
POLITECNICO
DI MILANO

# SmartCityAdvisor

# INTEGRATION TEST PLAN DOCUMENT

*Hasancan Sayılan*

# Contents

# INTRODUCTION

## Purpose

This document is made with the purpose of explaining the test process for the integration of the components. In this document, the test plan between the integrated components will be described.

## Scope

This document includes the information about the necessities that must be done before the integration test, the overall information about the components to be integrated and how should they be integrated during the proces and the information about the tools for the integration.

## Definitions, Acronmys, Abbrevitions

- RASD : Requirement Analysis and Spesification Document
- DD : Design Document
- API : Application Programming Interface
- UI : User Interface
- System : The controller and manager of the service
- Component : The subpart of the system interacting with the system.

### Reference Documents

- Assignments AA 2016-2017 April 2017.pdf (Provided by the professor of the class)
- Previous SmartCityAdvisor documents (RASD and DD)
- Integration testing example document.pdf (Example document)

## INTEGRATION STRATEGY

### Entry Criteria

There are a few conditions that must be known, applied and met to create the and manage the integration testing process thus the following criterias must be met :

- The previous documents those are RASD and DD must be completed.
- Test data must be collected and available.
- The tools which are required for the test must be acquired and available.
- All test scenarios must be written and completed.
- The components of the system which were mentioned in previous documents must be completed and available by a small percentage of tolerance which are :
    - No tolerance for Data Access Utility components meaning %100 functionality
    - At most %40 tolerance for the Client Application meaning at least %60 functionality
    - At most %10 tolerance for the SmartCityAdvisor Management System subsystem meaning at least %90 functionality
    - And finally at most %30 tolerance for the System Administration subsystem meaning at least %70 functionality.

All the percentages refer to the tolerances in the beginning of  the integration test process.

## Elements to be Integrated

This section explains which elements to be integrated together and put to the integration test. It is known from the Design Document of SmartCityAdvisor, the system is made of high-level components which maintain a spesific set of functionalities. These high-level components are User App (Includes web/browser app and large displays, Application Server (The core of the system) and the Database (the data collected by the devices such as sensors, auctators etc.) Those high-level components, as their definition points out, are made of some set of lower-level components. For the partial systems mentioned in previous section, the subsystem of those are the followings :

- Data Access Utilities
- Client System
  - SetAvailability System
- SmartCityAdvisor Management System
  - AvailabilityControl System
  - ParkControl System
  - RouteControl System
- System Administration
  - Sensor Management
  - Actuator Management

The lower-level components mentioned before, will be combined and integrated to acquire the high-level components. For a detailed integration description,

- The lowest-level components Sensor Management and Actuator Management must be integrated to obtain the higher-level component System Administration.
- The lowest-level components AvailabilityControl System and ParkControl system must be integrated to obtain the higher-level component SmartCityAdvisor Management System.

These sub-level compoenents are directly linked to the higher-level compoenents and also Data Access Utilities component since this component will collect and store the data the other subcomponents achieve and transfer it whenever it is a necessity.

Some components are already in existance so working and testing on them will be much faster and easier. Those are the followings :

- Notification System
- Remote Services Interface
- DBMS

## Integration Testing Strategy

The lowest-level components and higher-level components are now explained. In the previous section, it was mentioned that these lowest-level components will be combined and integrated to obtain the higher-level components thus a bottom-up strategy will be used in the integration testing process to acquire better results. Firstly the lowest-level components which do not rely on each other will be combined and integrated, secondly the higher-level components will be put to the integration test. Doing so, the errors and bugs that might happen in lower-levels will be avoided with this strategy thus there will be no critical errors or no looking backs when dealing with the higher-level components. For this purpose again, the critical-module-first strategy will be used. First, we will deal with the most dangerous and risky component then the other ones, doing so when we are dealing with a non-risky component, we will avoid looking backs that might happen occasionally.

## Sequence of Component/Function Integration

### Software Integration Sequence

This section describes the sequence of the integration test meaning which components will be integrated in which order will be explained.

### *Data Access Utilities*

Data Access Utilities component is going to be the first component to be integrated with DBMS because this component interacts with most partial of the rest of the system. The system requires the data collected here to function and serve as it is supposed to be. That is why this component is the most risky component and according to the critical-module-first strategy, it will be integrated with DBMS first.

