|  |
| --- |
| Group 02 |
| Quality Management |
| Project Quality |
| G.U.A.R.D |
| Gothenburg University Assistive Response Device |
|  |

|  |
| --- |
|  |

**Supervisors**

Maria Chiara Lucatello

Mayra Soliz

**Team members**

Axel Granli

Boyan Dai

Erik Laurin

Gabriel Marian Bulai

Joacim Eberlen

Justinas Stirbys

Shaun McMurray

Contents

[1. Introduction 2](#_Toc481437838)

[2. Quality Planning 2](#_Toc481437839)

[2.1 Thresholds 2](#_Toc481437840)

[2.2 Frequency and process 2](#_Toc481437841)

[2.3 Justification 3](#_Toc481437842)

[3. Software Quality Assurance 3](#_Toc481437843)

[3.1 Requirement Engineering 3](#_Toc481437844)

[3.2 Software Architecture 3](#_Toc481437845)

[3.3 Testing 4](#_Toc481437846)

[3.4 Process Following 4](#_Toc481437847)

[4. Conclusion 4](#_Toc481437848)

[References 5](#_Toc481437849)

# 1. **Introduction**

The project aims to increase the personal security of individuals traveling Gothenburg on foot by utilizing a SmartCar that follows and monitors the individual. This will hopefully lead to a quick response to any potential threats. To achieve such complex functionality, the project has to incorporate various devices running high quality code. The three main components that makes up the system include a mobile application, a Raspberry Pi and an Arduino.

The mobile device application will act as a mitigator between the SmartCar and user, granting the user control over the SmartCar. The Arduino acts as a motor controller for the SmartCar. Lastly, the Raspberry Pi shall act as a master for the Arduino. Algorithms and code for the more complex functions, such as GPS following and obstacle avoidance, will be executed on the Pi.

Perception of quality differs amongst users, with different people focusing on different aspects of the system and what quality is according to their definition of the term. Therefore it is difficult to define quality[1]. To obtain a definitive perception of quality amongst users, a unified definition of quality must be established. This definition will take the following areas of the project into account: code quality and the quality of hardware. To measure external quality, emphasis must be put on analyzing the system’s internal attributes.

The quality of code will be measured with aid of internal quality attributes, such as cyclomatic complexity and block depth. The code will be considered of acceptable quality if these attributes meet the established thresholds. The SmartCar’s hardware quality will be defined in terms of performance and compliance to the commands provided by the mobile application. To analyze hardware quality, instrumented unit tests shall be carried out.

# 2. Quality Planning

We have defined our customer as a fictional security company. The said company will be granted ownership of the SmartCar and later outsource it to their clientele that wish to have extra protection in the form of small robot car following them and recording their movements. It was decided that for a security company, external attributes such as maintainability, reliability and usability are of the utmost importance. Therefore, throughout the development process focus will be put on these attributs.

## 2.1 Thresholds

The quality target is defined in terms of thresholds. To measure code maintainability, testability and simplicity in the project, the threshold for cyclomatic complexity was set to M <= 15, where M is the measure for cyclomatic complexity. Due to the Arduino’s single thread of execution, as a mean of making the sketch responsive, the threshold for the Arduino sketch has been set to M <=7. These thresholds will be used to find classes that need refactoring. To further enhance our measured quality, block depth was also considered when measuring internal attributes. The acceptable depth was designated as 7.

Hardware test shall be carried out by instrumented unit testing, with set requirements. The thresholds for hardware will be in accordance with the manufacturer’s specifications.

## 2.2 Frequency and process

The frequencies of test vary by the system aspect measured. The software test will be carried out once every sprint. After the software tests are run the developers will engage in code review to improve shared code ownership. After which goal oriented refactoring will take place if necessary. This aims to reduce code complexity if any classes/functions in need of refactoring are identified.

When major modifications are done to the hardware, the hardware tests will be conducted to verify that the SmartCar performance has not been compromised. The actual sensors will be used during the unit tests to verify their ability to provide data with sufficiently low uncertainty to fulfill their assignments, such as providing sufficient accurate data to stop the car before hitting an obstacle.

## 2.3 Justification

Throughout the development process the project is geared towards realizing external attributes; maintainability, reliability and usability. These attributes ensure that simple and maintainable code is created. Therefore, for quality analysis focus will be put on internal attributes that impact the desired external characteristic.

The choice to focus on cyclomatic complexity was due to it rendering an exponential reduction of test cases required to achieve full branch coverage [1]. This would lower the amount of resources required to properly test the application and thus making it more maintainable in the future. Furthermore, lowering the depth value within the project may result in increased readability which consequently increases system maintainability. The UI testing is used to find and remove the faults in the project, which further improves usability [2].

