FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Integration of Software Engineering good practices for mobile development at CERN

**Inês Pereira da Cruz**


Mestrado em Engenharia de Software

Supervisor: Gil Gonçalves

July 19, 2019

# Integration of Software Engineering good practices for mobile development at CERN

**Inês Pereira da Cruz**

Mestrado em Engenharia de Software

July 19, 2019

# Abstract

The Acquisition, Processing and Control Software section, part of the Survey, Mechatronic and Measurements group in the Engineering department, is responsible for developing and maintaining the software used for acquisition, processing and control of measurements. As it is mandatory to have a smart phone per team, in case it is necessary to reach emergency services, almost all employees at the European Organization of Nuclear Research carry a smart phone and, despite smart phones getting more capable by the day, mobile development is only starting to be explored at the European Organization of Nuclear Research.

Since there is not yet a consistent and established software development process for mobile development, the main goal of this thesis is to draft and validate a set of practices, ideas and tools for mobile software development resulting in a common software development process used at the European Organization of Nuclear Research.

In order to do this, an analysis of the software development process at the Acquisition, Processing and Control Software section was done and adopted. A set of good practices for software development and specific to mobile development were introduced and an application that tests the proposed methods was developed. Techniques and methods such as refactoring techniques, continuous integration and continuous deployment, code review, among others were used.

To validate the proposed development approach, we developed Survey Measurement Acquisition in Real Time, an Android application that supports surveyors by collecting and validating measurements. It is able to connect to measurement devices and analyse the measurements, locating points or stations, a combination of points, where measurements need to be repeated, and the instruments status, such as its temperature and distance from the instrument to the levelling staff. Survey Measurement Acquisition in Real Time was tested in the tunnel of the Large Hadron Collider, where surveyors perform the measurements, and the results show that the followed process pleases surveyors due to their involvement in the project and reducing the issues of lack of communication.

With this set of practices, there is now a basis for mobile development at the European Organization for Nuclear Research, this process follows Agile methodologies and good practices for software development.

**Keywords**: software engineering, mobile development, software engineering good practices, development process

# Resumo

A secção de *Acquisition, Processing and Control Software*, que faz parte do grupo *Survey, Mechatronic and Measurements* do departamento de Engenharia, é responsável por desenvolver e manter o software utilizado para recolha, processamento e controlo de medições. Como é obrigatório haver um smart phone por equipa, em caso de ser necessário alcançar os serviços de emergência, quase todos os empregados da Organização Europeia para a Pesquisa Nuclear têm um smart phone e, apesar de smart phones terem cada vez mais capacidades, o desenvolvimento móvel só está a começar a ser explorado na Organização Europeia para a Pesquisa Nuclear agora.

Como ainda não existe um processo de desenvolvimento de software consistente e estável para desenvolvimento móvel, o objectivo principal desta tese é projetar e validar um conjunto práticas, ideias e ferramentas para o desenvolvimento de software móvel resultando num processo de desenvolvimento de software comum usado na Organização Europeia para a Pesquisa Nuclear.

Para tal, uma análise ao processo de desenvolvimento de software na secção *Acquisition, Processing and Control Software* foi feito. Um conjunto de boas práticas para o desenvolvimento de sofware e mais especificamente para o desenvolvimento móvel foi introduzido e uma aplicação que testa os métodos proposto foi desenvolvida. Técnicas e métodos como *refactoring*, integração e deployment contínuo, revisão de código, entre outros, foram utilizadas.

Para validar a abordagem de desenvolvimento proposta, desenvolvemos *Survey Measurement Acquisition in Real Time*, uma aplicação Android que auxilia *surveyors* a recolher e validar medições. É capaz de se conectar a instrumentos de medição e validar as medições, localizando pontos ou estações, combinações de pontos, onde medições têm de ser repetidas, o estado do instrumento, como a sua temperatura e distância aos pontos.

*Survey Measurement Acquisition in Real Time* foi testada nos túneis do *Large Hadron Collider*, onde os *surveyors* realizam as medições e os resultados mostram que o processo agrada mais os surveyors devido ao seu envolvimento no projecto e reduzindo questões relacionadas com a falta de comunicação.

Com este conjunto de práticas, agora existe uma base para desenvolvimento móvel na Organização Europeia para a Pesquisa Nuclear, este processo segue metodologias ágeis e boas práticas para desenvolvimento de software.

**Keywords**: engenharia de software, desenvolvimento móvel, boas práticas em engenharia de software, processo de desenvolvimento

# Acknowledgements

I want to thank Rémi for all the adventures, from showing me CERN to teaching me how to ski and a big wish that we still have many more adventures to complete.

To Gaelle, Jan, Kacper, Luisa, Tim, Nanna and Berengere, for all the great moments and for all the love they give me, a big hug full of love.

I want to thank André for giving me the courage to be who I am every single day and for all the company and love.

I would also like to thank everyone at CERN for giving me so much, especially Francis and Pierre for all the patience with me, Laure and Clément for the company when it was most needed.

To my friends, a big thank you, Pedro and Beto for being the best buddies a girl could ask for. Joana, Inês, Patty, Cláudia and Maria Carolina for being the best company in bad and good times. Roberto and Bea for always caring and helping with math and robotic stuff. Rafa and Diogo, for always being my friend, no matter how many years pass. And to Tiago a big thank you for putting up with me for the last months.

A big thank you to my supervisor Gil Gonçalves and Zé Pinto for all the knowledge shared and support.

*All the very best of us string ourselves up for love*
The National

# Contents

# List of Figures

# Abbreviations

| | |
|---|---|
| ALICE | A Large Ion Collider Experiment |
| ATLAS | A Toroidal LHC Apparatus |
| APC | Acquisition, Processing and Control Software |
| API | Application Programming Interface |
| APK | Android Package |
| CCD | Charge-Coupled Device |
| CMS | Compact Muon Solenoid |
| CD | Continuous Deployment |
| CI | Continuous Integration |
| CI/CD | Continuous Integration and Continuous Deployment |
| EN-SMM | Engineering - Survey, Mechatronic and Measurements |
| EN-SMM-APC | Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software |
| CERN | European Organization for Nuclear Research |
| IDE | Integrated Development Environment |
| LHC | Large Hadron Collider |
| LHCb | Large Hadron Collider beauty |
| MVC | Model-View-Controller |
| PDA | Personal Digital Assistant |
| PPE | Personal Protective Equipment |
| PFB | Pocket Field Book |
| EMQ | Root Mean Square |
| SDK | Software Development Kit |
| SDLC | Software Development Life Cycle |
| SMART | Survey Measurement Acquisition in Real Time |
| UI | User Interface |
| VCS | Version Control System |

# Chapter 1

# Introduction

As Sommervile refers in [39], software can be defined as computer programs and the associated documentation and that, as we are aware, modern world can not run without software.

Software Engineering is described by Sommervile as an engineering discipline concerned with all aspects of software production from system specification to maintaining the system after put into production and a software process as a set of related activities that lead to the production of a software product [39].

In order to have cheaper, better software delivered faster, it is important to have a defined process of development that adjusts to both the product and the team developing it and that can be adapted to increase the value of the product and the productivity of the team. As Sommervile states, the development process implemented to create a software system influences the quality of that system and by improving the software development process, the quality of the software will also be improved [39].

Application development for mobile devices has been around for some time now, the first mobile phone with the Android operational system was released in 2008, despite that, the demand for mobile applications is always increasing, yet, there is still a lack of research when it comes to the research of activities in development process of mobile applications [24].

## 1.1 Context

The European Organization for Nuclear Research (CERN) [1] was established in 1954 to unite European scientists and share the costs of nuclear physics facilities. CERN is located at the swiss-french border in Meyrin, near Geneva and has 23 member states. The main research area is particle physics and for this, the interaction of particles is studied with accelerators and detectors. The Large Hadron Collider (LHC) is 27km accelerator, shaped as a ring, where protons or ions are accelerated and collided at four interaction points, also known as "the experiments": A Large Ion Collider Experiment (ALICE), A Toroidal LHC Apparatus (ATLAS), Compact Muon Solenoid (CMS) that can be seen in Figure 1.2 and the Large Hadron Collider beauty (LHCb) experiment.

[1] https://home.cern/

The accelerator has two vacuum pipes in which one beam is accelerated clockwise and the other anticlockwise. The LHC is the most famous accelerator but not the only one at CERN. Figure 1.1 shows the whole accelerator complex.

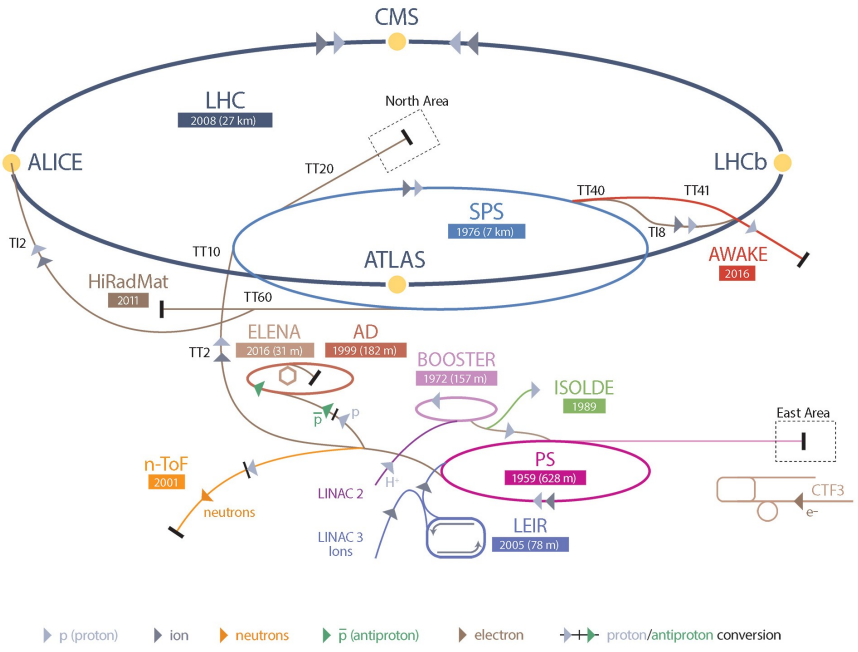


Figure 1.1: CERN's Accelerator Complex

These accelerators operate at high energy levels almost without stopping, so maintenance periods are required to properly align all the elements inside the accelerators that move with the passage of the particles. Inside these accelerators there are magnets that need to be repositioned and realigned in order to verify that they oppose the least resistance to make the particles' trajectory optimal.

When the accelerators at CERN stop, a team of surveyors is sent to verify fixed points in the magnets present in the tunnels, as shown in Figure 1.3, these measurements will tell if and what mechanical adjustment are needed to be done on the field. Since precision is key, measurements to the same point are done at least two times, making this a hard task that takes a long time, and which requires surveyors to annotate the data on the field.

At the Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software (EN-SMM-APC) section, a team of surveyors and developers work together so the acquisition and processing of this data is done faster and less prone to human errors.

## 1.2 Motivation and objectives

Surveyors need a tool to help their work with record keeping, as not only there are many measurement types, they need to collect data, process it, see what adjustments in the tunnels are needed and go down to the tunnels to perform those adjustments. As surveyors need a tool they can carry

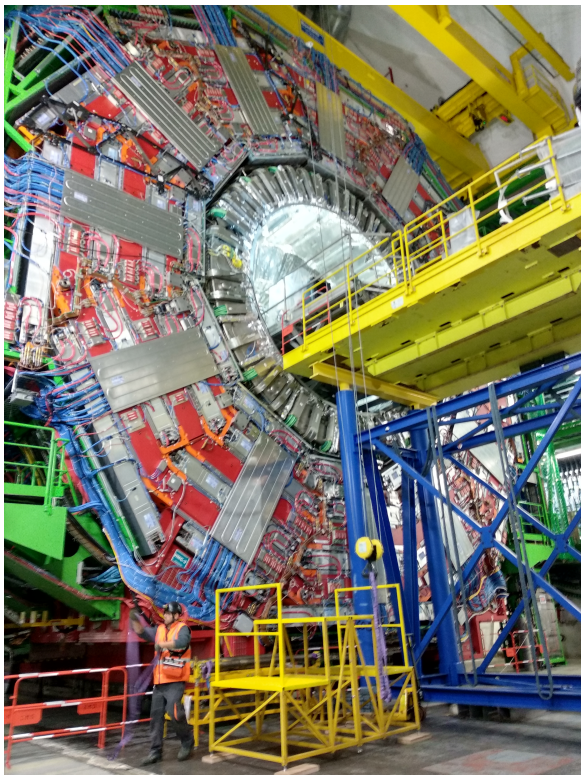

Figure 1.2: Part of the CMS - Compact Muon Solenoid

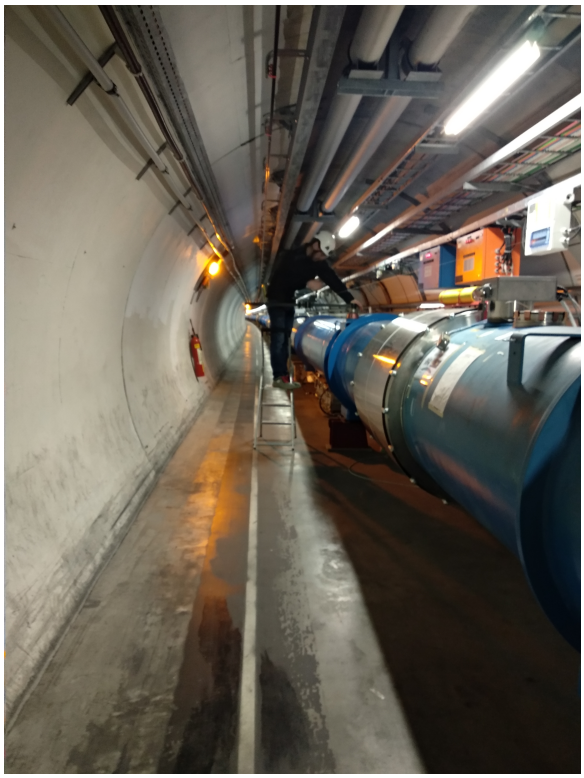

Figure 1.3: Surveyor on the LHC-tunnel

in the tunnels and that can be used in a simple way, a mobile application would fit their needs. The Acquisition, Processing and Control Software (APC) section develops applications mostly in C++ and no mobile application has previously been developed. Due to this, a process needs to be designed to help the team deliver a product that meets the users needs.

Since mobile development in the whole of CERN is scarce, it is important to devise a development process that others can follow and adapt to their needs. The proposed process needs to take into consideration good practices as one of the goals of the section is to find a way to improve software quality of future developments and to use this process for different software products.

The main objective of this thesis is to propose and implement a common software development process based on a set of practices, ideas and tools that can be used in order to improve the productivity of teams and the quality of the code developed with respect to mobile development. To validate the development process it is also an objective to develop a handheld application with the same process.

## 1.3 Structure of the Thesis

This thesis is structured as follows. Chapter 2 describes the state of the art of good practices for software development, some of which being implemented at CERN and the technologies being used in the section. Chapter 3 introduces the problem that this dissertation addresses, addressing the states of mobile development on both CERN (as a whole) and specifically the APC section. Chapter 4 describes the analysis of good practices for software development and changes to the development process at the APC section. It describes all the work done to improve the software development process found. Chapter 5 discusses the results of implementing a development process and the results of the final product. Chapter 6 discusses the findings of this dissertation, the obstacles found, future work and further improvements.

# Chapter 2

# State of the art

In this chapter we present the related work, describe existing good practices for software development, some of which being implemented at the European Organization for Nuclear Research (CERN) and the technologies to be used.

## 2.1 Software Engineering

According to Sommervile, *"Software engineering is an engineering discipline that is concerned with all aspects of software production."* and that *" software engineering is concerned with the practicalities of developing and delivering useful software."* [39]

There is not a fixed set of techniques and methods that we can call the best software engineering techniques and methods as for different systems, different techniques exists.

## 2.2 Good Practices for software development

The upcoming sub-sections list a set of good practices for software development. They are part of the Software Development Cycle (SDLC).

### 2.2.1 Prerequisites analysis

To ensure that all prerequisites are satisfied, it is necessary to verify that the life cycle, requirements, architecture of the software system and design are defined [5].

To define them is part of the software development cycle. It is necessary to analyze the envisioned product and know the development team to base the choices done in what adapts best to the development team and leads to the envisioned final product.

For example, in [29], the authors refer that, to apply the waterfall model, a set of prerequisites should be fulfilled, such as familiarity with domain, methods, techniques, tools, engineering processes, very good understanding of the requirements, stable requirements and high capabilities for effort estimation.

#### 2.2.1.1 Life cycle

The life cycle of the application can show how the development is structured. It depends on the process adopted, it can be the waterfall model, an agile methodology or an iterative spiral approach, among others. The main difference between these methodologies is the SDLC, the waterfall model is a sequential development approach as it assumes that the requirements can be defined at the start of the project, while the other methodologies try to reduce the impact of requirement changes during the development process.

In [43], Beng Leau, Khong Loo, Yip Tham and Fun Tan do a comparison between Agile an Traditional approaches. The authors reach the conclusion that agile SDLC have more advantages than traditional SDLC, but that agile SDLC also has its disadvantages, for example, they conclude that agile is more suitable for small to medium teams and, for large-scale projects, it is still better to adopt traditional SDLC. It is important that the development methodology chosen is the one that best suits the project.

#### 2.2.1.2 Requirements

Requirements describe the services a software must provide and the constraints under which it must operate. Techniques to obtain these requirements differ between agile and traditional approaches, Paetsch, Eberlein and Maurer conclude that the techniques used in agile development processes can be very vaguely described and that the real implementation is done by the developers and that in traditional approaches they are more detailed, providing more guidance but that determining upfront what is needed to be done in the context of a project can be very difficult [32].

But it is certain that gathering requirements effectively is the basis for having a finished product that meets the clients expectations and objectives. Rongala refers that it is imperative for the stakeholders to have discussions and agreement on documented requirements, that functional and non-functional requirements need to be defined, stating that "Functionality can be captured via the use-case scenarios. Performance, fault tolerance, system, design and architectural requirements should also be well-addressed." [35].

#### 2.2.1.3 Architecture

As N. Medvidovic and R. N. Taylor refer in [26], software architecture is defined as the set of principal design decisions about the system.

A good architecture needs to take into consideration the requirements and, limitations and constraints of the system. It must engage software systems design and development.

The architecture of a software system needs to be given attention over its entire lifespan so that the system's development and long-term evolution can be effective and efficient.