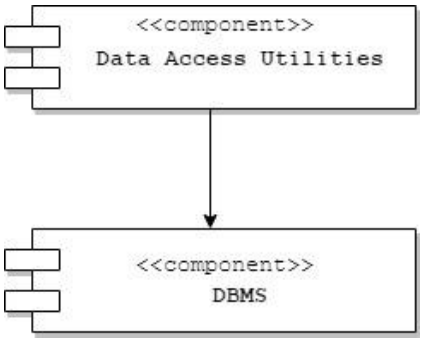


Figure 1 : Data Access Utilities Integration

### *SmartCityAdvisor Management System*

As a software functioning over a set of collected data, the Data Access Utilities component were the most risky component thus it was handled first. The second most risky component is the main component that handles the actual service of the system that is the SmartCityAdvisor Management System.

The lower-level components of SmartCityAdvisor Management System were explained in the previous section, those are AvailabilityControl System and ParkControl System and RouteControl System. Those three lower-level components will work with the data collected from the first component that was integrated so now we will integrate those subcomponents with Data Access Utilities hence the second most risky component in fact the first risky component considering without the accurate data the service wont work at all, will be handled.

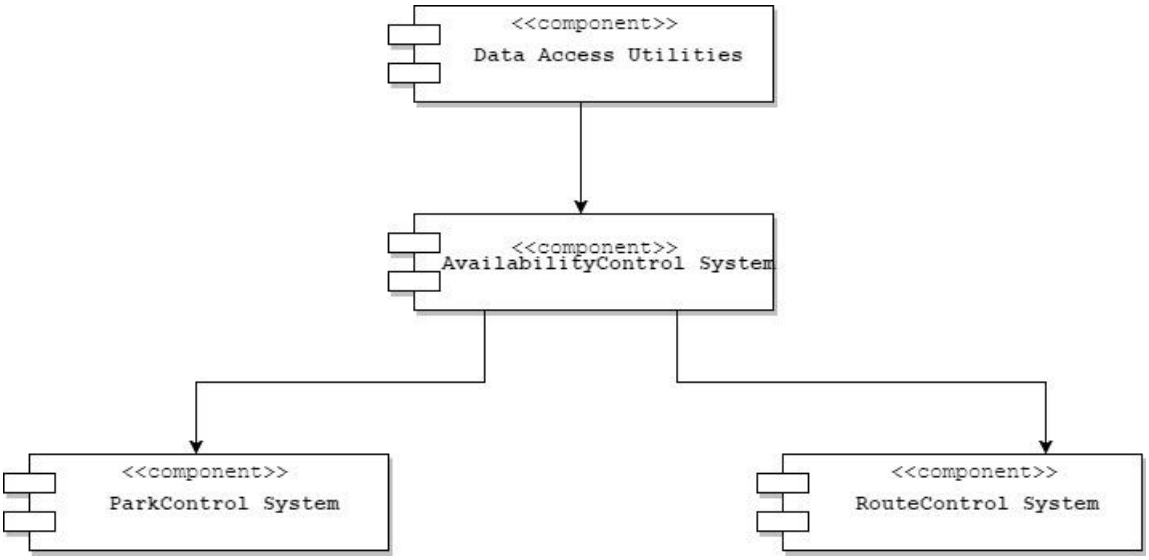


Figure 2 : SmartCityAdvisor Management System Integration

### *System Administration*

The subcomponents of this high-level component has huge impacts on the functionality and reliability of the system. Those subcomponents will be integrated after the SmartCityAdvisor Management System subcomponents due to the critical-module first strategy.

