GTA-Group

WaterMe
Software Architecture Document

Version 1.13

Revision History

Date	Version	Description	Author
28.11.2016	1.0	First Version	Paul Giesa, Olga Akymenko, Chris Todt
20.12.2016	1.11	Updated MVC	Paul Giesa
09.06.2017	1.12	Update DataView	Chris Todt
14.06.2017	1.13	Add Architecturally Significant Classes	Chris Todt
19.06.2017	1.14	Add new MVC	Chris Todt

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system “WaterMe”, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document shows the architecture of the project WaterMe.

1.3 Definitions, Acronyms, and Abbreviations

Definition/ Acronym / Abbreviation	**Meaning**
SAD	Software Architecture Document
MVC	Model View Controller

1.4 References

(n/a)

2. Architectural Representation

WaterMe will use the MVC-Principles. A framework is not used, because Android does not work according to MVC. The architectural representation has been converted into MVC manually.

3. Architectural Goals and Constraints

3.1 Patterns

As a pattern we have chosen the Observer Design Pattern.
In this pattern an object (subject) maintains a list of its dependents (observers) and notifies them automatically of any state changes.
In our case it is mainly used for button-clicks and switches.
Several On-Click listeners (for each button one) and a manager are implemented in the background deciding which button was clicked and which function to execute.

A brief example:

The class implements the interface View.OnClickListener

```
public class PlantSelect extends AppCompatActivity implements View.OnClickListener{
```

The subjects get their dependency (observer). They report to the observer when an action/update was executed.

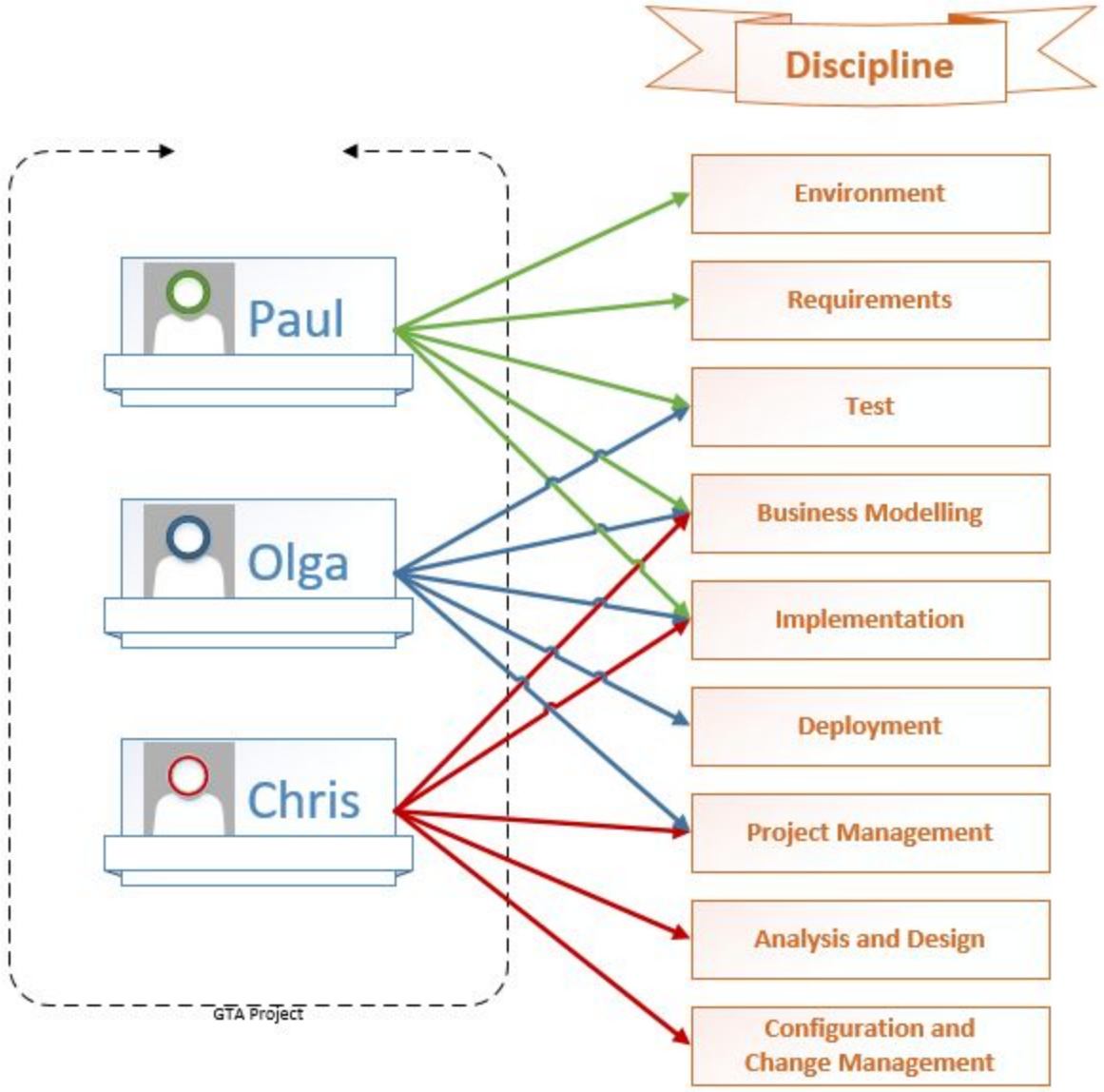
```
ImageButton del1 = (ImageButton) findViewById(R.id.btnDel1);
del1.setOnClickListener(this);
ImageButton del2 = (ImageButton) findViewById(R.id.btnDel2);
del2.setOnClickListener(this);
```