#### 2.2.1.4 Design of the system

In [16], Martin Fowler states that design allows software changes easily in the long term and as design deteriorates, so does the ability to make changes effectively. This deterioration will lead to software harder to change and bugs harder to find and safely kill.

Building a flexible design will allow to deal with changing requirements, this depends on what type of changes to expect.

#### 2.2.1.5 Use of Design Patterns

As stated in [2] by Alexander Shvets, a design pattern is a solution, that is general repeatable, to general problems that software developers faced during software development.

In [19], Gamma, Helm, Johnson, Vlissides define 23 design patterns. Each is a description or template to solve a general problem. Its main idea is to allow the reuse of design information, thus allowing better communication.

#### 2.2.1.6 Defining Coding Standards

Coding standards define a set of guidelines for developing software code. The benefits include the consistency of the code throughout the solution and assures the software quality at the source code level. Although these rules are used to develop a more maintainable code and improve its consistency throughout the whole project, some studies show evidence that suggest that the compliance with coding standards in software development can improve code quality, reduce program errors and in case of a team project, enhance team communication [33].

The definition of coding standards is important, especially if the project is to be developed by more than one developer. This is also important to have maintainable code[33].

Coding conventions can include information as the description on how comments should be written, definition of proper naming conventions, write simple code that can be understood by others and having documentation.

Documentation helps to maintain understanding of the software and it can be used as a reference in future for users or developers.

### 2.2.2 Code development phase

A list of good practices is required to serve as guideline during the software development, they will be studied in the following sub-sections. An example are agile methodologies practices such as continuous integration (CI) and continuous deployment (CD) that are used to allow a more iterative and incremental software development [41].

#### 2.2.2.1 Commenting

Comments in the code make the code more readable for other developers, and help them understand the code for future modifications or reuse. In [40], Steidl, Hummel and Juergens prove that

code with proper comments is easier to understand than uncommented code and consider this artifact the second most used for code interpretation. Comments should follow good naming practices, the intent of the code should be written and the reason why the system has that behaviour.

#### 2.2.2.2 Continuous Integration

Continuous integration is a software development practice that contributes to the improvement of software quality and the reduction of risks.

On each commit there is a verification by an automated build that runs tests to detect integration errors between the different modules of the application. An example of CI can be found in the following article [17].

CI is a good way to catch bugs more efficiently due to the automated tests that run during the automated build. They make the bug fixing process easier by showing the exact location where the problem happened and notifying the author of the commit that introduced the bug timely over email.

CI also allows for a development team to stay synchronized with each other, removing delays that occurred due to integration issues.

#### 2.2.2.3 Continuous Deployment

Continuous deployment is a software development practice that ensures that all changes that passed the automated build and automated tests are deployed to production automatically.

This allows for continuous customer feedback, as many times developers focus on wrong features or a task might not be clear, with this feedback it is possible to eliminate the work that does not produce value for the customer. In [30], Olsson, Allahyari and Bosch study a multiple-case exploring barriers in the transition from agile development towards continuous deployment of software.

#### 2.2.2.4 Testing

Software testing is the evaluation of the software against requirements gathered from the users and system specifications.

Automated tests are crucial in the implementation of a continuous integration and continuous deployment (CI/CD) to ensure that the software behaves as intended and to check for the existence of bugs [12].

#### 2.2.2.5 Code inspection

Fagan describes inspections, in [14], as a formal, efficient, and economical method of finding errors in design and code.

Code inspections allow an improvement in software quality and code productivity and the reduction of errors. The main goal of code inspection is to identify defects in the code. Two

examples of types of inspection are code review and peer reviews. The authors of [34] offer a comparative study of software inspection techniques for quality perspective in which they found that Fagan's methodology, shown in [14], is considered as a base of formally based inspection approaches.

#### 2.2.2.6 Refactoring

According to Martin Fowler, *"Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior."* [18]

Via refactoring, the design of the code gets improved, as it cleans up code and minimizes the chances of introducing new bugs, that is if the refactoring is well made. Refactoring code, even the smallest change, can have a big impact in the code design. If well done, refactoring can change an initially bad design into well-structured code [18].

### 2.2.3 Assuring Software Quality

According to [38], software quality is the degree to which a software product meets established requirements and that quality depends upon the degree to which those established requirements accurately represent stakeholder needs, wants, and expectations. Software quality assurance is a set of activities that define and assess the adequacy of software processes to assure that the software processes are appropriate and produce software products of suitable quality for their intended purposes [3].

### 2.2.4 Android development

In [24], Kaleel and Harishankar do an analysis of Android development and its challenges and conclude that when developing for Android, one should take into consideration that these devices are resource-constrained with storage capacity and battery lifetime and thus, performance will be a problem to have in consideration when developing mobile applications.

Good practices for Android development should be applied having in consideration the problems when developing Android applications and that the increase of efficient programming is one of the main goals.

In [9], the author concludes that Java programming practices can be a source of a memory problem for Android.

## 2.3 Good practices applied at CERN for software development

CERN is composed of departments, each with multiple groups and sub-sections. The organization of their projects and developments greatly depends on the field (engineering, computer science, physics of accelerators, theoretical studies and so on). At the Engineering - Survey, Mechatronic and Measurements (EN-SMM) group, the software and hardware developments are mainly based on Agile methodologies, especially on a Kanban-based management of the activities. It can be

thought of as a large, prioritized to-do list. Requirements in Kanban are tracked by their current stage in the process (to-do, in development, in test, done, among others). When a developer is ready for the next task, he/she pulls it from the to-do list. Moreover, a few end-users of the team (surveyors in our case) have full access to these tasks lists in order to facilitate the organization of their tests.

The practices adopted at Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software (EN-SMM-APC) section will be described further on.

### 2.3.1 Mobile development at CERN

At CERN, mobile development is starting. There is not a set of practices to follow when developing mobile applications and the existing mobile applications are written in ReactJS and are mainly for administrative proposes.

## 2.4 Surveying at CERN

To ensure the precise alignment of the Large Hadron Collider (LHC), the CERN survey team performs regular measurements in the LHC-tunnel. Our section is responsible for the metrology and alignment of the accelerator components and their associated beam transfer lines. This means acquiring, among others, levelling measurements, roll angles and offset distances on CERN installations.

More about measurements at the LHC can be consulted in "The Alignment of the LHC" [28]. The type of measurements performed by surveyors of this section will be presented in chapter 3.

## 2.5 Android Platform

Android is a mobile operating system developed by Google. It has been the best-selling OS worldwide on smart phones since 2011. It is an open source software that is available through Java Application Programming Interface (API) created for a wide range of devices. It consists of an operating system, a middleware and a set of basic applications. According to [4], by 2020, 70% of the world will own a smart phone. There are more than 24,000 Android devices made by nearly 1,300 different brands. By providing a free and open OS, Android has helped proliferate affordable mobile devices around the world. In 2015, the average Android device cost $208. That's in stark contrast to phones on closed platforms, which can average as high as $651.

The last version of the operating system is Android 9 (Pie). It has new features for accessibility, battery, display, system usability enhancements and many others that can be consulted in Android's webpage [1].

Some key concepts are presented regarding the development of applications in Android. These key concept were adapted from the documentation by Google for Android Developers [11].

### 2.5.1 Activity

As stated in [11] by Google, an Activity is an application component that interacts with the user. Every activity has a window that shows the user interface and has a specific use.

### 2.5.2 Activity Life cycle

In the documentation by Google for Android developers it is shown that activities are managed as a stack. If an activity is on top of the stack, it is active or running. If it has lost focus but is still visible, it is paused. If it is obscured by another activity it becomes stopped, meaning that it retains all state and member information but it is no longer visible to the user. An activity that is paused or stopped can be cleared by the system and it should be verified if the application can handle those cases. If the application is to have a life cycle as described here it should extend all the methods in the diagram in Figure 2.1. The square rectangles represent callback methods and the colored ones are major states the Activity can be in.

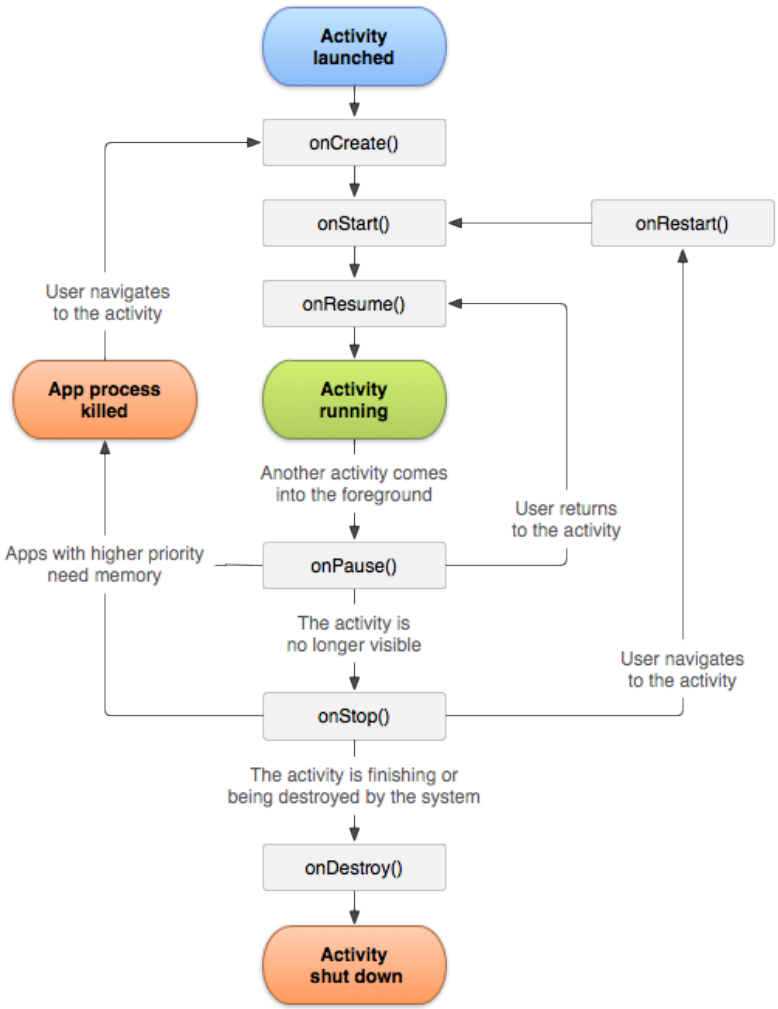


Figure 2.1: State paths of an Activity [10]

## 2.6 Technologies used

When choosing the technologies for this project it was taken in consideration that these had to be open-source or free of use for open-source projects, that their services could be hosted in CERN's servers and when choosing between two technologies for the same purpose, choose the one already used or implemented at CERN. The technologies must also run in Linux, MacOS and Windows.

### 2.6.1 GitLab

Git is a distributed Version Control System (VCS). As a tool it is to be integrated in the project's workflow. With Git it is possible to manage project files, by saving them on a server (security) and keep a track of the modifications (history). GitLab is a web-based manager for Git-repositories. It provides issue-tracking and Continuous Integration and Continuous Delivery pipeline features [1]. It is an open source software which allowed CERN to have its own instance of GitLab on CERN's servers[2]. GitLab is a streamlined solution for code review, it has the capacity to get a large number of projects and users up and running quickly, which was also a challenge for CERN. This partnership has allowed the increase of visibility and access to research, improved code quality and documentation and reusable research [20].

GitLab also allows CI/CD with Android using Docker, that will be referenced in section 2.6.3. With GitLab's continuous integration we can have an automatic build, automatic tests and automatic deployment (in our project a mobile Android Package (APK)).

### 2.6.2 Android Studio

Android Studio is Google's Android operating system official IDE. It was built on JetBrains' IntelliJ IDEA software and designed specifically for Android development [13]. It has a set of features that makes developing Android applications more developer friendly, like an APK Analyzer that inspects the contents of the APK file created for deployment, an emulator, so testing is possible even without a physical device, an intelligent code editor that helps developers write better code, a flexible build system powered by Gradle and realtime profilers, amongst others [3].

Since Android Studio has testing tools and frameworks for testing applications with JUnit5 and functional user interface (UI) test frameworks and allows code inspections, VCS with Git, code review and refactoring, it was chosen as the IDE for our project.

### 2.6.3 Docker

Docker is a platform for developers to develop, deploy, and run applications with containers.

As explained in [23] *"[...] A container is a runtime instance of an image–what the image becomes in memory when executed (that is, an image with state, or a user process). A container*

[1] https://about.gitlab.com/product/continuous-integration/
[2] http://gitlab.cern.ch
[3] https://developer.android.com/studio/

*runs natively on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight."*

Since Docker container are a standardized unit for software development that includes everything for the software and its dependencies to run such as the code, runtime, system tools and libraries, it guarantees that applications will run the same and it makes collaboration as simple as sharing a container image [23].

A docker image for projects in C++ and other programming languages already exist at CERN, making it easier to include Docker in many projects. Since Docker is already used at CERN but there is no integration for Android, we decided to use it in this project and create a container image so future mobile developers can integrate Docker in their projects easily.

The docker image created will be used as a job runner (or worker) for Gitlab continuous integration. Its goal will be to compile, run tests and automatically build a signed APK for the SMART application. The image will contain all the prerequisites to build and run the Android Software Development Kit (SDK) so there will not be a need to re-install them at each run. For the Android SDK to run it is necessary to install the Android SDK tools, Platform tools and Build Tools.

This will allow to have an automatic build, which means that as soon as a commit is done to the project repository, the image will automatically be built thanks to Gitlab-CI. It will be possible to use the image as a runner for other project's CI.

### 2.6.4 JIRA

JIRA is a software for issue and project tracking. According to [7], it's the most used software development tool by agile teams. This software is chosen by many teams at CERN as it combines agile project management with issue tracking and customizable workflow [27].

For this project it was required a tool that would allow users to create tasks associated with improvements they desired or even bugs they found during tests and to write down and update their list of features with a priority ordering. For the developers it was necessary to update the state of tasks to track which were being worked on. Moreover it was necessary a tool that integrated with source control programs, such as Git and that would be free for use for open-source projects. Since JIRA allows all of the previous, as explained in [21], it is possible to take a list of features, assigned to a team and later the team members assign the individual tasks to themselves, and also allows the hosting of JIRA projects on the central issue tracking instance of CERN[4] it is an obvious choice for any team at CERN.

[4] http://information-technology.web.cern.ch/services/JIRA-service

# Chapter 3

# Problem statement

In this chapter, the problem statement is presented. In Section 3.1 an overview of the software development at the Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software section (EN-SMM-APC) is presented, followed by an overview of surveying, focusing on levelling measurements, roll angle adjustments and offset distances, in Section 3.2. In Section 3.3, a brief description on mobile development at the European Organization for Nuclear Research (CERN) and at the Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software section (EN-SMM-APC), giving focus on the Android application Survey Measurement Acquisition in Real Time (SMART) and what requirements the application is not fulfilling and the main problems surveyors have while using the application, such the lack of validations while doing measurements. These problems are due to the lack of a consistent and established software development process for mobile development both in the section and at CERN. Section 3.4 shows what is expected to be achieved with this dissertation, which will later be described in Chapter 4 with more detail.

## 3.1 Software development at the EN-SMM-APC section

The EN-SMM-APC section teams follow a Kanban methodology, composed by small teams of 2 or 3 elements.

Each team is free to choose the technologies and processes to use on their project, the only constraint is that the technologies need to be open-source or for CERN to have a licence. Each developer is responsible for choosing their own user stories from the Kanban board to develop. As the software developed is to help surveyors on their work, they are mainly the product owners. The product owners are responsible to fill the Kanban board with their user stories and are responsible for testing the user stories developed by development teams. Developers and product owners work together to define the requirements, life cycle, architecture and design of the product.

## 3.2 Surveying

As stated in Chapter 2, to ensure the precise alignment of the Large Hadron Collider (LHC), the CERN surveying team performs regular measurements in the LHC-tunnel.

As P. Valentin states in [31] "*LHC accelerator and its injectors represents about 22 000 elements to measure and align.*".

How do surveyors know which elements have been measured or what elements they have to measure? Each element has a set of fixed points to be measured, these points are identified as, for example, LHC:MB.A1Q11.E where LHC represents the zone, MB the class, A1Q11 the number and E if it is an Entry point or an Exit point, represented by S (*Sortie*).

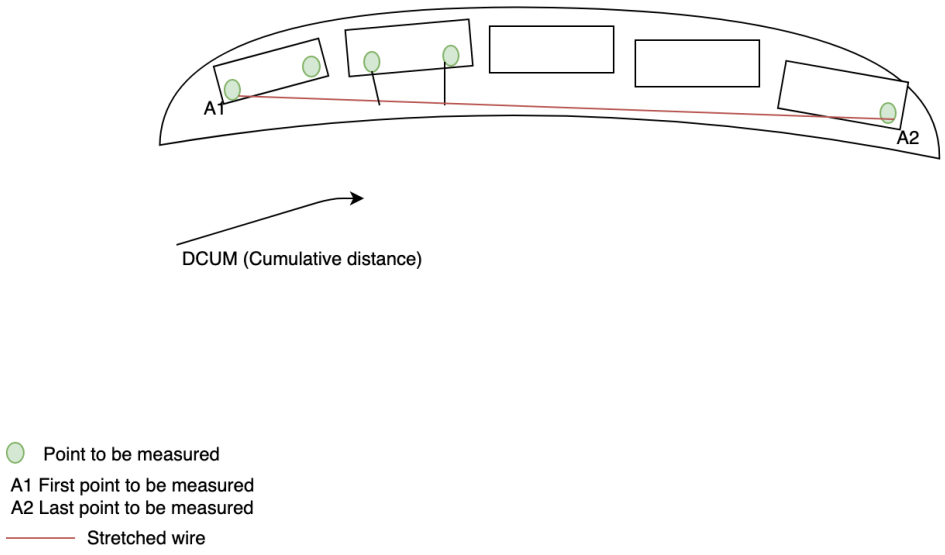


Figure 3.1: Example of elements and points in the LHC-tunnel

In Figure 3.1 a diagram of a part of the LHC-tunnel is presented where A1 is the first point to measure and A2 the last. The elements are represented by the boxes and the dots the points to measure. The red line is a stretched wire used to do offset measurements that will be discussed further on.

The measurements to these points are collected and compared with values that are stored in files, these files are called by surveyors as "theoretical files" and they contain the list of all points in all elements of the accelerators and their known coordinates. If the measurements taken don't match the values of the theoretical file, then the element needs to be aligned. Theoretical files have information regarding the points, such as the cumulative distance, which can be seen in Figure 3.2.