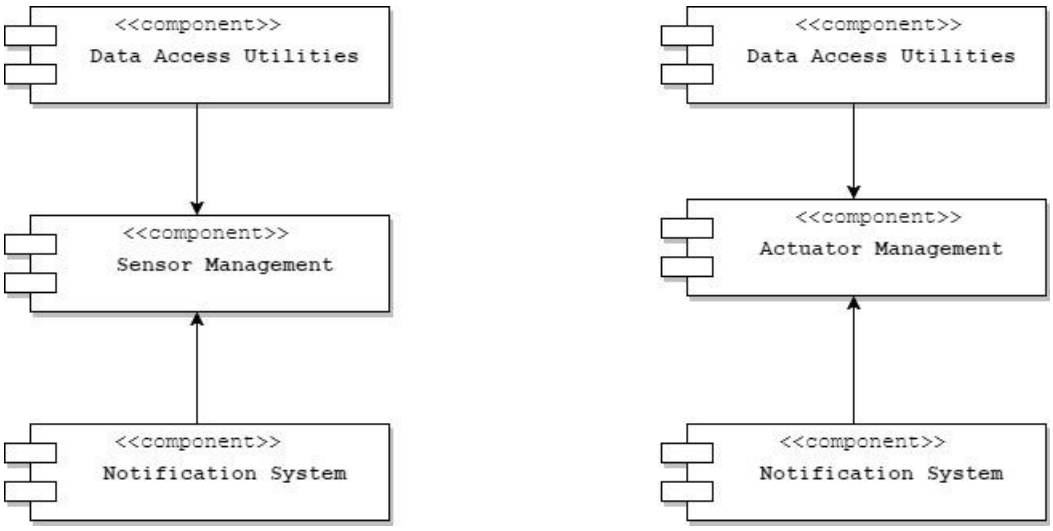


Figure 3 : System Administration Integration

### *Client System*

The last component to be integrated in the system is the Client subsystem. The functionalities and maintanence of the component Client System greatly depend on SmartCityAdvisor Management System and Data Access Utilities. Thus the subcomponent of ClientSystem that is SetAvailability that mostly works with Data Access Utilities and SmartCityAdvisor Management System.

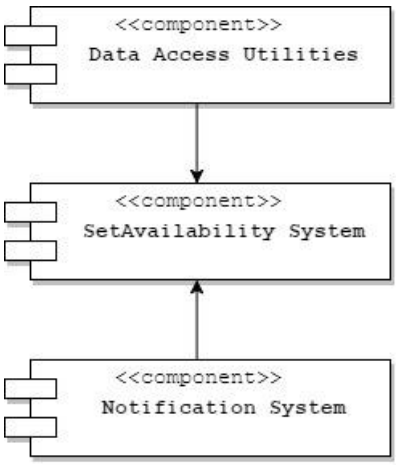


Figure 4 : Client System Integration

## Subsystem Integration Sequence

As mentioned in the previous sections, the first strategy to be used when doing integration test process is bottom-up strategy. Hence, after the completion of the integration of the lower-level components, high-level components will be integrated, those are Data Access Utilities, SmartCityAdvisor Management System, System Administration and Client System.

