We plan to use HockeyApp crash reports and metrics to generate issue specific test cases once both User Device Mobile App and Onboard Ride Assistant App have been deployed to testing devices and also during future alpha and beta testing phases.

5 Program Stubs and Test Data Required

List of all program stubs and test suites which require them; stubs listed here are only those used to substitute external components of our system. Their existence is intended to allow parallel testing of multiple subsystems while decoupling different integration steps. It is recommended to re-run each test suite with the actual component before any internal or public build if feasible.

- **Client API/Accounting Stub**
  Test suites:
  Account.Creation
  Account.Authentication
  Account.DataUpdate
  Needed as long as test runs in non deployment environment and fake client applications are not available.
- **Client API/Booking Stub**
  Test suites:
  MapExplorer.ShowMap
  ReservedCarView.ShowMap
  StatusMonitor.SearchAddress
  StatusMonitor.SpaListing
  StatusMonitor.MoneySavingDestination
  StatusMonitor.ShowDiscounts
  StatusMonitor.StatusUpdate
  Needed as long as test runs in non deployment environment and fake client applications are not available.
- **Live Vehicle API Stub**
  Test suite:
  VehicleTracker.Controlling
  Needed as long as test runs in non deployment environment and fake vehicle platform is not available.
- **Mail Server Stub**
  Test suites:
  AccountManager.Creation
  AccountManager.ProfileManagement
  Needed to avoid coupling with an external or remote mail service during testing phases.
- **Payment Processor Provider Stub**
  Test suite:
  PaymentProcessor.RequestProcessing
  Needed to emulate payment transactions.

To enable various testing scenario a consistent dummy dataset must populate the PEJ Context as one of entity framework supported datasource, such as XML, SQLite, SQL-Express or static classes.
It is recommended to provide dummy data covering most of the interesting world instances. To get help into this task it is advised to build a complete model of the data context with tools like Alloy or similar and let them generate different scenarios.
Here follows a non-exhaustive list of rules that must be met by a minimal dummy dataset:

- more than one user account exists
- all existing user must have a complete profile
- more than a vehicle exists in test zone
- at least a vehicle for each status combination exists
- at least one not expired booking exists
- at least one user has pending payments
- some Special Parking Areas must belong to the testing zone
- at least one Special Parking Area has a number of parked vehicle which is less than the exclusive slots amount
- testing zone has a non empty set of Safe Parking Areas

6 Effort Spent

Teamwork	∼11h
Piccirillo Luca	∼8h
Zampogna Gian Luca	∼11h
Zini Edoardo	∼10h