The points in the theoretical files are ordered by cumulative distance, this is helpful so when surveyors are in the LHC-tunnel they can have a better orientation as there are no signs or labels visible. Sometimes surveyors do measurements in a specific order, different from the cumulative distance, in this case, a sequence file can be created, this file contains the points they want to measure in that specific order.

```
TT1       ;ISR.405.                          ;    -.03254; 1753.43669; 2060.07518;2434.94274;434.94682;      ;
TT1       ;R.1A.                             ;     .00000;     .00000;     .00000;2399.34333;400.00000;      ;
TT1       ;R.1B.                             ;     .00000;     .00000;     .00000;    .00000;400.00000;      ;
TT1       ;SAT.769-1.                        ;     .00000; 1632.00000; 2017.50000;2434.38968;434.40000;      ;
TT1       ;SAT.769-2.                        ;     .00000; 1671.00000; 2031.00000;2434.39197;434.40000;      ;
TT1       ;SAT.769-3.                        ;     .00000; 1735.00000; 2056.00000;2433.99515;434.00000;      ;
TT1       ;SAT.858-1.                        ;     .00000; 1632.00000; 2017.50000;2434.38968;434.40000;      ;
TT1       ;SAT.858-1B.                       ;     .00000; 1632.00000; 2017.50000;2434.38968;434.40000;      ;
TT1       ;SAT.858-2.                        ;     .00000; 1671.00000; 2031.00000;2434.39197;434.40000;      ;
TT1       ;SAT.858-3.                        ;     .00000; 1735.00000; 2056.00000;2433.99515;434.00000;      ;
TT1       ;TRIPOD.1.                         ;     .00000; 1736.00000; 2056.00000;2434.11934;434.12414;      ;
TT1       ;ISR.405B.                         ;     .24920; 1753.89600; 2057.96240;2434.89582;434.89990;      ;
TT1       ;WPS1.1.                           ;    1.10000; 1752.19510; 2060.18140;2434.27717;434.28130;      ;
TT1       ;WPS3.1.                           ;    1.10057; 1752.15080; 2060.30346;2434.22559;434.22972;      ;
TT1       ;HLS.1.                            ;    1.10058; 1752.11973; 2060.39025;2434.41642;434.42055;      ;
TT1       ;STA.01.                           ;   11.93923; 1740.83011; 2056.76707;2434.68000;434.68460;      ;
TT1       ;D.1.                              ;   15.31000; 1736.20361; 2055.04810;    .00000;   .00000;      ;
TT1       ;ISR.406.                          ;   17.77639; 1735.97155; 2056.01637;2435.01149;435.01629;      ;
TT1       ;HLS.2.                            ;   25.64144; 1729.02244; 2052.09659;2434.41566;434.42078;      ;
TT1       ;WPS3.2.                           ;   25.64170; 1729.05401; 2052.01041;2434.24279;434.24791;      ;
TT1       ;WPS1.2.                           ;   25.64185; 1729.09720; 2051.88731;2434.25522;434.26034;      ;
TT1       ;R.1.                              ;   26.00000; 1727.50000; 2052.00000;2433.96401;433.96920;      ;
TT1       ;R.2.                              ;   28.84000; 1723.47136; 2050.46021;2434.12096;434.12633;      ;
TT1       ;D.3.                              ;   33.08000; 1719.48480; 2049.04411;2434.12427;434.12983;      ;
TT1       ;R.3.                              ;   33.08000; 1719.47987; 2049.04231;    .00000;   .00000;      ;
```

Figure 3.2: Example of a theoretical file

There will be a focus on levelling measurements, tilt/roll angle adjustments and offset distance measurements, as these are the most common measurements done by the surveyors in the section at the LHC-tunnel.

### 3.2.1 Levelling measurements

Levelling is the process of determining the height difference between two points. It is used in surveying to measure levels of different points with respect to a fixed point such as elevation of a building, height of one point from the ground, among others.

As can be seen in Figure 3.3, from Leica Geosystems [1], the height difference is calculated from the difference between the two staff readings for the points A and B respectively.

The levelling process is a complex activity, very time consuming, that requires the use of specific and heavy equipment not easy to transport in the LHC-tunnel.

### 3.2.2 Roll angle adjustments

The measurements of the roll angle are used to determine the horizontal angle of an element relative to a horizontal plane. This way it is possible to adjust the angle of components. In Figure 3.4, from NASA [2], it is shown the roll angle in a rocket.

### 3.2.3 Offset distances

As the LHC is slightly curved, each magnet needs to describe a slight curve. For this, elements need to be aligned horizontally and it is necessary to measure the difference between a line of

[1] https://w3.leica-geosystems.com/flipbook/surveying_made_easy/en/index.html/
[2] https://www.grc.nasa.gov/WWW/K-12/rocket/rotations.html

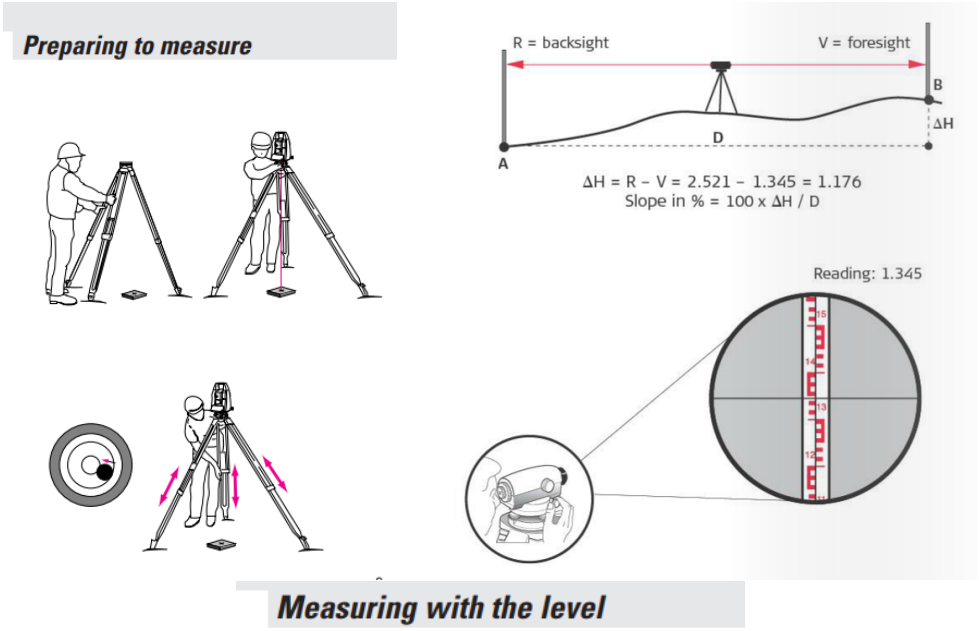


Figure 3.3: Levelling process

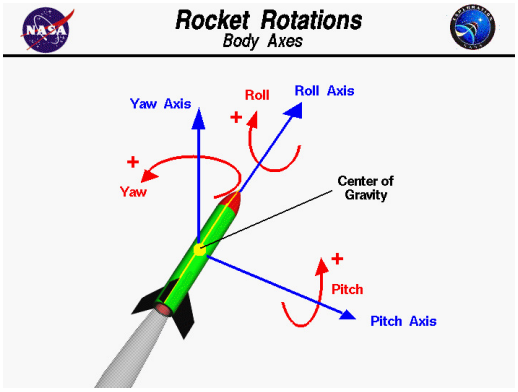


Figure 3.4: Body Axes

reference and the magnets. A wire is stretched between several magnets and simulates that line of reference. To obtain different levels of precision one can apply more or less tension on the wire. The difference between the wire and the magnet is measured with a wire positioning system. The purpose of this measurement is to be able to realign the various elements correctly to allow the particles to circulate with the least resistance possible during their acceleration.

Figure 3.5: Preparation for measurement of offset distance

If the wire is moved during the measurements, surveyors need to redo all previous measurements, as their reference is not the same. This means that the offset measurements require a lot of attention and precision. In Figure 3.5, we can observe a staff to perform offset measurements and, in Figure 3.6 we can see a surveyor performing offset measurements and also the wire.

## 3.3 Mobile Development at CERN

At CERN, Android development is starting. For other projects there is an implementation of a process to follow when developing software that helps and guides the developer following a set of good practices allied with different technologies to have a better product faster. This was not the case for mobile applications, specifically Android development. Mobile development is a growing area, people use mobile phones and tablets in their daily lives and work. The LHC is a 27km long ring and employees who work in the LHC-tunnel need an alternative to laptops, since employees in the tunnels all have mobile phones, it would be an advantage if at least part of their work could be done on a mobile phone or tablet. Employees at CERN have to go to different departments, since workshops and laboratories are scattered around CERN, and many times this even includes crossing the border to another country, it would also be an advantage to have more mobile applications for these employees not to carry a laptop.

Figure 3.6: Offset distance measurements

Due to all of this, CERN would benefit from a standard development process for mobile development that can enable more projects to be idealized and created. It would also prospect the growth of mobile software development at CERN.

### 3.3.1 SMART - before the development process implementation

Survey Measurement Acquisition in Real Time is an application developed in Android at the Engineering department in the APC section. Since there was not any development process specified, the application reached a point where it could no longer be extended.

When the accelerators at CERN stop, a team of surveyors is sent to verify fixed points in the magnets present in the tunnels, these measurements will tell if and what mechanical adjustment need to be done on the field. Since precision is key, measurements to the same point are done at least twice, making this a hard and time-consuming task, and which requires surveyors to annotate the data on the field. SMART was created to help surveyors in this work. It is an Android application that allows the collection of raw data manually written down by surveyors or data acquired automatically from measurement devices via Bluetooth communication.

As described in [31], in 2005 CERN developed the Pocket Field Book (PFB) software supplied on Personal Digital Assistant (PDA) platform for straight data acquisition of leveling, offset distances and roll angle measurements, but hardware obsolescence and evolving needs led to the

need for developing a new software application. Since it is impractical to do measurements and use a laptop simultaneously, there was a need for a new software component, mobile.

SMART has been specifically developed for the onsite acquisition of levelling measurements, roll angles and offset distances on CERN installations, and also proposes several validations on the field that could lead to less user mistakes and even eliminates the risk of typing errors with the Bluetooth acquisition mode.

The goal of the application is to save measurements so surveyors can analyse the collected data and to validate the measurements onsite.

### 3.3.2 SMART process

SMART has five main phases as followed:

1. Module selection
2. Team and Operation identification
3. Data Acquisition
4. Data Validation
5. Save data

In the first phase the type of measurement is selected, either levelling measurements, roll angle adjustments or offset distances, as in Figure 3.7.


Figure 3.7: Module selection - Application Home Page

Next the user inputs job metadata (operators initials, instruments serial numbers, operation identification) as shown in 3.8. Then, the user defines the folder path containing the file of theoretical co-ordinates (theoretical file).

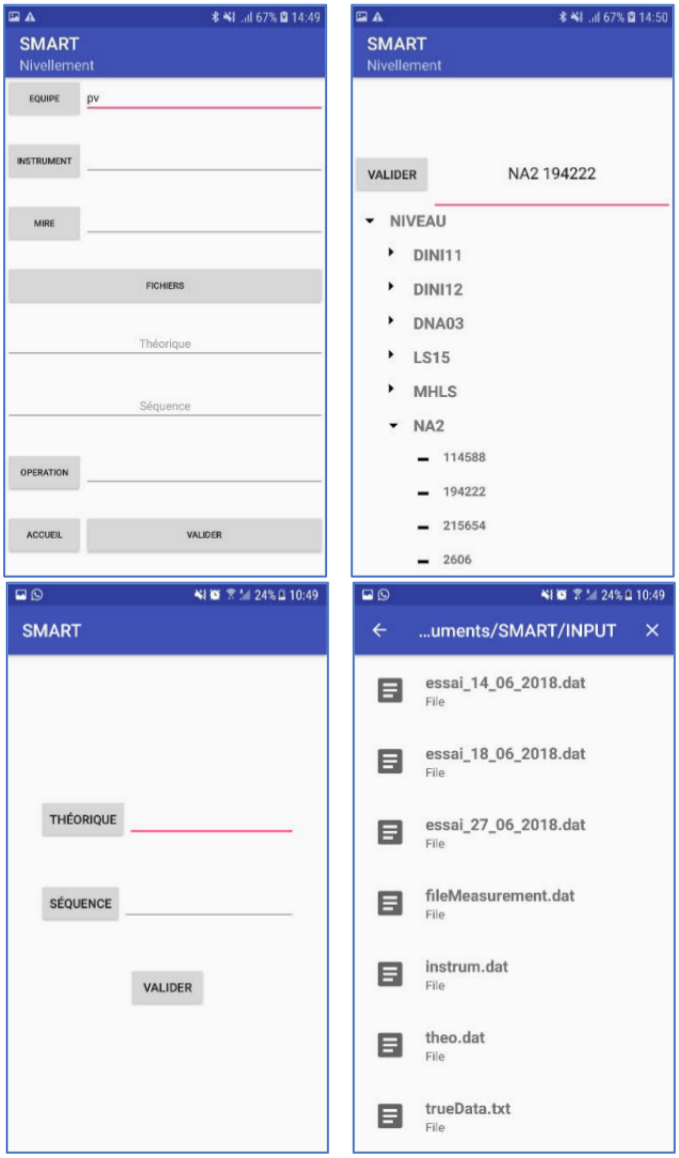


Figure 3.8: Input data - Selection of instruments and theoretical files

After that, a selection of the points to be measured must be done. This can be done accessing the list of points in the theoretical file as in Figure 3.9.

Then the data needs to be written, either by the surveyors or, in the levelling module, by the instrument via Bluetooth. In Figure 3.10 the data acquisition view is presented.

As the data is recorded, SMART does a set of checks to ensure if the point or station needs to be measured again. The data is recorded in a .dat file that is kept in the memory of the phone and is displayed to the user in the Journal tab, which can be seen in Figure 3.11.

### 3.3.3 Levelling measurements with SMART

In Figure 3.12, there is a set of points to be measured, point P1, P2, P3 and P4, these set of points can be considered a station.

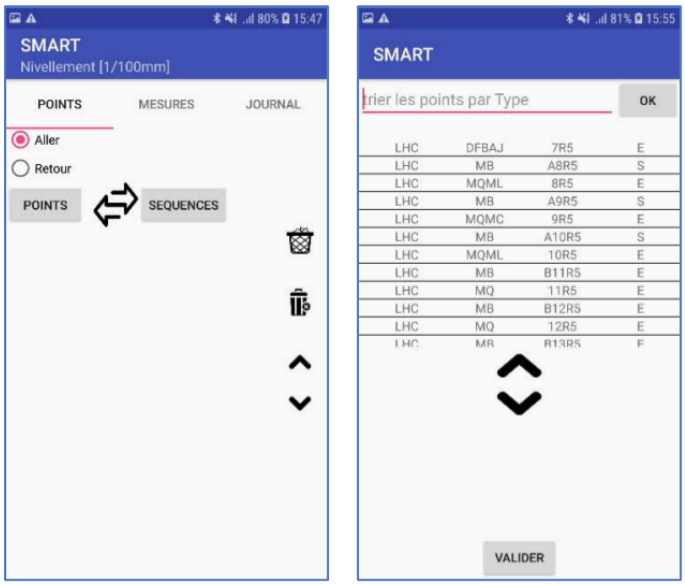


Figure 3.9: Selection of points to be measured

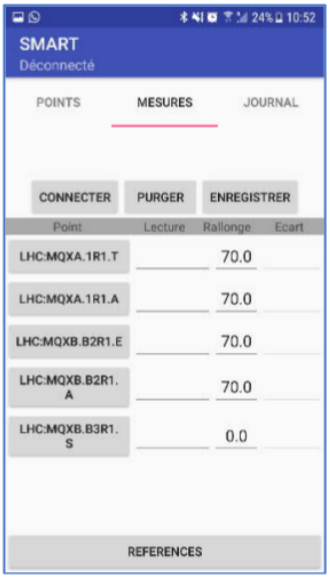


Figure 3.10: Data acquisition

On the levelling measurements, SMART computes, for each station, the deviation between forward and backward measurements, which, in Figure 3.12, is represented by Mx (forward measurements) and Mx' (backwards measurements) lines in the points. The user will be notified if the measurements are not correct and if a recovery measurement is needed, in this case, the user needs to redo the measurements.

SMART is able to connect to levelling instruments via Bluetooth and collect data automatically but not autonomously.