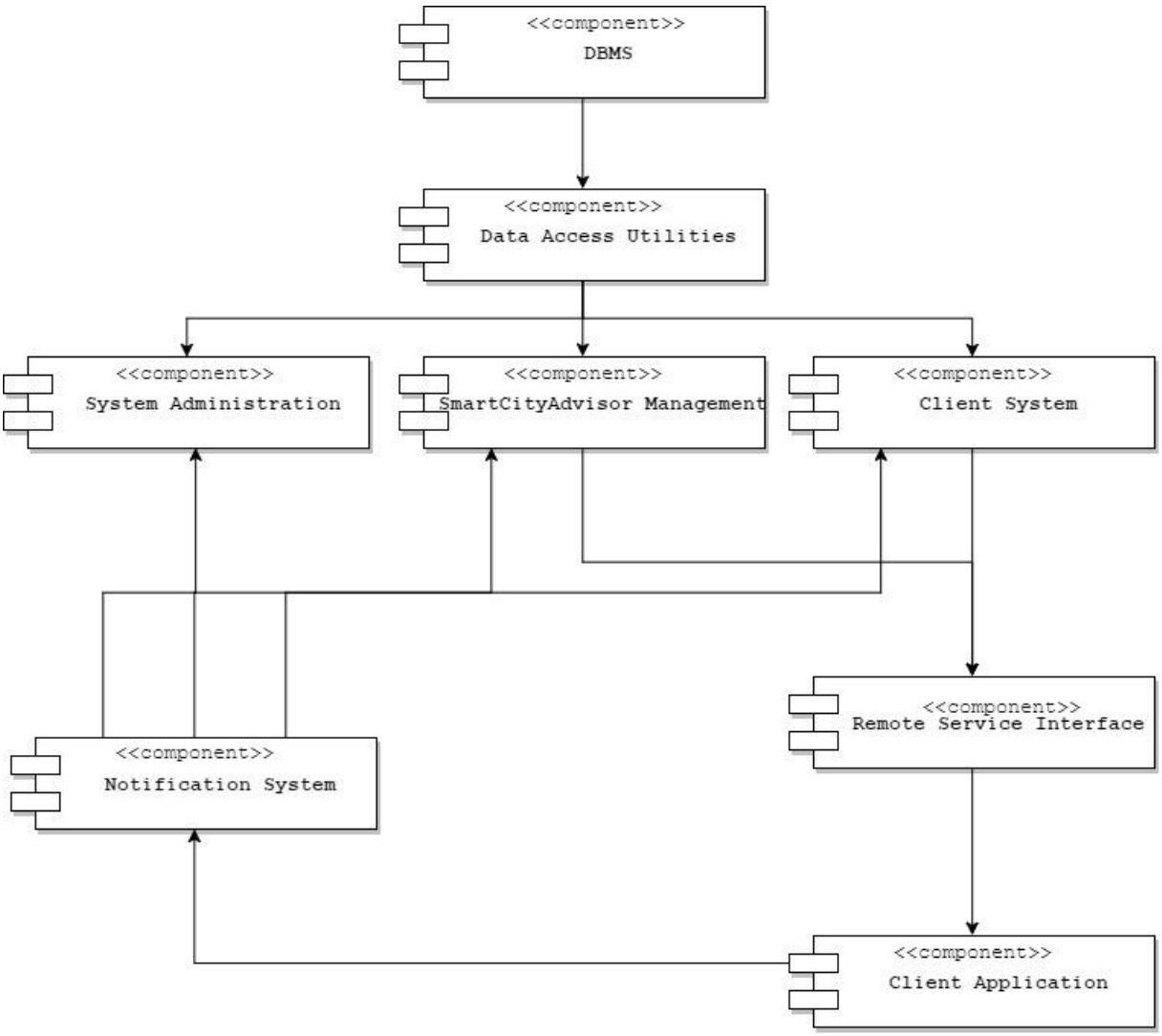


Figure 5 : High-Level Components Integration

## INDIVIDUAL STEPS AND TEST DESCRIPTION

This section will explain the tests that needs to be done. The integration of each pair of components will be described and during the tests, there will be a pair of information to be shown those are the input and the effect of that input.

A null parameter of input will result in as an effect of calling a NullArgumentException related to the subject.

### SmartCityAdvisor Management System

#### Data Access Utilities – AvailabilityControl System

| controlAvailability | |
|---|---|
| Input | Effect |
| $CO_2$ Level less than or equal to tolerance | Set availability to true, transfer it to database |
| $CO_2$ Level greater than tolerance | Set availability to false, transfer it to database |

#### AvailabilityControl System – ParkControl System

| controlParkingPlaces | |
|---|---|
| Input | Effect |
| Availability is true but no parking places | Set parkavailability to false |
| Availability is true and there are parking places | Set parkavailability to true |
| Availability is false and there are parking places | Set parkavailability to false |
| Availability is false and there are no parking places | Set parkavailability to false |

#### AvailabilityControl System – RouteControl System

| setAlternativeRoutes | |
|---|---|
| Input | Effect |
| Availability is true | Set settings to normal GPS |
| Availability is false | Set settings to alternative GPS |

## System Administration

### Data Access Utilities – Sensor Management

| calculateCO2Level | |
|---|---|
| Input | Effect |
| $CO_2$ level higher than tolerance | Transfer the information to the database |
| $CO_2$ level lower than or equal to tolerance | Transfer the information to the database |

### Data Access Utilities – Actuator Management

| controlLights | |
|---|---|
| Input | Effect |
| $CO_2$ level higher than tolerance | Set light settings to availability = false |
| $CO_2$ level lower than or equal to tolerance | Set light settings to availability = true |

## Client System

### Data Access Utilities – CheckAvailability

| setAvailability | |
|---|---|
| Input | Effect |
| Availability in the database = False | Transfer availability to client application |
| Availability in the database = True | Transfer availability to client application |

## Integration of High-Level Components

### Remote Service Interface – Client System

| checkAvailability | |
|---|---|
| Input | Effect |
| Set Show Availability | Transfered availability is transfered to Client Application Component. |

