LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Industrial Engineering and Management
Global Management of Innovation and Technology

Master's Thesis

Quality Management in Mobile Software Development

Daniel Quesada Otarola

Supervisors: Docent, Associate Professor Ville Ojanen
Professor Ajantha Dahanayake

Abstract

Author: Daniel Quesada Otarola
Title: Quality Management in Mobile Software Development
Year: 2018
University: Lappeenranta University of Technology

Type: Master's Thesis
Specification: 63 pages, 13 figures, 3 tables.

First supervisor: Prof. Ville Ojanen
Second supervisor: Prof. Ajantha Dahanayake

Keywords: Software Quality Management, Mobile Development, Agile Methodologies, Software Development.

Summary: Over the last decades the number of mobile devices and applications has grown exponentially. Therefore, the industry of mobile applications has emerged and it presents several software development challenges due to specific practices of the mobile world.
This research project explores if a team dedicated to build mobile applications can apply agile development practices while taking quality management standards into consideration.
To reach an answer, it explains the main characteristics of the mobile industry focusing on the development of applications. Then, it explores the principles of quality management and the main development frameworks to provide a clear understanding of both concepts.
For the empirical section, several interviews are conducted with experts of the mobile industry in the areas of software development, user experience design and project management. The interview questions are based on the literature and their goal is to learn about practices in the industry to compare them to the theory.
The research finds that there is a close relationship between the needs in the mobile industry and the principles of both agile development and quality management. However, while mobile teams rely on agile methods of development, quality management standards are not widely spread in the industry.

Acknowledgments

I would first like to thank my thesis supervisors Prof. Ville Ojanen and Prof. Ajantha Dahanayake at Lappeenranta University of Technology. Their advice, patience and guidance during this research has been invaluable and I will be forever grateful.

I would also like to thank Bruno Sarmiento, Jason Gamboa and many others who donated their time in several ways to contribute to this project. Whether it was with long interviews or small casual conversations, their inputs allowed me to validate the theory with real information on the mobile industry and gave depth to the project.

In addition, I must express my profound gratitude to my family. My mother, who is my main source of motivation and my brother and sister who have cheered me on from the distance during my studies.

My group of friends who have made Finland feel like second home: Amila, Abhishek, Alejandro, Sohail, Saeid, Soumyajit and many other people I met along the way. Special thanks to 'Valentinos', my friends back home who keep in touch with me and keep encouraging me during this years. And of course, Rafael, the best companion I could wish for this peculiar adventure.

Finally, my gratitude towards the finish people who made my life in Finland a time I will never forget.

Daniel Quesada
June, 2018.

Table of Contents

List of Abbreviations

SQM	Software Quality Management
SPI	Software Process Improvement
QLC	Quality Life Cycle
QMS	Quality Management System
SQA	Software Quality Assurance
SQC	Software Quality Control
CMMI	Capability Maturity Model Integration
KPA	Key Process Areas
SDLC	Software Development Life Cycle
ISO	International Organization for Standardization
ISTQB	International Software Testing Qualification Board

List of Figures and Tables

1. Introduction

1.1 Background

The accelerated growth of the mobile market, referring to tablets and mobile phones, has created a need for an approach to software development with slightly different characteristics and processes than the ones proposed by formal methodologies (Flora, Chande and Wang, 2014). Due the rapid change of the mobile business environment, software development processes must be assessed and adapted frequently (Oberscheven, 2013). And formal methods of development are not able to adapt to these changes in requirements because they depend on planning in advance for long periods of time (Fowler, 2017).

Agile methodologies have gained tremendous acceptance in the last two decades in part because they can deliver software in an environment where requirements, budget and time can be adjusted in several occasions (Huo, Verner, Zhu, and Babar, 2004,). Also, they tend to target collaborations between development and customers and deliver products early with constant improvement (Huo et al., 2004).

Mobile software development refers to manufacturing software for portable devices, characterized by small screens and low power (Alsabi, 2016). When developing mobile applications, teams encounter some special characteristics such as: 1) Short marketing times, 2) Requirements that change rapidly and 3) Early feedback from users (Alsabi, 2016). The volatility of the mobile application market makes agile development a viable choice for mobile development, since traditional software development fails to efficiently accommodate unexpected changes (Gartner, 2014).

According to Huo et al. (2004), agile methods have not proved that they provide the same level of quality than formal methods such as Waterfall. And in case they do, it is still not clear how the quality is achieved and guaranteed.

Software quality is defined as the "capability of a software product to satisfy stated and implied needs under specified conditions" (ISO, 2011 p.6). However, quality is dependent on the final product and on the quality of the development process itself (Oberscheven, 2013). For this reason, it is relevant to find out whether agile methodologies provide the appropriate processes to produce high quality mobile products.

Quality Management refers to establishing a "framework of organizational processes and standards that will lead to high quality software" (Sommerville, 2004 p.625). There are standards such as ISO 9001 and CMMI which provide best practices and processes to be applied by teams in the software industry. The standards aim to reduce errors and improve the overall quality of the organization (including the final product). However, there is a traditional belief that standards and agile methodologies are not compatible. This is because standards are perceived as bureaucratic and focused on documentation, while agile is all about the opposite (McMichael and Lombardi, 2007).