Ultimately, the application’s overall performance is only as good as the hardware used. Thus, testing the hardware is vital to verify that its performance is sufficed to fulfill our quality requirement. Moreover, the applications UI testing provides a measure of reliability as well.

# 3. Software Quality Assurance

In order to assure a high level of quality in terms of usability, maintainability and reliability, software will be analyzed using SourceMonitor and Visual Paradigm. Additionally, Trello is used for setting and keeping track of the project requirements and Espresso framework for testing.

## 3.1 Requirement Engineering

The project utilizes aspects of release management which involve both the customer and the development team. The developers think of desired features and break them down into tasks, accompanied by acceptance criteria, and added to Trello boards that are used to track requirements throughout the course of the project. The development process will be done in two week long sprints. At the beginning of each sprint the process made will be showcased to the customer i.e. product owner, for their input. GitHub is used to monitor the progress made and keep track of the product version.

## 3.2 Software Architecture

Software architecture shall be used as a means of quality assurance and will be discussed during sprint meetings. The meetings begin with examining the current architecture and continue by discussing possible improvements. This is done with the aid of UML domain modeling. During this period, the project’s source code is reverse engineered with the aid of Visual Paradigm to obtain a view of project structure and patterns. Focus is put on the coupling and cohesion of classes. The aim of this process is to provide a desirable level of maintainability. By reducing the amount of coupling, it becomes easier to maintain the software [4]. Similarly, maintainability can be increased by making highly cohesive classes that are singly purposed [5]. However, creating satisficing architecture is very time consuming. The risk is that the time spent on the architecture might not provide enough value to be worth it. Additionally, every Friday, code reviews take place where the developers get acclimated with others’ code to verify that high quality code is produced. Doing this helps to improve the code, thus the project’s maintainability.

## 3.3 Testing

To assure that the project is reliable, testing shall be conducted. The majority of which shall take the form of aforementioned UI tests. The selected framework for testing is Espresso, a UI testing tool developed by Google and JUnit. These UI test should be written for every function that interacts with the graphical interface, and thus impact user experience. When all features in the UI get properly tested, we have a good basis of quality in terms of critical failures and logical errors. After a new function has passed its particular test, regression tests shall take place, in order to assure that the old functions have not been compromised by the new addition. The regression tests are written as multiple test suites that run in a combined suite. Human resources impact the quality of software and are detrimental when determining the project’s success [3]. Therefore by following this process a more in-depth inspection of functions occurs possibly removing the threat of a human failure. This makes the project more reliable, which is crucial for a project in the security sector.

## 3.4 Process Following

Knowledge from past courses and experiences provided an initial definition of quality, in terms of coupling, cohesion and software design. This simple definition led to an initial process, which was expanded throughout the course. As the project transitioned more established practices, thresholds and requirements were incorporated into the process. With the aid of UI testing it was possible to greatly reduce the number of errors. SourceMonitor exposed the functions in need of refactoring that were managed to be fixed. However, in the beginning of the project there was a lot of focus on creating a product and getting everyone acclimated with developing for Android and Arduino. These leads to momentary abandon the focus on quality planning. Moreover, UI tests or any type of source code analysis did not occur for the first two iterations of development.

# 4. Conclusion

To conclude, it became clear that a quality assurance plan is an integral part of the project. This plan should make sure everyone in the development team knows what is required from each individual. A definite change would be to make an initial quality plan in order to obtain a quality standard from the beginning and save time that was later spent on refactoring. Firstly, by focusing on software architecture, the code produced became more modular and decoupled, creating a more maintainable project. Secondly, UI testing exposed faulty parts of the project that were overlooked by developers, by fixing these issues additional reliability was obtained. Thirdly, the tools used SourceMonitor, Visual Paradigm and Espresso, were of great benefit when aiming to creating a usable, maintainable and reliable system and would be used once more. With the feedback received from customers during demonstrations and focus groups, the project was able to collect data on user experiences and change the user interface based on the feedback. In the future a more in depth release management plan would be developed and followed.

# References

[1] David Chappell. The three aspects of software quality: functional, structural, and process. David Chappell & Associates, San Francisco, 2011. [Link](http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf)

[2] Gomes, Francisco, and Vard Antinyan. "Internal Software Quality and Measures." Gothenburg. 16 Mar. 2017. Lecture (PowerPoint presentation).

[3] B.S. Dhillion. Reliability in Computer System Design, 4.5(4):79, 1987

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological review, 63 (2), 81.

[4] James E. Tomayko, David P. Darcy, Sandra A. Slaughter, Chris F. Kemerer, "The Structural Complexity of Software: An Experimental Test", *IEEE Transactions on Software Engineering*, vol. 31, no. , pp. 982-995, November 2005, doi:10.1109/TSE.2005.130

[5] Larman, C. (2004). *Applying UML and Patterns* (3rd ed.). Upper Saddle River, New Jersey: Pearson Education.