The functions which are executet by the observer

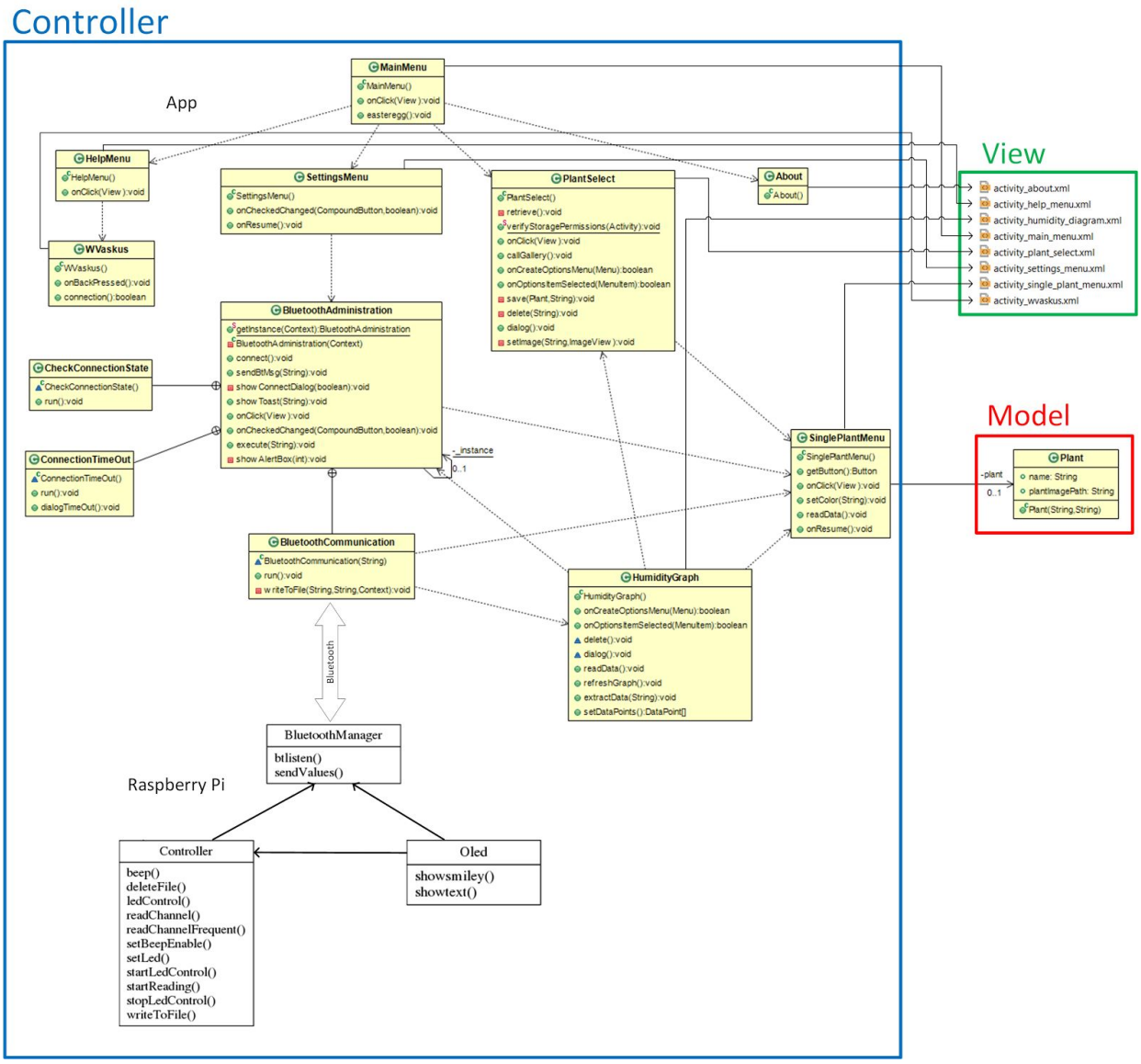
```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.tvPlant1:
            id = 1;
            startActivity(new Intent(this,SinglePlantMenu.class));
            break;
        case R.id.tvPlant2:
            id = 2;
            startActivity(new Intent(this,SinglePlantMenu.class));
            break;
```