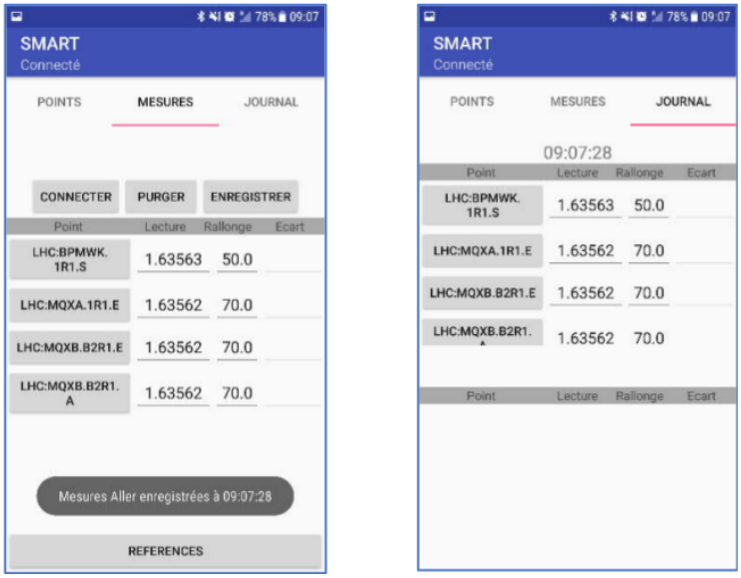


Figure 3.11: Recording measurement data - Journal tab

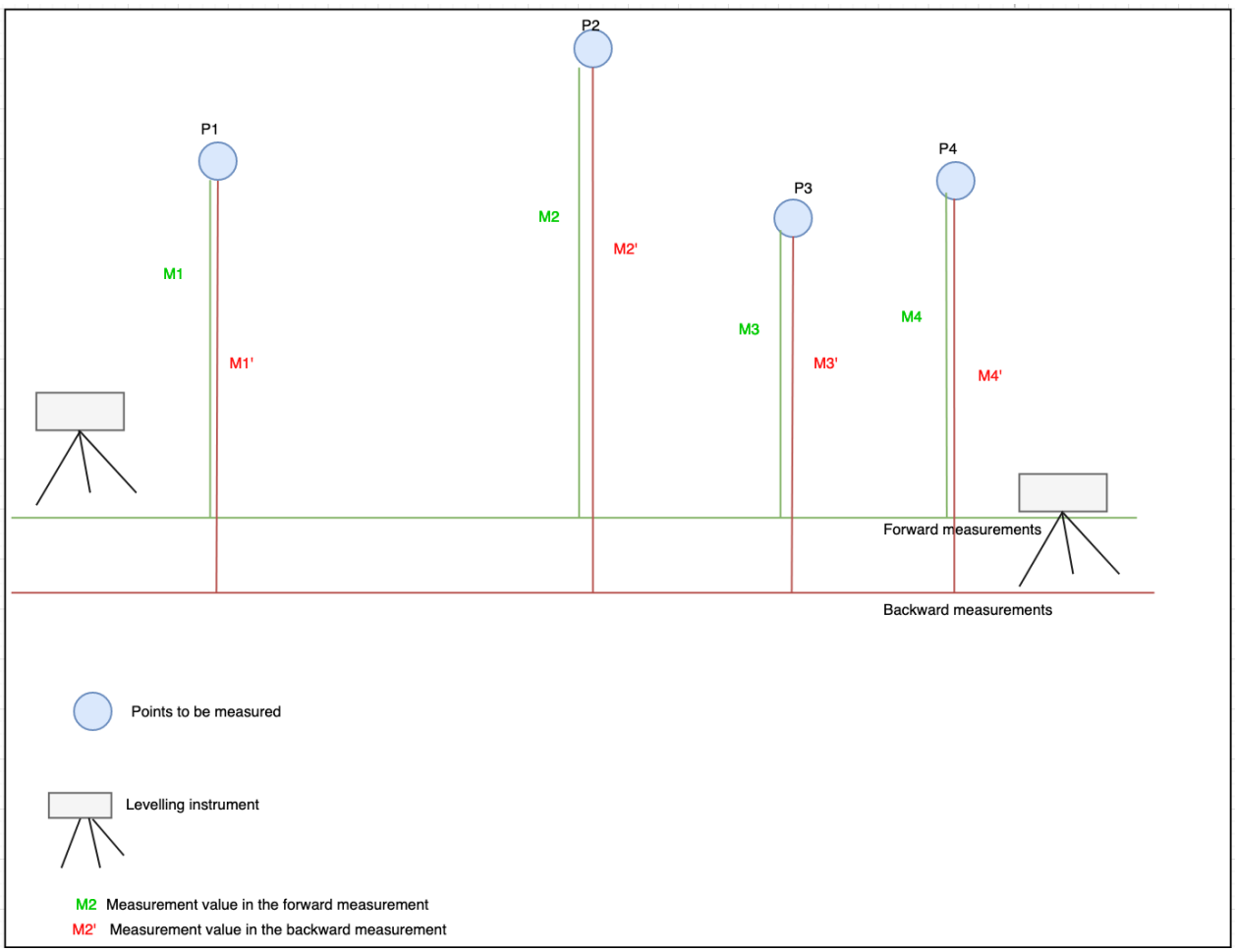


Figure 3.12: Example of the levelling process

### 3.3.4 Roll angle adjustments with SMART

The live roll angle adjustment can be corrected as they are measured in the field. SMART computes the deviation of the roll angle with respect to the theoretical value that is found in the theoretical file selected by the user.

### 3.3.5 Offset distances with SMART

Fiducials represent marking points clearly visible on objects that have to be measured by surveyors. In general, their spatial coordinates (x,y,z) have been previously calibrated (measured by different means and surveying instruments) and their values are defined with respect to a reference coordinate-system attached to the object. In the case of the LHC, the fiducials are located on the outside surface of the elements and their positions are well-known with respect to internal elements such as the magnetic axis of a magnet (particle beam axis). By measuring onsite a few visible fiducial points, surveyors are able to precisely align the accelerator element with respect to the theoretical position of the particle beam.

With SMART, the surveyor can focus on positioning the offset staff that is used to make these measurements correctly and writing down the measurements quite fast, instead of having to leave their position to write on a laptop with the risk of making a mistake and not being able to get the precise measurement again.

### 3.3.6 Validations of the measurements

In the levelling module, the deviation between forward and backward measurements is calculated. For a measurement to be considered a valid measurement, this difference needs to be very small, e.g. 0.00001, one hundred of a millimeter between the two measurements. This difference is called the root mean square, or *Erreur Moyenne Quadratique* (*EMQ*) 3.1.

$$EMQ = \sqrt{\frac{\sum_{i=1}^{N}(e_i)^2}{N-1}} \tag{3.1}$$

Which is the square root of the quadratic sum of offsets (deviations) divided by the number of measurements (minus 1 because we started at element one instead of zero). This number will give an average error of each measurement.

For the offset, the coordinates of the points present in the theoretical file are used to compute theoretical offset distances between fiducials and the stretched wire, when comparing this result with the measurements done on the field, the surveyor can detect errors, either typing or reading error

or movements in the wire.

In the roll angle measurements, SMART computes the deviation of roll angle comparing with the theoretical value, present in the theoretical file. Depending on the result and the associated tolerance, operators might have to adjust the element.

### 3.3.7 Problems SMART is not addressing

SMART was not tested in the LHC-tunnel, this was due to a set of requirements still not being fulfilled.

The application did not validate the measurements correctly, for example, validations, like the *EMQ*, were not correct and in some cases SMART would alert the user that the measurements were not correct, when they were.

The Bluetooth communication was only possible for one type of measurement, this did not give a lot of autonomy to the users, as they still had to write down all measurements by hand in the smart phone.

Another issue was that there were only three types of measurements available.

For the surveyors, the biggest goal was to use and test the app in the field, on the LHC tunnel to give correct feedback on the way to use the application and this was an issue yet to be resolved.

Bad feedback towards the user interface not being intuitive and compromising usability were also received.

## 3.4 Work from this point

For this Dissertation, the main work consisted in, from this state of the application:

- Propose a set of good practices to be applied to the existing code
- Test the application on the field
- Extend the application by adding new features such as adding communication with more measurement devices
- Add measurement modules such as the acquisition of gyroscopic measurements
- Add more validations to measurements
- Create a development environment for good practices for mobile development at CERN
- Validate the set of good practices

Firstly, in order to propose a set of good practices to follow in our development process, an analysis of good practices in Software Engineering is required.

If, by applying these good practices, the appliction is stable and surveyors feel more confidence to test the application in the field, the next step would be to do tests in the LHC-tunnel.

If the tests are successful, new features that would help surveyors could be added, extending the application.

A study on more measurement types, such as the gyroscopic measurements should be done in order to acquire more information and requirements on how to add this module to SMART. By adding more validations to the measurements, surveyors would feel more comfortable performing their work and using the application.

The creation of a development environment for mobile development will introduce a new way of developing better quality software.

SMART will be used to validate our set of good practices and our development process.

# Chapter 4

# Approach

In this chapter an overview of the developed work for this dissertation is presented. A description of the development process encountered can be found in Section 4.1. In Section 4.2, an analysis of software development methodologies and of the software development processes at the Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software section, followed by an analysis having an agile approach for mobile application development and of best practices for Android development. There is a revision of the main changes to the existing code in Section 4.3.

## 4.1 SMART's development process

The product owner, a surveyor that was responsible for the SMART project, and one of the developers of SMART who was still at the European Organization for Nuclear Research (CERN) clarified SMART's development process as the following:

1. The product owner would write users stories in JIRA, on the SMART Testing board;
2. The developer would choose one or more user stories;
3. The developer would do the task and put it in the Testing board in the Kanban board;
4. The product owner would verify the changes and approve or not the task.

Due to the problems referred in 3.3.7, we decided to make changes toward this development process. In the following sections it is explained the decisions of the development process approach.

## 4.2 Analysis

Given the problems presented in Chapter 3, an analysis of the different software development methodologies was conducted, then a study of the development process at the Acquisition, Processing and Control Software (APC) section and an analysis of an agile approach for mobile application development. These analyses help to verify which methodologies go according to the

requirements of the product owner. In Section 4.3 we will list a set of practices that are considered good practices while developing to achieve solid and well-designed software.

### 4.2.1 Analysis of software development methodologies

According to Špundak,project management methodology can be defined as a set of methods, techniques, procedures, rules, templates, and best practices to be used on a project [45].

After analyzing literature about methodologies we focused on the main differences when using Traditional methodologies, depicted in Figure 4.1, or Agile Methodologies, depicted in Figure 4.2, in software development. Both Figures can be found in Medium [1]

As defined by Kumar and Bhatia, traditional models like waterfall are easy to understand and reinforce the notion of "define before design" and "design before code". These models expect complete requirements early in the process that do not change, which are unrealistic, while more modern models, like agile methodologies, are based on iterative and incremental development. The main characteristics that are fundamental to agile methodologies are adaptive planning, iterative and evolutionary development, rapid and flexible response to change and promote communication. This way, agile methodologies can achieve higher quality software in a shorter period of time, have self organizing teams and customer collaboration with less documentation and reduced time to market [25].

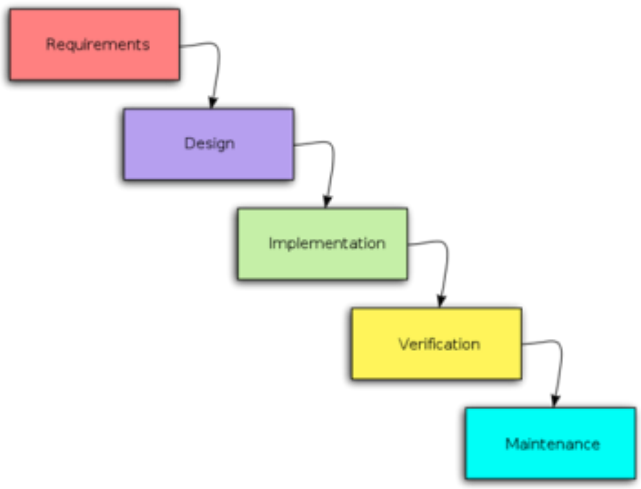


Figure 4.1: Waterfall development methodology

As concluded in [44], [8] and [37], when having a small-medium project development, agile methodologies excel traditional methodologies. Sharma, Sarkar and Gupta affirm that "*In the comparative study of agile software development with other software development models we conclude that agile project is much better than other software development process in terms of productivity, performance, faster time cycles, risk analysis.*" [37]. According to [22], Agile methods have

[1] https://medium.com/@AbdulD/the-6-steps-to-a-successful-agile-software-project-9084c23e2efb

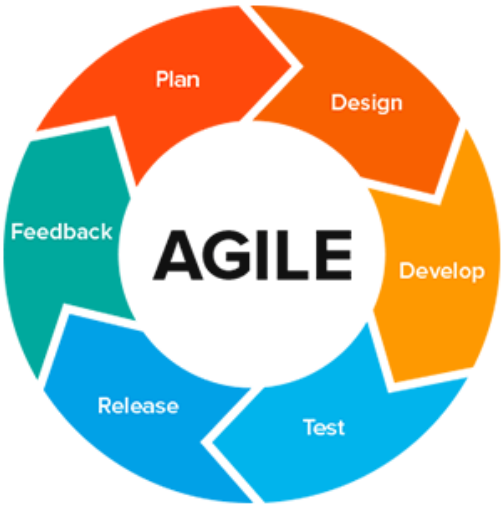


Figure 4.2: Agile development methodology

practices that have quality assurance abilities and that these practices occur more than in waterfall development and are available in very early process stages due to the agile process characteristics.

As stated previously, at CERN, the organization of projects and developments greatly depends on the field (engineering, computer science, physics of accelerators, theoretical studies and so on). At the Engineering - Survey, Mechatronic and Measurements (EN-SMM) group, the software and hardware developments are mainly based on Agile methodologies, especially on a Kanban-based management of the activities.

Given the analyses done and having in consideration the methodology used at CERN, we opted to use an Agile methodology.

#### 4.2.1.1 Project requirements and Agile approach

The project owner and the developers agreed to have regular beta-releases to be tested in the field throughout the development process so users can give feedback and the project goes in the direction of their requirements. So, an approach that deals with early releases, frequent customer feedback and volatile requirements was needed.

As supported by [22], two fundamental characteristics of agile approaches are the handling of unstable requirements throughout the development life cycle and the delivery of products in shorter time-frames when compared to traditional development methods. Agile methods use techniques such as simple planning, short iteration, earlier release, and frequent customer feedback to deal with unstable and volatile requirements.

### 4.2.2 Software development processes at the Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software section

CERN is composed of departments each with multiple groups and sub-sections. The organization of their projects and developments greatly depends on the field and at EN-SMM group, the software and hardware developments are mainly based on Agile methodologies, and especially on Kanban-based management of the activities. It can be thought of as a large, prioritized to-do list. Requirements in Kanban are tracked by their current stage in the process (to-do, in development, in testing, done, among others). When a developer is ready for the next task, he/she pulls it from the to-do list. Moreover, a few end-users of the team (surveyors in our case) have full access to these tasks lists in order to facilitate the organization of their tests.

In this section, software is mostly done for surveyors. Most projects are developed in C++ and many good practices are followed from the beginning of the development process, such as having unit tests from the start, coding conventions are already defined, continuous integration, among others.

### 4.2.3 Agile approach for mobile application development

Mobile development has been increasing throughout the years, this is also due to the increasing programming environments available for developers. Many development process models are used for traditional methodologies, which do not deal with highly volatile requirements that require adaptive software development methods.

As concluded in [24], [15] and [6], a methodology that can adjust to changes of requirements over time, producing valuable software in short time periods and with low budget is necessary and that Agile methodologies should be adopted to deal with these constraints and also that deals with uncertainty and a number of unknown-variables.

In [36] there is an investigation of challenges when adopting agile practices. The authors conclude that there are five main challenges in mobile development and eight benefits in adopting an agile methodology. It is found that "[...] the main challenge to develop mobile applications is to define UI and UX followed by achieve different users' expectations. Regarding to the benefits, we found improvements on management and control as well as development speed."

The choice of a methodology to follow is important in mobile software development as applications are always changing and evolving based on the users requirements.

With this analysis, it is concluded that Agile practices are the most appropriate for the development of mobile applications.

### 4.2.4 Best practices recommended for Mobile Software Engineering from literature

We can say that by choosing an agile methodology we are already implementing a good practice. As referred in [24], the environment of developing mobile applications is an environment with rapidly changing user demands that requires an agile approach. By following an agile approach

we follow a set of characteristics that align with requirements for mobile application development, such as:

- Test-driven development
- Continuous costumer involvement
- Prioritization of requirements
- Effective Communication
- Enhanced quality assurance
- Skilled developers
- Process-wide reviews and learning sessions
- Adaptive process

More information about these characteristics can be found in [24] and [42].

We followed the Developer's Guide for Android [2] that includes a Best Practices section that addresses application compatibility, user interface guidelines, and designing for performance and responsiveness, among others.

All these characteristics will be taken in consideration while developing our application and while applying our development process.

### 4.2.5 From Stupid to Solid code

In order to have a good quality code we followed a set of principles. We avoided the STUPID principles and adopted SOLID principals.

These principles tell us to avoid a set of points, which follow:

- Singleton
- Tight Coupling
- Untestability
- Premature Optimization
- Indescriptive Naming
- Duplication

And to adopt a collection of design principles for good code that was invented by Robert C. Martin. The principles are presented as follow:

[2] https://developer.android.com/distribute/best-practices

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion Principle

More from Stupid to Solid Code by William Durand can be found here [3]

## 4.3 Implementation

Subsequently to the analyses of the software development process at the APC section, and after investigating the best methodology and best practices to apply when starting the work, the next step was to review the application that already existed. As said before, SMART was already being developed by two students but no tests on the field were performed.

### 4.3.1 Code Inspection

The first step was a static code analysis. An analysis to the source code against set rules. These analysis are performed before code execution. So, a code inspection was done using Android Studio on the 24 of September of 2018, it is depicted in Figure 4.3. Code inspections detect anomalous code even before the compilation. It can find problems like unused code, probable bugs and can suggest improvements to the overall code structure. In Android Studio inspections can be done to the whole project or to specified scopes. Inspections have severity levels that specify the extent to which a problem can affect the code. These severity levels can also be configured. Android Studio also provides a tool called Lint that helps find structural problems in the code. "The lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization. " [4]

Manual code inspections are also possible but human errors could make it not as effective or efficient. It was concluded that many changes needed to be done to the existing code in order to make it more readable, maintainable, to be able to add new features and more.

It was also decided that regular code inspections should be a part of our development process and that they would occur at every merge request to the master branch.

### 4.3.2 Changes to the Architecture and Design

The next stage was to see the followed architecture and design.

SMART followed a Model-view-controller (MVC) pattern, where the model that contains the data and core functionalities is separated from the view where the information to the user is

[3] https://williamdurand.fr/2013/07/30/from-stupid-to-solid-code/
[4] https://developer.android.com/studio/write/lint

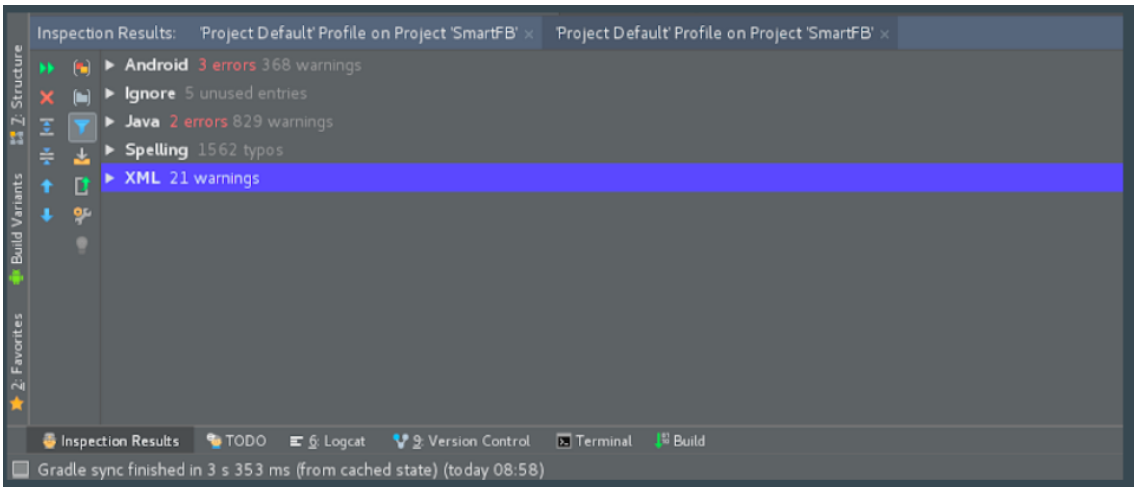


Figure 4.3: First code inspection performed at 24 of September 2018

displayed and from the controller, that handles the input from the user. With this, the controller and the view depend on the model, the controller to update the data and the view to get it.