### Remote Service Interface – SmartCityAdvisor Management

| checkParkingPlaces | |
|---|---|
| Input | Effect |
| Set Show Parking Places | Parkavailability is transfered to Client Application Component |
| findAlternativeRoutes | |
| Input | Effect |
| Set Show Alternative Routes | The settings on the GPS is transfered to Client Application Component |

## PERFORMANCE ANALYSIS

The clients will connect and use this service by large displays, smartphones and computer browsers. Thus the minimum requirements for these devices must be met to be fully able to use the service and those requirements are explained in the followings :

- Large Displays positioned on the highways must be large for people to see the availability clearly and its heigh must be greater than its width to prevent accidents originated by crashing the large displays hence the height/width must be at least 2 meter / 1 meter and at most 3 meter / 1.5 meter.
- The smartphones will be the most crucial device for the clients to be able to fully use the service. Smarthphones must be at least with a single-core processor with 800Mhz or more and connected to the internet.
- The current computers are more than enough to execute the browser application but the speed of the internet must be at least 1mbps.

## TOOLS AND TEST EQUIPMENT REQUIRED

### Tools

With the purpose of increasing the test accuracy and reliability alongside with the speed, the project SmartCityAdvisor will be put on tests with automated testing tools.

Since the main programming environment for the project is Java Environment, related tools will be used during the tests.

- Firstly, JTest (Parasoft JTest) will be used during the test phase of the project. The static code analysis and data flow static analysis of JTest will be quite beneficial when testing the data flow from the data access utilities to other related components.
- Secondly JUnit will be used for the test process of SmartCityAdvisor. Its link between JAR at compile time will be most advantageous when doing repeated unit tests.

Also for performance analysis tools we will use the followings according to the related operating system :

- For Mobile Phones (Android / IOS / Windows ) : Dynatrace for application performance management.
- For Windows : Dynatrace again for application performance management.

## Test Equipment

The devices to be used during the test phase of SmartCityAdvisor project, must also met some boundaries and criterias for the test process to be fully maintained and finalized. There are some minimum conditions for devices to be used in the tests and those are :

- At least one android phone with 4'' of display size, meeting the requirements that was mentioned in the section "Performance Analysis".
- At least one IOS smartphone with 4'' of display size meeting the requirements that was mentioned in the section "Performance Analysis".
- At least one computer with a windows operation system and meets the requirements that was mentioned in the section "Performance Analysis".
- At least one large display with height and width 2m/1m and one large display with height and width 3m/1.5m.

# REQUIRED PROGRAM STUBS AND TEST DATA

## Program Stubs and Clients

In the previous chaptes, the strategy for the test phase of the project was set as bottom-up and critical-module first strategy.

Using the JUnit framework and some drives with the test process, the integration tests will begin.

- **Data Access Driver :** This module will invoke the methods which were used by Data Access Utilities, to test its interaction with DBMS and its independencies.
- **System Administration Driver :** This module will expose the methods used by System Administration subcomponents to test its interactions with SmartCityAdvisor Management and Data Access Utilities.
- **SmartCityAdvisor Driver :** This module will expose the methods used by SmartCityAdvisor Management subcomponents to test their interactions with other related components such as Data Access Utilities and the Notification System
- **Client System Driver :** This module will expose the methods used by Client System subcomponent to test its interactions with other components.

## Test Data

As mentioned in previous chapters, this testing will also include the static data flow of the subcomponents of the system. With this purpose and the samples collected by individual steps of the tests that was mentioned in previous chapter named “Individual Steps and Test description” along with some drivers mentioned in the previous chapters, the following tests will be executed :

**Park Availability :** A set of parks in the city center to use with SmartCityAdvisor Driver

- Null park availability
- Park availability with availability is set to false
- Park availability with availability is set to true

**Alternative Routes :** A set of routes to use with SmartCityAdvisor Driver

- Null routes
- Routes when availability is set to false
- Routes when availability is set to true

**Entrance** : A set of boolean results to use with SmartCityAdvisor Driver and System Administration Driver

- Null availability
- Availability when $CO_2$ level is higher than tolerance
- Availability when $CO_2$ level is lower than or equal to tolerance

## WORKING HOURS

### Hasancan Sayılan

- 05/06/2017 : 1h
- 06/06/2017 : 1h
- 09/06/2017 : 3h
- 14/06/2017 : 2h
- 15/06/2017 : 5h
- 19/06/2017 : 3h
- 22/06/2017 : 5h
- 23/06/2017 : 7h