3.2 Team structure

The team is structured according to the following graph:



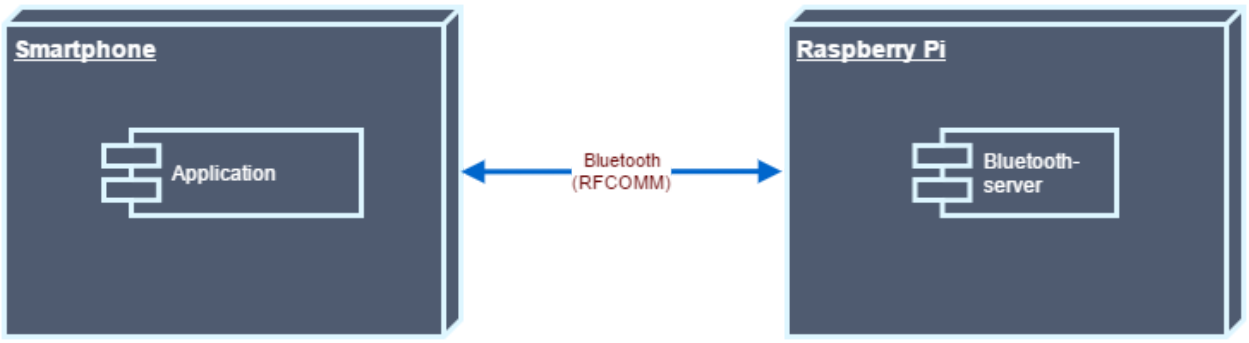
4. Logical View



4.1 Architecturally Significant Classes

- SinglePlantMenu:
  Used to store and receive measurement data from the pi.
- PlantSelect:
  Used to create new plants and store their data (e.g. name and image).
  Within this class is decided which plant is displayed in the SinglePlantMenu.

5. Deployment View



6. Data View

Android:
When creating a new plant, an object of plant, with a name and an imagepath, is saved with SharedPreferences.
These saved objects are accessed and loaded for the Text- and ImageViews in the MyPlants menu and for the menu of each plant.

Pi:
The pi writes the measured data into a textfile. There are 4 textfiles on the pi, one for each sensor.
When the app requests data the pi reads line by line and sends one after another to the app. When the app receives this data, it writes it locally into a file.
When the pi has sent one line it waits until the app responds with the receipt before sending the next line.

7. Size and Performance

It is important to keep the size of the app as small as possible due to the limited storage of smartphones.
Performance is desirable, but secondary because it does not influence the functionality.

8. Quality

Easy navigation standards have been used to provide awesome usability.