This was the proposed architecture used which helped achieve a clear data flow, the code to be more readable and maintainable and to have more testable code. By applying this pattern, we can export the application template as a library and other developers can use it in their application development, making it easier to reuse code.

In the documentation for Android Developers created by Google [5], there is documentation for Design & quality [6], it focuses on material design guidelines for visual and navigation patterns, and also follow quality guidelines for compatibility, performance, security, and more.

Following the concept of material design, layouts were modified to be more intuitive, predictable, consistent and responsive. This means, having consistent user interface (UI) regions, consistency in layouts, like consistent padding and being adaptive and react to inputs.

An example of layout modification in SMART can be seen in Figure 4.4. The size of the items was increased to make it more user friendly, increased the font size so everything was more readable. Some items that were not necessary were removed and the selection window of the instruments was modified, instead of going to a new window and select an instrument from a tree, a pop up simply appears with a list of all instruments.

For the core app quality that deals with visual design and user interaction, functionality, compatibility, performance and stability and security, a set of defined aspects were followed.

These criteria can be seen in the Core app quality documentation [7]. Not all criteria could be applied to our application but the ones that were possible and made sense in the context of our application were applied.

[5] https://developer.android.com/
[6] https://developer.android.com/design
[7] https://developer.android.com/docs/quality-guidelines/core-app-quality

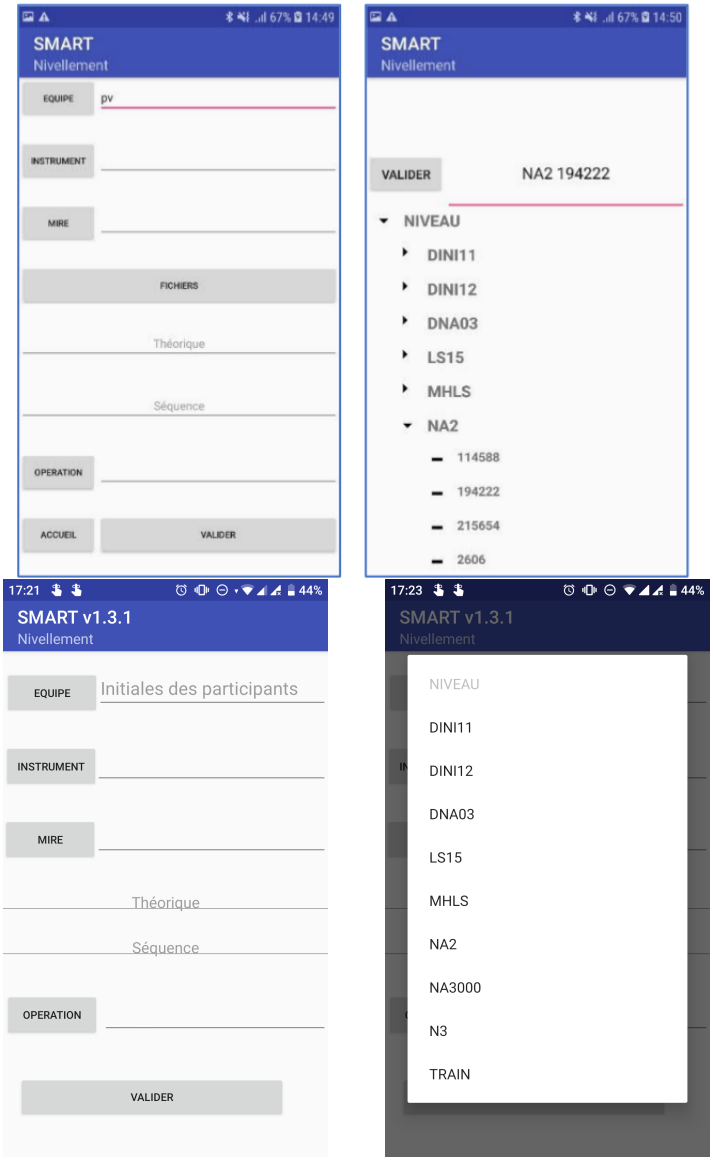


Figure 4.4: Example of layout modification in SMART

### 4.3.3 Defined requirements

The requirements for the project were already defined. Since they were created by surveyors that had the goal of having an application to ease their work, the requirements were defined with agreement between them, having in consideration their difficulties and how the application should help surveyors overcome them.

On JIRA [8], issues with the specific type for requirements were created and sub-tasks for them too. In Figure 4.5, there is an example of SMART's Kanban board.

[8] its.cern.ch/jira

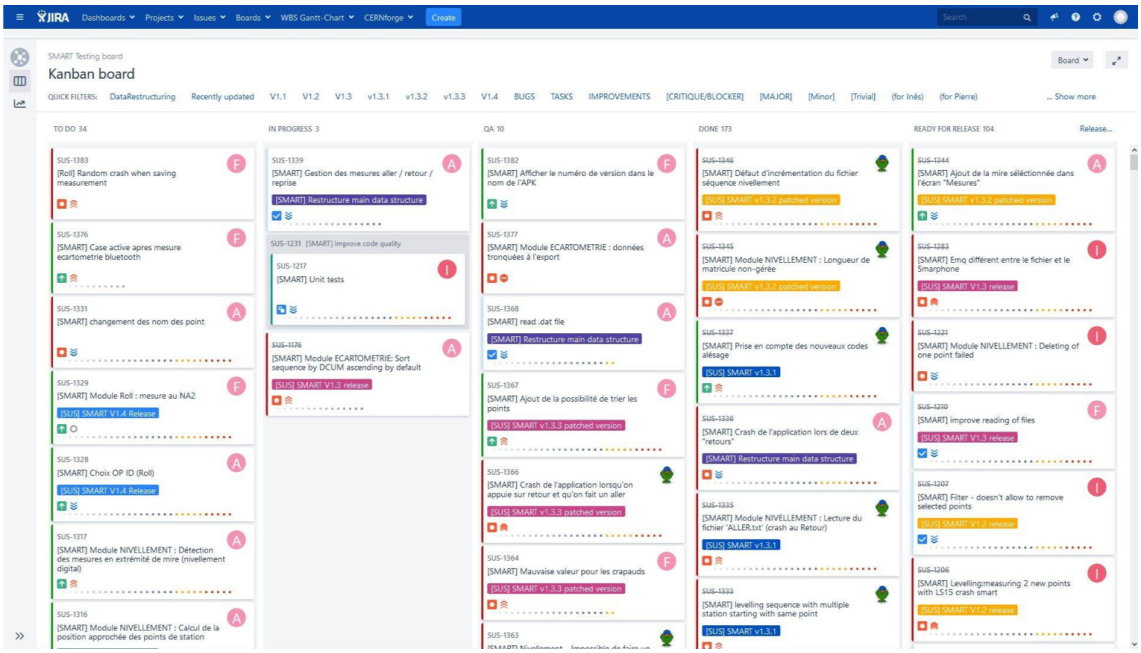


Figure 4.5: SMART's Kanban board on JIRA

### 4.3.4 Planning

After the analyses of the state of the application, the first goal is to improve the existing code. Tests in the office were planned in order to verify that the application works as intended.

After having a code that is maintainable and stable, the next goal is to test the application on the field, adapt the application and our development process accordingly with the users needs and finally extend the application.

The product owner planned tests on the field with different teams of surveyors. Each test was planned accordingly with the features to test in the application.

### 4.3.5 Changes to the existing code

As the code needed changes, some coding standards were defined, design patterns were applied, as well as refactoring techniques.

#### 4.3.5.1 Defining Coding Standards

After the code inspection that revealed that the code needed to be changed, and since the code is to be developed by more than one developer, it was defined a group of coding standards to make the code more readable and maintainable.

We followed the official Android code style guidelines [9].

Besides adopting these guidelines, it was also decided to follow the conventions listed in Android Coding Standards [10].

[9] https://source.android.com/setup/contribute/code-style
[10] https://github.com/ustwo/android-coding-standards

#### 4.3.5.2 Design Patterns

In order to improve the readability of the code and to have a way to find problems sooner in the development phase, design patterns were applied to the existing code. An example using the Composite design pattern will be presented as well as the advantages of it.

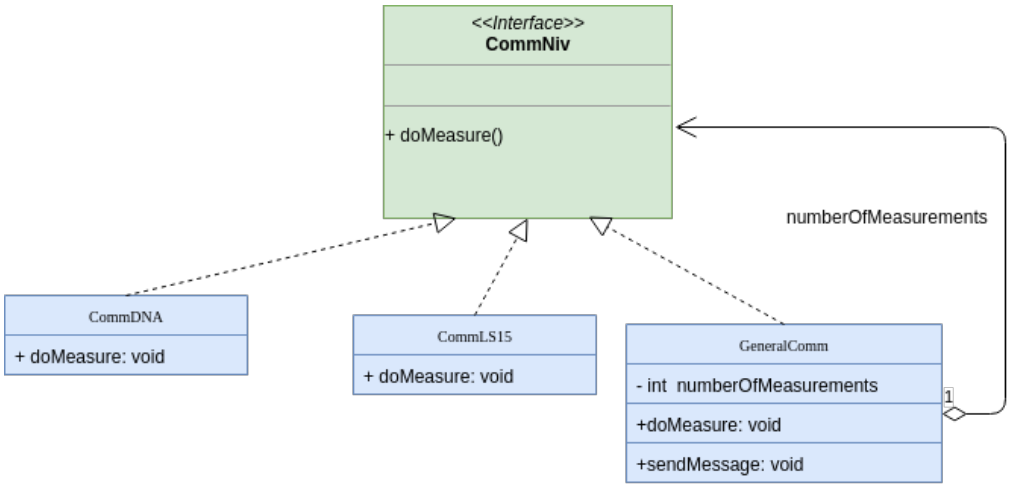


Figure 4.6: Composite Design Pattern

In Figure 4.6 it is presented a class diagram showing the implementation of the composite design pattern. The CommNiv interface was created to manage the communication for the leveling. To perform the levelling measurements, instruments like the DNA and LS15 were used and are represented in the diagram by the classes CommDNA and CommLS15 respectively. These classes implement the doMeasures() method, present in the interface, where a command is sent to the instrument to perform the measurements.

This allows treating individual objects and compositions of objects the same way.

#### 4.3.5.3 Refactoring

Refactoring techniques are used to improve the software quality and make it easier to understand. This helps to reorganize the code without changing its external behavior. Essentially, refactoring improves the design of the code after it was written and minimizes the chances of introducing bugs.

#### 4.3.5.4 Code smells

In this section it will be presented a list of code smells as well as a brief description of each one. The identification of code smells is important to help identify them in the code so refactoring techniques to eliminate them can be applied. This process helps improve the design of the code, making it more readable and maintainable.

- Bloaters - part of the code that has increased to such proportions that it can no longer be maintained and are hard to read.

- Object-Orientation Abusers - incorrect or incomplete application of object-oriented programming principles.
- Change Preventers - if a change is necessary in one place in the code, it will require changes elsewhere in the code too.
- Dispensables -Something that if not present would make the code cleaner

In order to clean the code and, if possible, eliminate these code smells, some refactoring techniques were used and will be presented in the following sub-section.

#### 4.3.5.5 Refactoring techniques

There are many refactoring techniques we can use, in Sourcemaking [11] more details about these techniques can be found.

The following are some of the used techniques:

- Extract Method
- Inline Method
- Encapsulate Field
- Consolidate Conditional Expression
- Form Template Method
- Pull Up Method

In Figure 4.7, we can see that before applying the form template method, the method fillTableJournal() was declared in 3 classes, after applying the technique, fillTableJournal() is declared in abstract class ActivityMesureMethods. This method was a method for presenting the taken measurements. A template method was created to eliminate this duplication by merging the shared algorithm steps in a superclass (the ActivityMesureMethods) and leaving just the differences in the subclasses (ActivityMesuresNivellement and ActivityMesures).

Forming a template method is an example of the Open/Closed Principle in action. When a new algorithm version appears, you only need to create a new subclass.

The Extract Method was also used by splitting the algorithm in the subclasses into their constituent parts described in separate methods. After this, we moved the identical methods for all subclasses to ActivityMesuresMethods by applying the Pull Up Method.

The parts of the methods that were not similar were moved as abstracts to ActivityMesuresMethods and the implementations in the subclasses.

By applying these refactoring techniques, the code became more readable, with less code duplication and part of the code was isolated, reducing the probability of errors occurring.

[11] https://sourcemaking.com/refactoring

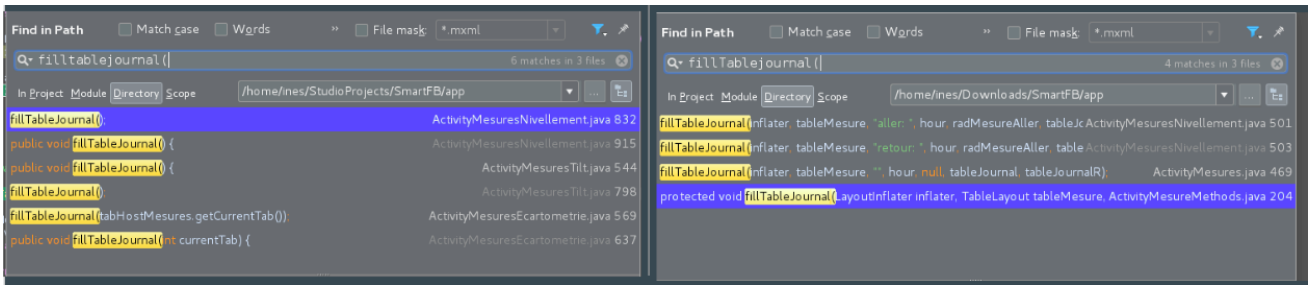


Figure 4.7: Example Form Template Method - Before and After

#### 4.3.5.6 Code Review

Code reviews ensure the code is effective, understandable, maintainable, and secure.

It was decided that for a merge request to be accepted it should be reviewed by two people, a developer of the project and a developer who was not part of the project. Only after both reviewers accepted the changes the merge could happen. Code reviews are done in GitLab's interface.

When asking for a review of the code, the developer is not just asking for second opinion on the chosen solution and implementation, but also an extra pair of eyes looking for bugs, logic problems, or uncovered edge cases.

In GitLabs documentation concerning code review, they list a set of good practices to be followed [12]. These practices will be followed by developers and code reviewers. The reviewers should verify if all the coding standards are followed and if the written code can be improved and look for bugs, logic problems and uncovered edge cases. If the reviewers have changes they would like to see done, GitLab allows them to choose the lines of code and comment and notifies the developer who did the merge request. Once the changes are done by the developer, GitLab marks those lines of code so the reviewers know changes have been done.

### 4.3.6 Tests

The application had no type of tests so before developing anymore code it was necessary to implement at least unit tests. Testing is a must in our development process, but due to time constraints it was decided that the developers that would come to keep developing the application should do other types of testing, such as integration tests, system tests and acceptance tests.

#### 4.3.6.1 Unit Tests

Unit testing is a way to test a unit, a unit can be almost anything, even a line of code or a whole class. Since these tests are usually smaller it gives a better view at how the code is performing. Unit testing encourages better coding practices making documentation easier and the code more maintainable and reusable, as the code needs to be more modular for unit tests. Its purpose is to validate that each unit of the software performs as designed.

[12] https://docs.gitlab.com/ee/development/code_review.html

JUnit framework, PowerMock, a Java framework, were used to help develop and execute unit tests in Android Studio.

In Figure ??, an example of Unit Testing in SMART to test the DNA class is presented.

```
/**
 * Tests for the DNA class
 */
public class TestDNA {
    //Runs after all the tests in this class, sets the variables to zero so when we run all tests together, t
    @After
    public void afterTest() {
        GeneralComm.staffTemperature = "";
        GeneralComm.distanceToStaff = "";
        GeneralComm.spread = 0;
    }

    //Tests if with an input for the temperature (95...6+) it returns the correct value for the temperature (
    @Test
    public void test_temperatureReceiver() {
        CommDNA commDNA = new CommDNA();
        String fakeText = "95...6+2599999999";
        Assert.assertEquals(commDNA.receiveText(fakeText), actual: "25999999");
    }

    //Tests it returns the correct value for the measurement (for the manual)
    @Test
    public void test_ManualTextReceiver() {
        CommDNA commDNA = new CommDNA();
        String fakeText = "110007+000000A1 32...8+00243730 331.08+001408877 110007+000000A1 110007+000000A1";
        Assert.assertEquals(commDNA.receiveText(fakeText), actual: "140887");
    }
}
```

Figure 4.8: Unit Testing in SMART to test the DNA class

```
/**
 * Tests for the methods of writing in the output file
 */
// this test case need to mock static methods so it uses PowerMock
@RunWith(PowerMockRunner.class)
// this static methods to be mocked are on Environment so that must be 'prepared'
@PrepareForTest({Environment.class})
public class TestOutputFile {
    // we store our file in the phone, so we need to 'pretend' that in our tests the output file also has a directory to be saved
    @Rule
    public TemporaryFolder storageDirectory = new TemporaryFolder();
    private File existentDirectory;

    //this will run before the tests, it mocks a file and the directory to save the file
    @Before
    public void setup() {
        File nonExistentDirectory = mock(File.class);
        when(nonExistentDirectory.exists()).thenReturn(false);
        existentDirectory = storageDirectory.getRoot();
        PowerMockito.mockStatic(Environment.class);
    }

    //tests if the name attributed to the file is the correct one
    @Test
    public void shouldHaveCorrectName() {
        when(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS)).thenReturn(existentDirectory);
        File dummy;
        boolean isNameCorrect = false;
        try {
            dummy = WriteActivityMesures.createNewFile( isAllerFile: false);
            String dummyName = dummy.getName();
            isNameCorrect = dummyName.contains("LEV");
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Checks if the name of the file contains the word LEV : " + isNameCorrect + " (Levelling file)");
        Assert.assertTrue(isNameCorrect);
    }
```

Figure 4.9: Unit Testing in SMART to test the creation of output files

In Figure 4.9, it is represented a unit test to validate certain parameters of the output file created by SMART with the measurement values. The parameter tested in this case is the name of the output file as it needs to be a specific name, as indicated in the test, it needs to contain the word "LEV".

#### 4.3.6.2 Continuous Integration and Continuous Deployment

Continuous Integration (CI) was implemented with GitLab Continuous Integration, Docker and Android Studio to ensure that the app compiles, passes unit tests and to automatically create a signed Android Package (APK).

GitLab CI/CD pipelines are configured using a YAML file called .gitlab-ci.yml within each project. This file defines the structure and order of the pipelines and determines:

- What to execute using GitLab Runner
- What decisions to make when specific conditions are encountered. For example, when a process succeeds or fails

For more information on this topic visit the documentation on GitLab [13].

GitLab CI uses GitLab runner to run jobs. The docker image is used as the environment and the tasks defined in .gitlab-ci.yml are executed, they can be seen in 4.10.

A docker image was used as a job runner/worker for the CI. Its goal was to compile, run the tests and automatically build a signed APK for the SMART application. The image contains all the prerequisites to build and run the Android SDK, removing the need to reinstall them at each run. The prerequisites are the following, SDK tools, Platform tools and Build tools. For more information about the usage of this image refer to GitLab [14].

As soon as there is a commit to the repository, the image will automatically be built thanks to Gitlab-CI. The automatic build may fail for various reasons such as CERN infrastructure down-time, network issues or unavailable resource among others. This image can be used as a runner for other project's CI.

When a merge request was accepted in the master branch of SMART's repository in GitLab, an APK was automatically deployed. The name of the APK is SMARTFB-version.apk, version corresponding to the version number of SMART.

In order to be able to build the release APK it is necessary to create a keystore and use its password during the APK generation. After the keystore being generated, the passwords are stored in the GitLab CI. It was also necessary to change the gradle.properties file to add the following variables: release_keystore_path, release_keystore_password, release_key_alias and release_key_alias_password. And modify the build.gradle file.

This APK can be downloaded directly from CERN's gitlab [15].

### 4.3.7 Life cycle of the application

Methods related with the life cycle of the application like the onStop(), onDestroy() and onResume() were modified because switching between the measurement tab to the points tab or the

[13] https://docs.gitlab.com/ee/ci/yaml/
[14] https://gitlab.cern.ch/apc/common/docker-image-susoft-android
[15] https://gitlab.cern.ch/apc/susofts/acquisition/SmartFB/-/jobs/artifacts/master/raw/SMARTFB-release.apk?job=release_job

.gitlab-ci.yml 1.56 KB

```yaml
# Check https://gitlab.cern.ch/gitlabci-examples/build_docker_image
# for a full reference of the variables you can use to customize your
# Docker builds

# Here, image will be pushed to gitlab-registry.cern.ch/dataprocessingandanalysis/sus_ci_androidworker

stages:
    - build
    - test

build_sus_ci_androidworker:
    stage: build
    image:
        name: gitlab-registry.cern.ch/ci-tools/docker-image-builder
        entrypoint: [""]
    script:
        # Prepare Kaniko configuration file
        - echo "{\"auths\":{\"$CI_REGISTRY\":{\"username\":\"$CI_REGISTRY_USER\",\"password\":\"$CI_REGISTRY_PASSWORD\"}}}" > /kaniko/.docker/config.json
        # Build and push the image from the Dockerfile at the root of the project.
        # To push to a specific docker tag, amend the --destination parameter, e.g. --destination $CI_REGISTRY_IMAGE:$CI_BUILD_REF_NAME
        # See https://docs.gitlab.com/ee/ci/variables/predefined_variables.html#variables-reference for available variables
        - /kaniko/executor --context $CI_PROJECT_DIR --dockerfile $CI_PROJECT_DIR/Dockerfile --destination $CI_REGISTRY_IMAGE

# This images are ready to be used on gitlab-CI builds. As an example, run a simple job in the image
# we have just built in the previous step
run_example:
    stage: test # So it is run after building the images
    image: gitlab-registry.cern.ch/apc/common/docker-image-susoft-android:latest # Use the image we just built and push to the registry
    tags:
        - docker
    script:
      - cd MyApplication/
      - ./gradlew testRelease
      - ./gradlew assembleRelease
```

Dockerfile 1.29 KB

```dockerfile
# Base CERN worker image
FROM gitlab-registry.cern.ch/ci-tools/ci-worker:cc7


# Install needed packages
RUN yum update -y \
    && yum install -y \
        java-1.8.0-openjdk-devel \
        htop \
        mlocate \
        wget \
    && yum clean all \
    && updatedb

# Sets the locales in UTF8
ENV LC_ALL=en_US.UTF-8

# Environment variables
ENV VERSION_SDK_TOOLS="4333796" \
    SDKMANAGER="28" \
    BUILD_TOOL="28.0.3" \
    ANDROID_HOME="/usr/local/"

# Install Android SDK & SDKs
RUN echo "Installing sdk tools" && \
    wget --quiet --output-document=sdk-tools.zip "https://dl.google.com/android/repository/sdk-tools-linux-${VERSION_SDK_TOOLS}.zip" && \
    unzip -q sdk-tools.zip -d "$ANDROID_HOME" && rm --force sdk-tools.zip  && \
    mkdir  --parents "$ANDROID_HOME/.android/" && \
    yes | "$ANDROID_HOME"/tools/bin/sdkmanager --licenses > /dev/null && \
    echo "Installing platforms" && \
    yes | "$ANDROID_HOME"/tools/bin/sdkmanager > /dev/null "platforms;android-$SDKMANAGER" && \
    echo "Installing platform tools " && \
    yes | "$ANDROID_HOME"/tools/bin/sdkmanager > /dev/null "platform-tools" && \
    echo "Installing build tools " && \
    yes | "$ANDROID_HOME"/tools/bin/sdkmanager > /dev/null "build-tools;$BUILD_TOOL"

WORKDIR /app
# For testing uncomment
#COPY ./MyApplication/ .
```

Figure 4.10: Dockerfile and gitlab's yaml file for CI with GitLab and Docker

journal tab as well as the action of locking the phone, caused a failure of Bluetooth communication with the instrument. This was due to the piece of code shown in Figure 4.12 and Figure 4.13.

Whenever onStop() was called, the Bluetooth connection would stop, but not when onDestroy() was called, so the code from onStop() was removed and added to onDestroy(). This way, if the application was no longer visible, the connection would still exist and would only stop if the application was destroyed.

When onResume() was called a new connection would start even if a previous connection was already in place, this made the connection that was established stop and the surveyor had to


Figure 4.11: SMART's gitlab-ci YAML file

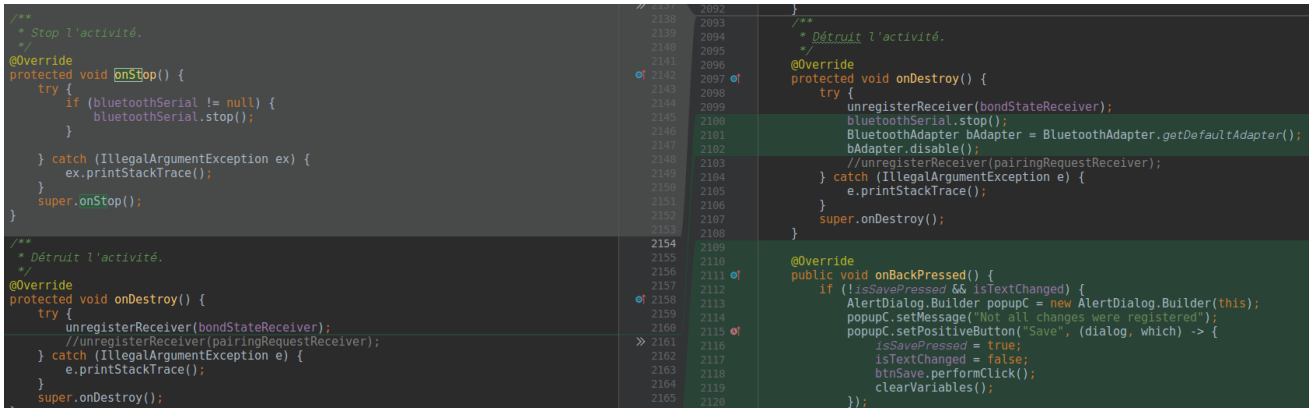


Figure 4.12: Changes to onStop() and onDestroy() - on the right side the changes performed

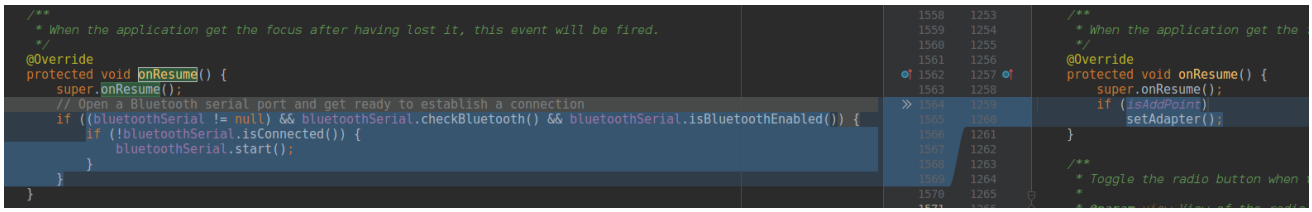


Figure 4.13: Changes to onResume() - on the right side the changes performed

manually connect to the instrument again. To fix this problem, this piece of code was removed.

### 4.3.8 Changes to the software development process at the APC section

As written in Section 4.2.2, at the APC section Kanban was the methodology adopted. As work evolved, some changes needed to be done to this process.

There was a need to meet the product owner and the section leader more often, to verify that changes met the requirements and the used development process fitted mobile development. This implied scheduling short daily meetings with both. Since this was an improvement for our development process the whole section adopted this activity.

The users should write all requirements in JIRA with a priority status/scale, and developers should tag every release on GitLab, listing all changes done and linking all JIRA tasks resolved on that release. With JIRA it is possible to receive user feedback, prioritize tasks and plan successive beta-releases to be tested in the field intensively. This strong cooperation with the users is very important in this context, where specifications are not always clearly defined, where the code needs to be adapted to new demands. Given the feedback received, users have the feeling to follow and actively participate in the developments.

As previously said, merge requests could only be accepted if the code was reviewed by at least two developers, where one was not directly involved in the project. With this activity, more errors in the code and improvements to the written code were found at each merge request and the section also adopted this into the development process.

### 4.3.9 Features added

While implementing the development process, the product also evolved. As the code was more readable and maintainable, it was easier to add new functionalities, always considering the set of good practices in our development process.

One of the main goals of the application is to simplify the measurements done by surveyors, so having Bluetooth communication with more instruments was the main requirement for this product. Bluetooth communication for the offset module was developed and SMART is capable of connecting to offset measurement devices and collect data automatically by pressing only one button in the offset staff.

More validations to the measurements were added so surveyors would increase their confidence when performing measurements. The first step was to analyse why the root mean square (*EMQ*) was not being calculated correctly. Two main problems were detected, first, the value of the *EMQ* was never set to zero, so for each station, the value was calculated and added the value of the previous station. Secondly, if there were blank lines, this would happen if the surveyors did not measure one or more points in the backward measurement that had been measured in the forward measurement, they would count as measured as zero, when that measurement should not be taken into consideration in the calculation of the *EMQ*.

After correcting this error and after it was approved by the users, more validations were added regarding the *EMQ*. The *EMQ* was calculated between forward measurements and backward measurements. In this last type of measurement, if the validations would fail this would cause a *Reprise* (Recovery). The calculation of the *EMQ* between the forward measurements and the recovery measurements was added, and the calculation of the *EMQ* between the backwards measurements and the recovery measurements. With this, surveyors can verify if the repeated measurements are

correct in reference to the forward measurements or if there was an improvement in reference to the backwards measurements.

It was added the notion of a station to the code of the application in order to calculate the height difference between two consecutive stations and to improve the code readability and maintainability. Whenever several points are measured from two consecutive stations, SMART computes the height difference between those stations for each point which can detect instrument movements between measurements, which means that measurements should be repeated.

$$ct = (xAller + rallAller) - (xRetour + rallRetour) \tag{4.1}$$

In 4.1, ct is the height difference between two stations, xAller is the measurement value in the forward measurement and xRetour the value of the measurement in the backward measurement. If needed, an extension on the ruler can be added, that value is represented by the *rallonge* (extension), and the rallAller is the value of the extension added in the forward measurements, and rallRetour the value of the extension added in the backwards measurement.

The Reader class reads the information of the files provided by the user, such as the file with the list of instruments to choose from. It was extended to read files with the name *Aller* (forward) in them. Whenever a surveyor did forward measurements of levelling measurements, these measurements were saved in a file with the name op_date_ALLER.dat, were op was the operation name and date the date of the operation. In some cases, surveyors performed the forward measurements in a day and go back another day to do the backward measurements. If a surveyor entered an operation name and choose, in the levelling module, to go directly to the backward measurements, SMART would verify the names of the files in memory and if a file corresponded to the operation name and had *Aller*, the stations or points would be loaded and the backward measurements could be resumed as if the measurements were all done in the same day (the *EMQ* was still calculated between the forward and backward measurements), if no file was found, a warning would advise the surveyor and no points or stations would appear.

A set of validations was added, such as a critical distance warning, in the levelling module, if the distance between the staff and the level is in a critical distance range the accuracy decreases, so a warning is sent to the user.

Collection of the distance to the instruments and their temperature was added for the levelling module while doing measurements via Bluetooth. The temperature collection is necessary as the levelling staff length is depending of the temperature due to the thermal expansion of materials. As a result, by collecting the local temperature, it is possible to apply a correction for each measurement.

As digital levels are used for the levelling, surveyors need the distance from the instrument to the levelling staff. There is a charge-coupled device (CCD) in the line of sight that takes a picture

of the levelling staff. Then, comparing this image with the image stored in the level, the processor is able to determine the height (i.e. the reading or measure). But in some cases (at the critical distances), the accuracy decreases because the size of the smallest element of the projected staff code on the CCD captor is very near from the pixel size. In this case, the correlation between the picture taken and the picture in memory is the worst. We are able to compute the distances to avoid. In fact, it depends on the pixel size of the CCD (14$\mu$m for the DNA03), the focal length (185mm for the DNA03) and the height of the smaller element of the staff code (2.025mm for the staffs used at CERN). The reason why the observed distance was needed was to compare it to the critical distances and in case of an approximation, a warning to the user appears.

A research was done in order to verify what requirements were necessary to be fulfilled in order to add a module for the gyroscopic measurements. It was analyzed what steps are taken to perform these measurements with the current equipment and software. The equipment was composed of two instruments connected and the measurements took quite a long time, to measure a point it took almost 45 minutes. After seeing that the equipment was very old (20+ years) and that it had no documentation of what possible commands could be sent to the equipment to request the measurements nor how to connect to SMART and observing that, if the instrument was connected via USB to a laptop, it would write the measurements directly into a terminal, it was decided that this measurements should be taken in a laptop.

### 4.3.10 Tests in the LHC-tunnel

When the application became stable enough to test on the field, tests in the LHC-tunnel were done and features were added to the application.

Whenever these features provided an improvement to the goal of the application, a release was delivered to the surveyors.

Tests in a workshop or at the office were scheduled weekly with different surveying teams. Tests on the field were weekly scheduled and releases by the developers were monthly scheduled, so the development process implemented could be adapted to the feedback reported by the surveyors and the developers.

The tests in the LHC-tunnel consist on having a team of two people or more. To go to the LHC-tunnel it is necessary to pass a set of both physical as written tests and to have certain access permissions. Asking for access can take from hours to days which can be a problem if tests are required urgently. Some areas of the LHC-tunnel are not accessible to some members due to controlled radiation areas. The team needs to take all the equipment necessary to do specific measurements that were scheduled and to take specific Personal Protective Equipment (PPE) like helmet, safety mask, among others. After descending into the LHC-tunnel, the team needs to be aware that other teams are also working in the tunnels. Work can be interrupted at any time either by other teams or by alarms, since there are always dangers when working in the LHC-tunnels like oxygen deficiency.

The measurements can take a few hours or weeks, depending on the number of points or stations to be measured and the type of measurements.

As shown in previous images, the instruments for the measurements are very big and very heavy and to have the liberty to not take a laptop and just take a smart phone is already a big plus on the field.

In one specific test, using SMART to do offset measurements, as explained previously, in the offset measurements a stretched wire is used, SMART gave an alert that the measurements were not good, and after a quick analyses in the field, the surveyors concluded that the wire was moved, and that the measurements needed to be repeated. This error could have been very difficult to spot and if the surveyors concluded the measurements without noticing it in the field, it would only be noticed when uploading the measurements into the database and they would have to repeat the measurements another day.

While testing the application on the field, it was also an opportunity to verify that the application ran the way the surveyors expected and it was also possible to collect more information about how surveyors really used the application and to understand their needs better.

## 4.4 Summary of the proposed process

In this Section, a summary with a description of the proposed development process is presented and the changes to the development process during the implementation phase are addressed.

### 4.4.1 Changes to the existing development process

As changes to the code were done and tests were performed, changes to the development process were also implemented to address the users and developers feedback. In the beginning of the development of SMART, there was not a lot of knowledge on how it was used and how. It was hard to understand the tasks to do and how to present what was asked in the application. It was also hard to understand why the users needed those tasks to be implemented. To change this, comments were added to the code, coding conventions were followed and tests on the field with developers were added to the development process. Once a week surveyors had to test the application either on a workshop or at the office with a developer and explain how they used SMART and what they wanted SMART to do in the near future and why. Some tests were also executed in the field with developers monthly. This way developers could see and understand better the work being carried out by surveyors and why SMART was so important. Meetings with the product owner and the team leader were scheduled weekly and daily a brief discussion of the work to be developed. This helped to understand if the work being developed was going accordingly to what was expected from the surveyor side. As the work progressed and more tasks were put on the to do list, more documentation was needed to understand certain tasks that were to be implemented, so documentation was also an important step that was needed, not only on the developers side but also on the users side.

In overall, the goal was to increase the involvement between surveyors and developers so cooperation would make the work of both run smoothly and easily.

Throughout this development process, releases to the product were done more often and more tests were also carried out.

### 4.4.2 The proposed development process

The first step would be to define the product owners that would manage the Kanban board. The product owners and the development team need to define together the life cycle, requirements, architecture of the software system and design. This requires to know well the development team and the envisioned product. The development team should then define coding standards for all members to follow and when to do code inspections.

Figure 4.14 describes the development process approach implemented.

In Figure 4.15, the workflow of the development process proposed is presented. In Phase 1, the development team and the product owner collectively choose a set of user stories with more priority. This prioritization of the user stories/tasks helps to achieve a faster product increment. In Phase 2, after this selection, the development team has to create a new branch for each feature and develop, commenting the intent of the code and documenting whenever a reference is necessary for future users or developers, the code has to pass the automatic tests implemented. This will help to find bugs sooner. In the case of SMART, the development team performs tests in the office with the instruments used by surveyors, whenever this can be applied, to verify that the application performs as intended. In a generic case, the developer should test the application before it is released. Then a code review must be done by at least two other developers and only after its accepted, the code can be merged to the master branch. After deployment, in Phase 3, tests to the product should be done with both parts on the field, so the involvement between the product owners and the development team always increase or maintain, this way an agreement on all the work being done is achieved at all moments and developers can understand better the environment where the application is used. In SMART these tests are performed in the LHC-tunnel. If a task is not accepted, a meeting is held with the product owner and the development team to see what is missing and to clarify what was expected and was not delivered. And this process restarts. Code reviews and inspections are methods to be applied whenever possible.

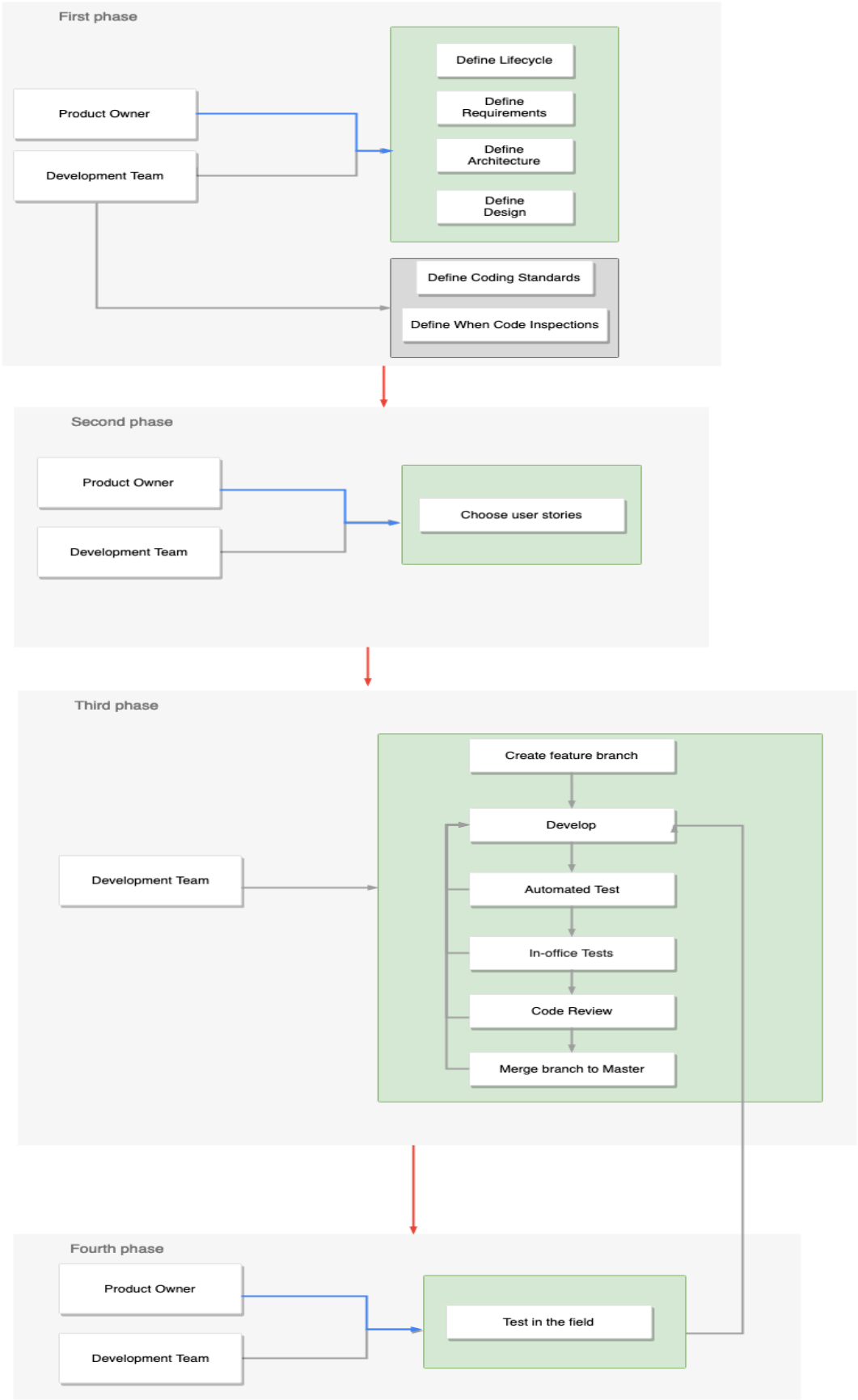


Figure 4.14: Proposed development process

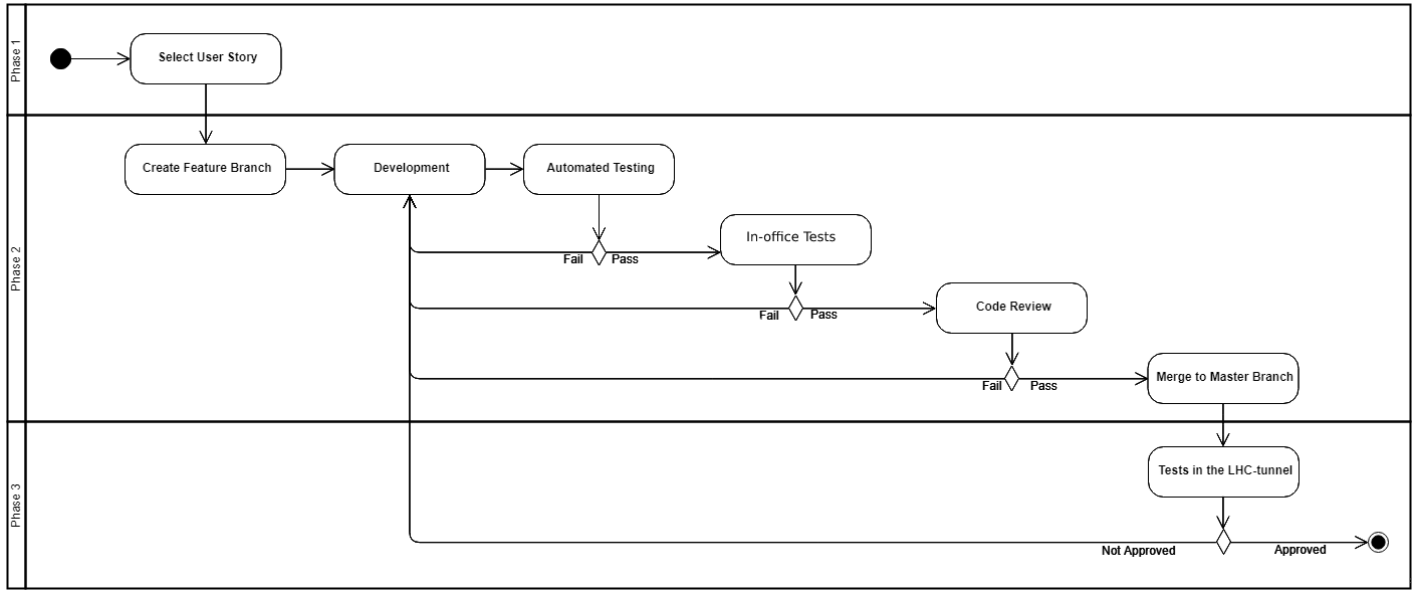


Figure 4.15: Workflow of the development process proposed for SMART

# Chapter 5

# Validation and Results

This chapter presents the results of the validation of the development process and of the product. In Section 5.1, a cumulative flow diagram is shown and explained. In order to have feedback and the appreciation from the surveyors and the developers team from the Acquisition, Process and Control Software (APC) section, a survey was conducted. The results of the questionnaires are shown in Section 5.2.

## 5.1 Validation of the process

Survey Measurement Acquisition in Real Time (SMART) was used as a mean of validation of the development process.

The cumulative flow diagram, seen in 5.1 is a fundamental tool of Kanban method. It allows the team to visualize their effort and the project progress. It shows the way the tasks mount up and their distribution along the different process stages. The graph is built from different colored bands of tasks gathered in different columns. One color per column - each band tells you how many tasks sits in what stage of the process in a given time (the horizontal value).

The product backlog did not exists and so all tasks that needed to be done were in a To Do list. Every week a meeting with the product owner and team leader was held to see what tasks had more priority. This is translated in a To Do band always growing and always bigger than the other bands.

The good part is, the number of tasks done is always growing.

The sudden rise in the band of the ready for release tasks was due to an accumulation of Done tasks and in QA. The product owner was on vacations and upon returning many tasks were marked as ready for release at once.

A new developer came in April and took over the project. This developer is a student from the same school as the previous two students who started to develop SMART and has the same background as them. He adopted the created development process and as we can see from the cumulative flow diagram, there was no sudden rise within any band of tasks that would point to an issue or the stall of the development process as previously had happened.

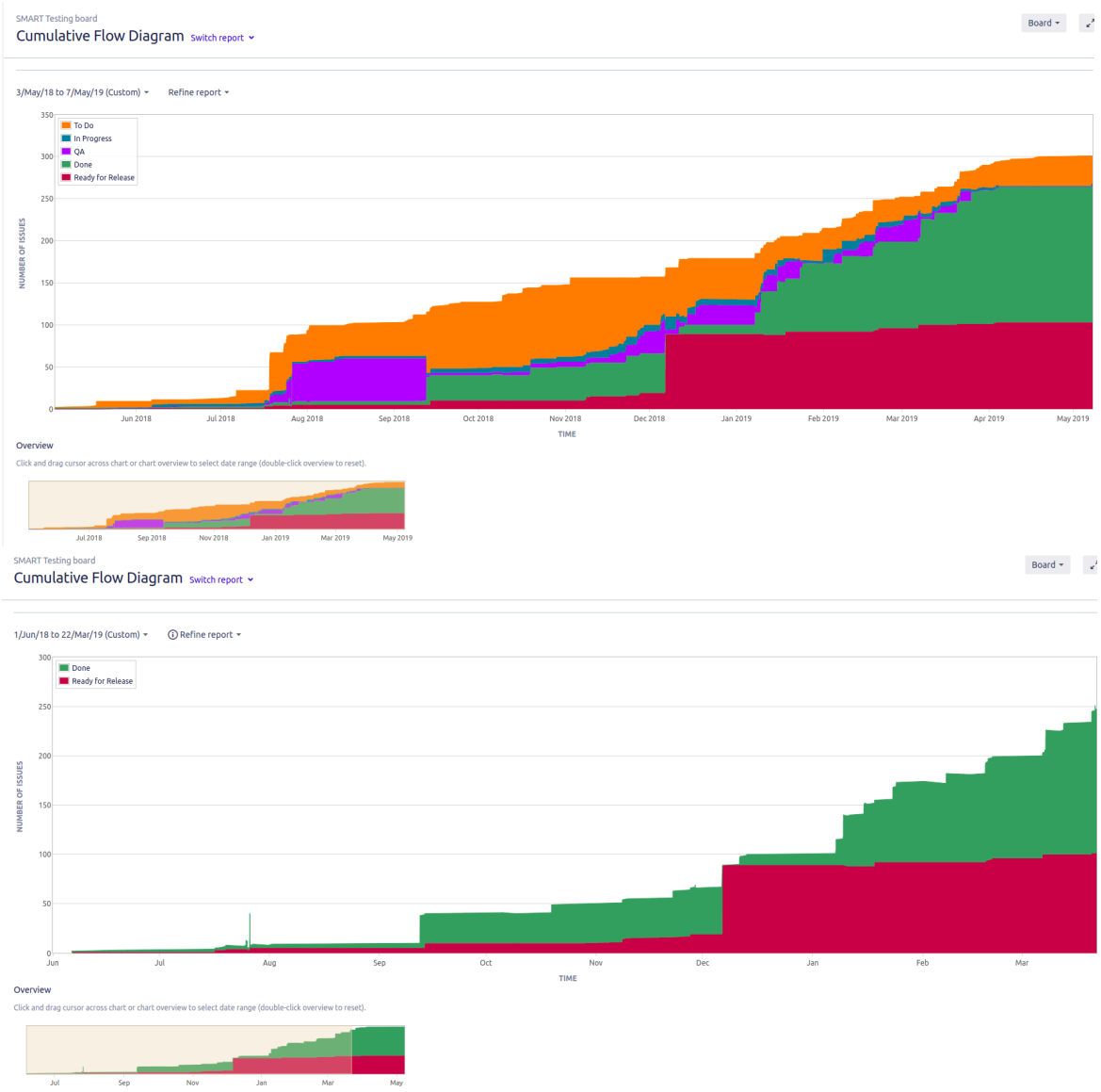


Figure 5.1: Cumulative Flow Diagram from JIRA

## 5.2 Validation of the product

To validate the requirements, tests in the office, workshops and the field were done. Different teams of surveyors tested the application each week, testing the different modules and the new features. Due to the team being small, a questionnaire was done by 6 members of the Engineering - Survey, Mechatronic and Measurements, Acquisition, Processing and Control Software (EN-SMM-APC) section.

The questionnaire was constituted by 13 questions related to the use of the application before the proposed development process was implemented and after its implementation. The list of questions is the following:

1. What is your role?

2. When did you first use SMART?
3. How satisfied were you using the app? (before October)
4. How satisfied were you using the app? (after April)
5. Did the application serve its purpose? (before October)
6. Did the application serve its purpose? (after April)
7. Would you use the app on the field? (before October)
8. Would you use the app on the field? (after April)
9. How intuitive was the application? (easy to use) (before October)
10. How intuitive was the application? (easy to use) (after April)
11. Was your opinion taken into account as a user during the development process?
12. Do you believe the approach followed by the developers after October gave value to the application?
13. Did the approach followed by the developers after October made communication between developers and users easier?

In Figure 5.3, the results indicate that there was an improvement in the satisfaction the users had towards the application.

In Figure 5.4, it is clear that users already thought the application served its purpose but it is visible an improvement on that feeling.

In Figure 5.5, it can be observed that the application was improved to the point where the users want to use the application to assist in their surveying works.

As shown in Figure 5.6, there is not a big difference in the usability of the application, since our focus was application functionality rather than user experience.

The feel that there was an improvement in each question can be confirmed by Figure 5.7, where all answers were taken into consideration.

In Figure 5.8, we can observe that the users feel like their opinion was taken in consideration and that the approach followed gave value to the application developed. A majority also believes that this approach made communication between developers and users easier.

In the following Chapter we present the conclusions from this dissertation and talk about future work.

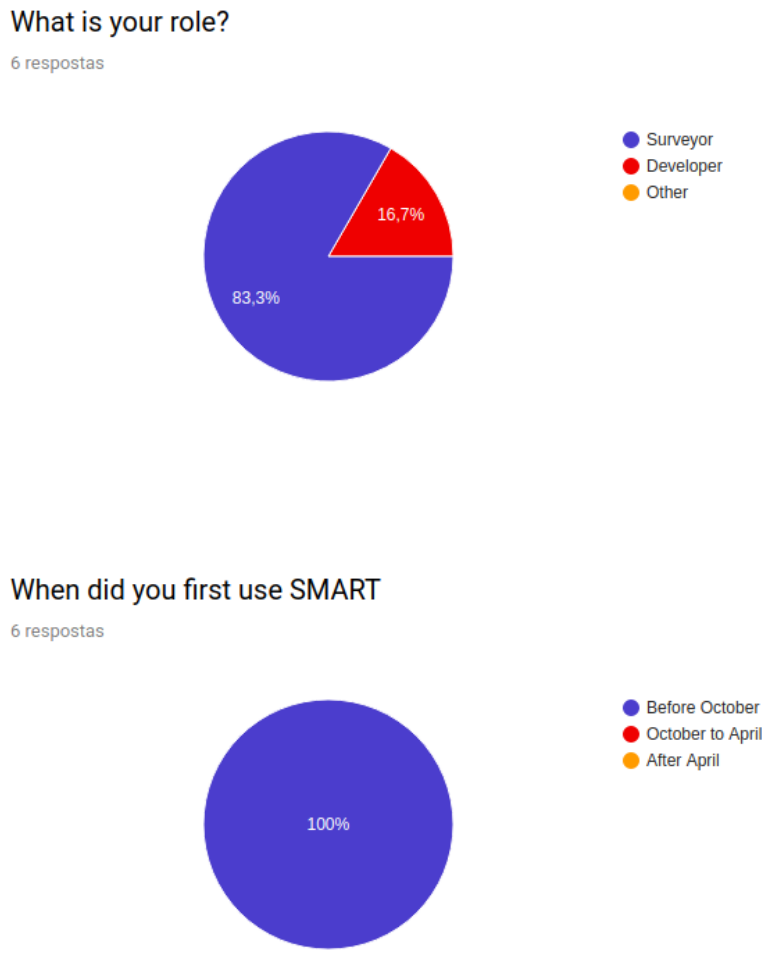


Figure 5.2: First plot: Percentage of users per role; Second plot: Amount of users who used SMART before or after the implementation of the new development process.

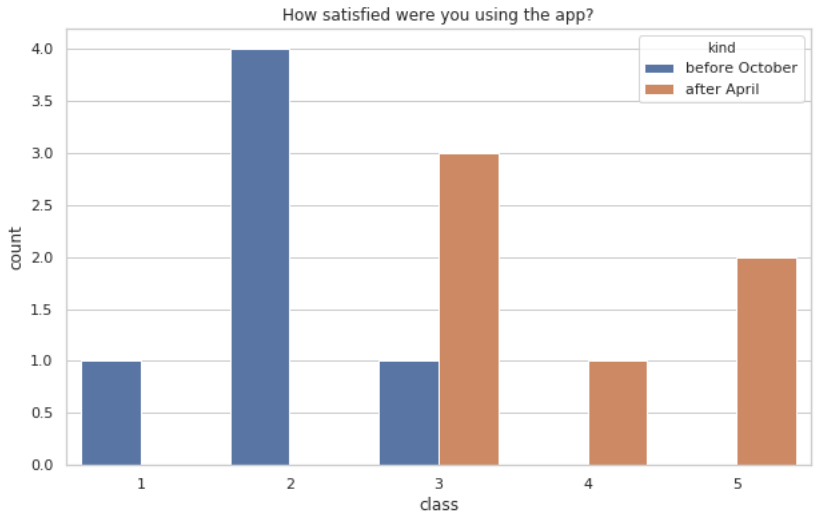


Figure 5.3: Amount of answers per class of satisfaction of SMART before and after the implementation of the development process

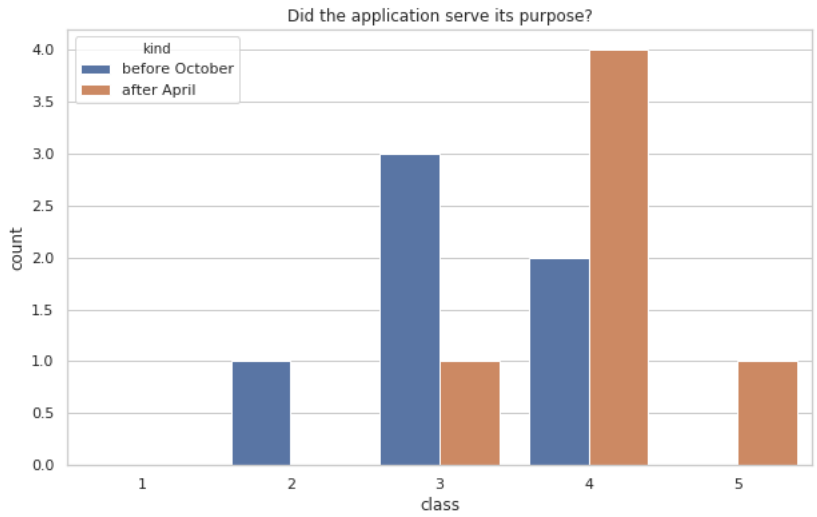


Figure 5.4: Amount of answers per class of users that believed SMART served its purpose before and after the implementation of the development process

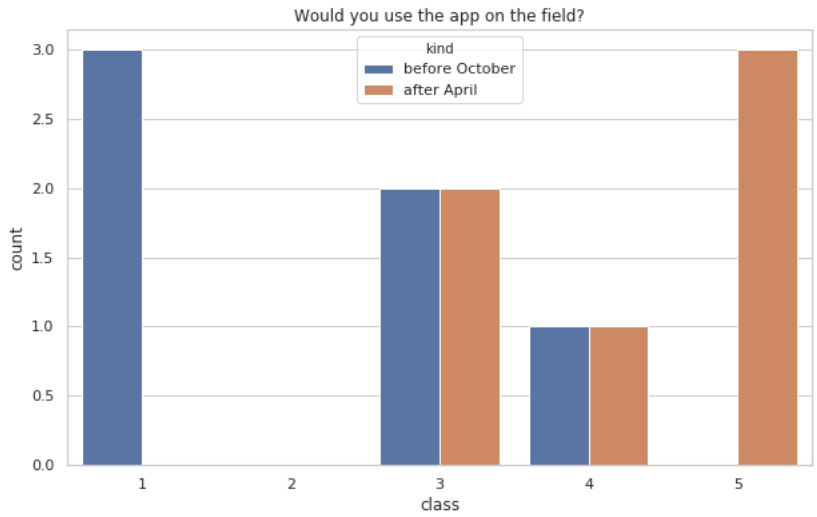


Figure 5.5: Amount of answers per class of users who would use SMART on the Large Hadron Collider (LHC) tunnel before and after the implementation of the development process

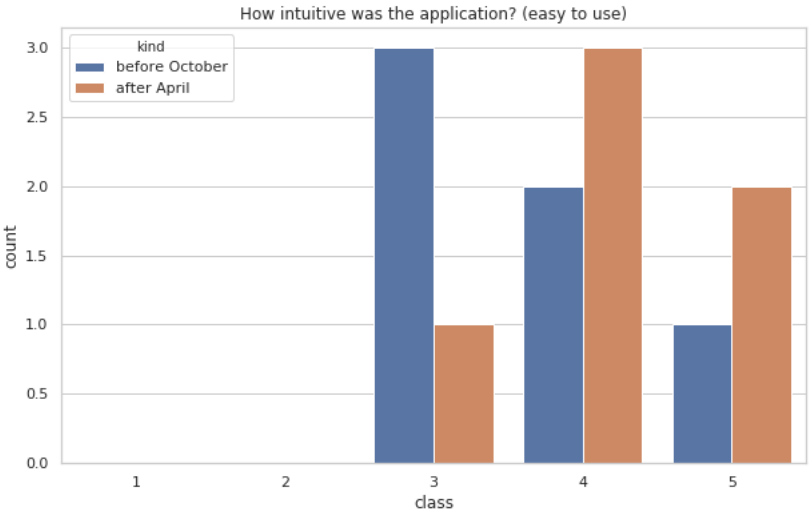


Figure 5.6: Amount of answers per class of usability of SMART before and after the implementation of the development process

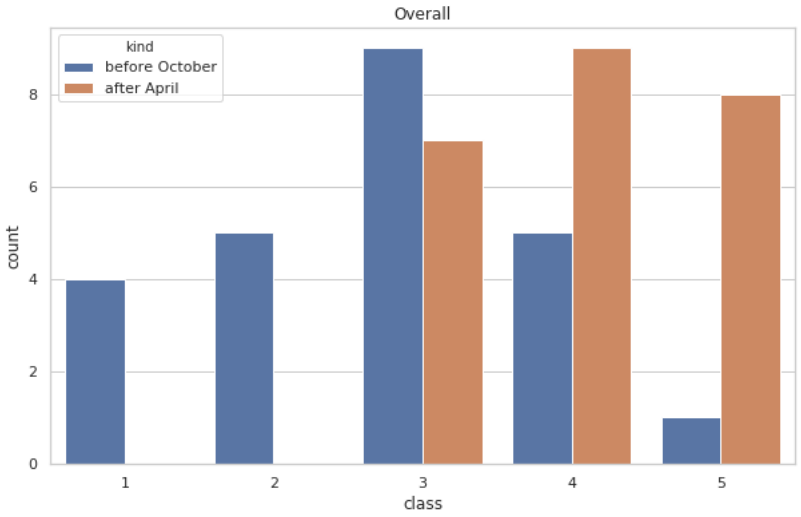


Figure 5.7: Aggregation of all answers

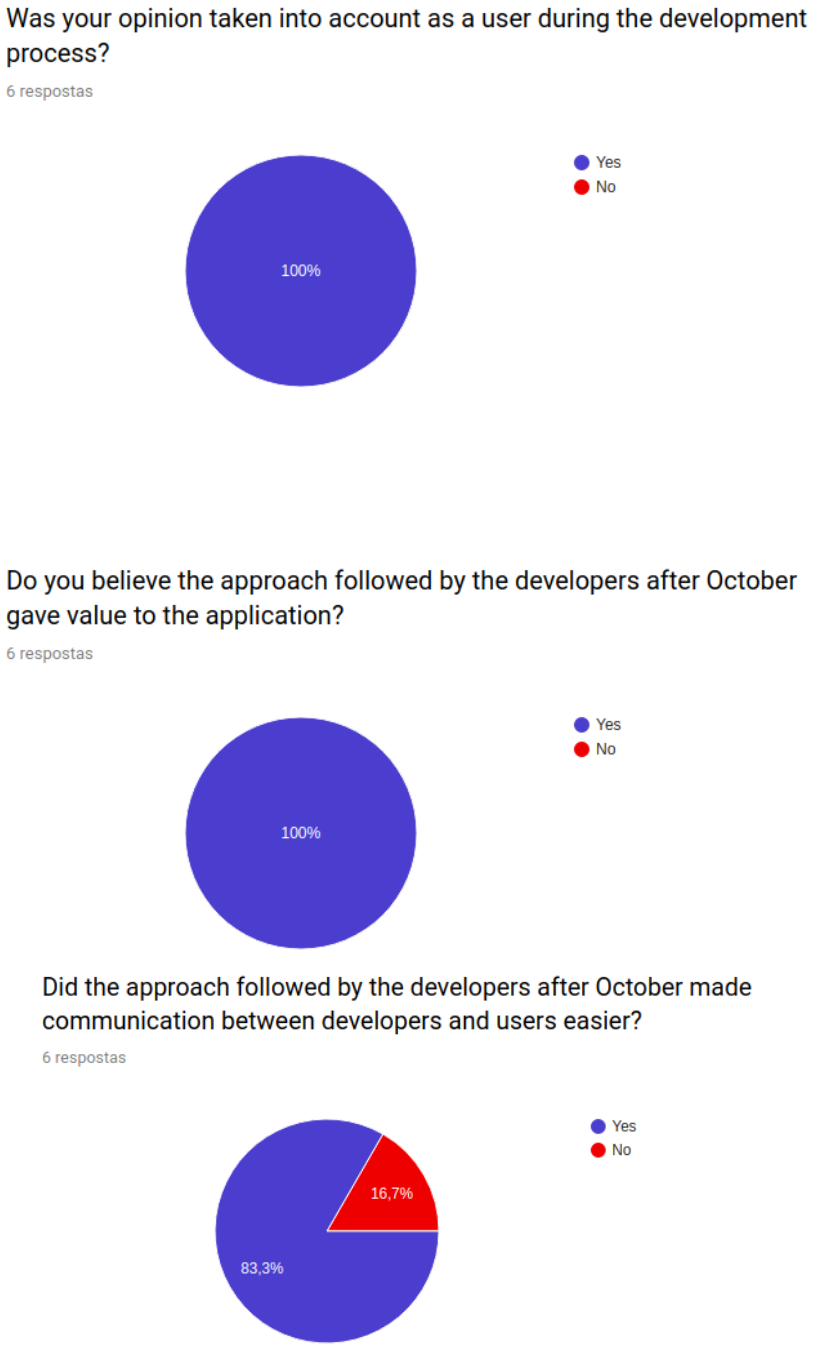


Figure 5.8: Results to questions based on user involvement

# Chapter 6

# Conclusions and Future Work

In this thesis, a development process was presented in order to refactor and extend an existing application, Survey Measurement Acquisition in Real Time (SMART), that was not extendable nor maintainable anymore.

This process was based on the integration of good practices of Software Engineering for mobile development, more specifically for Android application development. As many studies support, following an Agile methodology we could deliver small increments of the product to the users and adjust our process to the feedback received.

While implementing our proposed development process, difficulties were encountered, the fact that the code was written by two students without any experience meant that the code was not readable and so it was very time consuming to try to improve the code. The first step was to understand the whole application logic and check how it translated in the code. This was also due to not having a defined process or methodology to follow. In order to try to understand the code, many tests had to be made and, in the beginning, these tests were manual, causing a time delay. We also had constraints, such as the testing environment, since going to the Large Hadron Collider (LHC) tunnels is not always possible, the tunnels are not open for weeks when the beam is on, and even if the tunnels are open, a team had to be available to go down to them. There was also the constraint of not always having the measuring devices available to take down to the tunnels or to even test locally.

The only technologies being used were Android Studio to write the code and run the application, GitLab to store the code and JIRA to follow tasks and no tools were used to automize, test or verify the quality of the code or the application. The introduction of good practices, as Continuous Integration and Continuous Deployment (CI/CD), unit testing, among others, allowed the finding of problems earlier in the development process and also a possibility to correct those problems before the users test the application on the field.

The results of this thesis contribute to software development at the European Organization for Nuclear Research (CERN), allowing other developers to follow a defined process that can be adapted to the user's needs. Another contribution was, for example, users that develop Android application can use the Android image for Docker, instead of creating one from scratch. With

this product, measurements are done faster and with more validations, making the surveyors work easier and less prone to errors.

Next steps to further develop and validate the process consist in the continuous extension of the application and to apply the developed process to other applications.

Since technology at CERN is always evolving and researchers are always trying to improve their systems, a need for mobile applications will appear and by using this development process from the beginning, developers can follow a single methodology and improve the product in a short time period.

The first step would be to define the product owners that would manage the Kanban board. The product owners and the development team need to define together the life cycle, requirements, architecture of the software system and design. This requires to know well the development team and the envisioned product. The development team should then define coding standards for all members to follow. Then develop and test the required user stories. The involvement between the product owners and the development team should always increase or maintain so an agreement on all the work being done is achieved at all moments, for this, tests to the product should be done with both parts. Code reviews and inspections are methods to be applied whenever possible.

Another developer will keep extending the application and a study needs to be done on how the process will grow and how the developer adapted to it. Other features already requested was for SMART to be able to compute approximate co-ordinates of stations using the theoretical co-ordinates of points measured (in the theoretical file) and the distance measured in the field. These coordinates are needed to apply a correction to the measurements: the sphericity correction, due to the earth's curvature. This correction should be applied to each point measured. Another correction requested was to do with the vertical deviation that requires knowing the approximate co-ordinates of the station. Also listed as future work is the creation of a direct connection to the database, eliminating the need of a manual insertion of the data into the database. Developing more tests is also an important task to be performed in a near future.

# References


[1] Android | the world's most popular mobile platform. https://www.android.com/, 10 June 2019.

[2] Design patterns and refactoring. https://sourcemaking.com/design_patterns, 20 June 2019.

[3] Iso/iec/ieee international standard - systems and software engineering–vocabulary. *ISO/IEC/IEEE 24765:2017(E)*, pages 1–541, Aug 2017.

[4] Android is enabling opportunity. https://www.android.com/everyone/enabling-opportunity/, 28 December 2018.

[5] Alain Abran and James W. Moore. *Guide to the software engineering body of knowledge*. IEEE Computer Society, 2004.

[6] Anureet. *Review on Agile Approach to Mobile Application Development*, volume 3. 2016.

[7] Atlassian. Jira | issue & project tracking software. https://www.atlassian.com/software/jira/, 28 December 2018.

[8] M A Awad. *A Comparison between Agile and Traditional Software Development Methodologies*. 2005.

[9] Yoonsik Cheon. Are java programming best practices also best practices for android? *International Conference on Advances and Trends in Software Engineering (SOFTENG 2017), Venice, Italy, April 23-27, 2017.*

[10] Google Developers. Understading the activity activity lifecycle | android developers. https://developer.android.com/guide/components/activities/activity-lifecycle/, 10 June 2019.

[11] Google Developers. Application fundamentals | android developers. https://developer.android.com/guide/components/fundamentals, 21 June 2019.

[12] Erik Dietrich. What is a best practice in software development? https://daedtech.com/what-is-a-best-practice-in-software-development/, 2015.

[13] Norbye Tor Chou Katherine Ducrohet, Xavier. Android studio: An ide built for android. https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html, May 2013.

[14] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Syst. J.*, 38(2-3):258–287, June 1999.

[15] Harleen Flora and Swati Chande. A review and analysis on mobile application development processes using agile methodologies. *International Journal of Research in Computer Science*, 3:8, 07 2013.

[16] Martin Fowler. Extreme programming examined. chapter Is Design Dead?, pages 3–17. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[17] Martin Fowler. bliki: Continuousintegrationcertification. https://martinfowler.com/bliki/ContinuousIntegrationCertification.html, 2017.

[18] Martin Fowler. Refactoring home page. https://refactoring.com/, 28 December 2018.

[19] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition, 1994.

[20] GitLab. Particle physics laboratory uses gitlab to connect researchers from across the globe. https://about.gitlab.com/customers/cern/, 28 December 2018.

[21] Rashina Hoda and Latha K. Murugesan. Multi-level agile project management challenges. *J. Syst. Softw.*, 117(C):245–257, July 2016.

[22] Ming Huo, June Verner, Liming Zhu, and Muhammad Ali Babar. Software quality and agile methods. *Computer Software and Applications Conference, Annual International*, 1:520–525, 10 2004.

[23] Docker Inc. Get started with docker. https://www.docker.com/get-started, 28 December 2018.

[24] Shakira Banu Kaleel and Ssowjanya Harishankar. *Applying Agile Methodology in Mobile Software Engineering: Android Application Development and its Challenges*. 2013.

[25] Gaurav Kumar and Pradeep Bhatia. Comparative analysis of software engineering models from traditional to modern methodologies. 02 2014.

[26] N. Medvidovic and R. N. Taylor. Software architecture: foundations, theory, and practice. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 471–472, May 2010.

[27] Alok Mishra and Deepti Mishra. Software project management tools: a brief comparative view. *ACM SIGSOFT Software Engineering Notes*, 38:1–4, 2013.

[28] D Missiaen, J P Quesnel, and Ralph Steinhagen. The alignment of the lhc. *Proceedings of PAC09, At Vancouver, BC, Canada*, 07 2009.

[29] Jürgen Münch, Ove Armbrust, Martín Soto, and Martin Kowalczyk. *Software Process Definition and Improvement*. Fraunhofer IESE, 2009.

[30] Helena Holmström Olsson, Hiva Allahyari, and Jan Bosch. Climbing the "stairway to heaven" - A mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *EUROMICRO-SEAA*, pages 392–399. IEEE Computer Society, 2012.

[31] F. Klumb P. Valentin, J.F. Fuchs. Development of a stand-alone software for data acquisition. *International Workshops on Accelerator Alignment (IWAA)*, 2018.

[32] F. Paetsch, A. Eberlein, and F. Maurer. Requirements engineering and agile software development. In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, pages 308–313, June 2003.

[33] Srđan Popić, Gordana Velikić, Hlavač Jaroslav, Zvjezdan Pavkovic, and Marko Vulić. The benefits of the coding standards enforcement and its impact on the developers coding behaviour-a case study on two small projects. 11 2018.

[34] Asad Qazi, Sidra Shahzadi, and Mamoona Humayun. A comparative study of software inspection techniques for quality perspective. *International Journal of Modern Education and Computer Science (IJMECS)*, 10:9–16, 10 2016.

[35] Arvind Rongala. 20 best practices for successful software development projects. https://www.invensis.net/blog/it/20-best-practices-for-successful-software-development-projects/, Apr 28 December 2018.

[36] Alan Santos, Josiane Kroll, Afonso Sales, Paulo Fernandes, and Daniel Wildt. Investigating the adoption of agile practices in mobile application development. 04 2016.

[37] Sheetal Sharma, Darothi Sarkar, and Divya Gupta. Agile processes and methodologies: A conceptual study. *International Journal on Computer Science and Engineering*, 4, 05 2012.

[38] IEEE Computer Society. 730-2014 - ieee standard for software quality assurance processes, 2014.

[39] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.

[40] Daniela Steidl, Benjamin Hummel, and Elmar Juergens. Quality analysis of source code comments. *2013 21st International Conference on Program Comprehension (ICPC)*, 2013.

[41] Sean Stolberg. Enabling agile testing through continuous integration. *2009 Agile Conference*, 2009.

[42] Anthony Wasserman. Software engineering issues for mobile application development. pages 397–400, 01 2010.

[43] Loo Wooi Khong, Leau Yu Beng, Tham Wai Yip, and Tan Fun. Software development life cycle agile vs traditional approaches. *International Conference on Information and Network Technology, At Chennai, India*, February 2012.

[44] Loo Wooi Khong, Leau Yu Beng, Tham Wai Yip, and Tan Fun. Software development life cycle agile vs traditional approaches. 02 2012.

[45] Mario Špundak. Mixed agile/traditional project management methodology – reality or illusion? *Procedia - Social and Behavioral Sciences*, 119:939–948, 2014.