The number of low quality apps in the market is rising (Inukollu, Keshamoni, Kang, & Inukollu, 2014) and by evaluating a mobile development team based on Quality Management Standards, it will be possible to identify if their processes are in alignment with the theory's best practices related to documentation, requirements, design and code.

There is a research gap identified when it comes to linking the topics of Quality Management and Agile methodologies. Even though both concepts are focused on having processes in place that help a team to achieve the best results possible, it is not common to see agile teams implementing QM standards. Also, by introducing mobile development it is possible to reduce the scope of the research and focus it in an industry that has grown exponentially.

1.2 Research questions, objectives and scope

The main research question of the research is:

- Can software development teams work in an agile environment and follow the Quality Management standards while overcoming the challenges presented by the mobile industry?

Sub-questions:

- What are the main recommendations from a Quality Management standpoint for teams that develop software?

- What are the main challenges of developing mobile applications using an agile environment?

- Which development method is more suitable for mobile software industry?

Objectives of the thesis:

- ❖ Identify the main principles, recommendations and best practices present in the Quality Management theory.

- ❖ Explore the main characteristics and challenges faced by software organizations in the mobile industry.

- ❖ Compare the benefits and disadvantages of traditional and agile software methods, linking them to the mobile industry needs.

Scope and limitations

The research is focused on mobile application development. However, considering that there are many industries and companies that develop mobile applications, the research focuses mainly in the processes of mobile software development in an agile environment and quality management processes, without targeting a specific industry. The research question and objectives are developed by observing a research gap between Agile development and Quality Management standards, thus presenting an opportunity to combine Quality Management standards to mobile development teams working on an Agile environment.

In chapter 2 one of the standards mentioned is ISO 9001, in this document the focus will be in the version of ISO 9001 launched in 2008 because, even though there is a newer version (launched in 2015), most of the literature is written based on the version of 2008 since it has been studied and applied on a larger scale.

1.3 Research methodology

The thesis project is based on a problem oriented research, consisting of a systematic literature review (SLR) and an empirical part analyzed in a qualitative way. The SLR consists on finding, classifying and studying existing meanings, descriptions, definitions and case studies related to: Quality Management, Agile Methodologies, Mobile Development and any relevant combinations of those topics.

A systematic literature review has been selected due to the opportunity that the problem presents. However, given that mobile applications can be related to a vast number of industries and that many companies follow diverse development practices, it makes more sense to study the subject from a more general point of view. In this case, the thesis focuses specifically in the Quality Management practices of the mobile development cycle, disregarding any specific type of mobile industry.

The information collected in the SLR is then used to elaborate questions for the interviews, which objective is to find out about the practices and processes followed by mobile development teams and how Quality Management is perceived and applied by team members. Some questions are formulated based on practices described in CMMI framework related to process adherence and others are based on QM definitions on quality, to find out how teams aim to achieve quality deliveries of software. The interview questions can be found on Appendix 1 at the end of the document.

The thesis has an empirical part that consists on a set of semi-structured individual interviews conducted to obtain information about the processes and methodologies followed by mobile development teams. The experts involved are Jason Gamboa, a mobile developer with over 5 years of experience who has been involved in several projects of mobile software both for business to business and business to consumer. And Bruno Sarmiento, an iOS developer who was involved in an outsourcing project that build an application for sports. Both of them are developers who reside in Costa Rica and agree to several online interviews in order to cover the questions. Also, Hans Sandberg and Mikkel Toudal are sources of information for the results, in the fields of User Experience Design and Agile methodologies respectively. Their answers provide more context by taking into consideration not only developers but also other aspects of a development team.

Subsequently, the findings are evaluated by using the information extracted in the literature review and analyzing the answers to the questions. There are several matches and discrepancies between theory and practices that are further explained in the section Practices of the Industry and in the conclusions. The empirical section of the thesis aims to demonstrate the practical application of the research on a development team and also explore the possibility of agile teams to include Quality Management standards in their processes.

Sources and keywords

LUT University library, online databases (Scopus, IEEE and ACM) and the LUT Finna system are the main sources for information collection. For the SLR, the thesis utilizes articles, books, and papers, focusing primarily in publications not older than 2010. However, some relevant theories regarding Software Engineering, Agile Methodologies and Quality Management were published over two decades ago and are not excluded from this research based on publication date.

Some of the main keywords for information searching are: Quality management, Mobile application development, Agile methodologies, Software project management, Scrum methodology, Software Quality, Quality Assurance.

1.4 Organization of the study

The first part of the thesis introduces the topic by explaining the motivation behind it. It provides an initial description of the main components such as Quality Management, Agile methodologies and Mobile development. In addition, the research questions are stated and the scope and limitations that shape the research are defined. The research methodology details are also explained, including the sources and keywords used for information searching and the people involved in the empirical part.

The second section of the thesis explores the literature regarding Quality Management in the software industry. The main concepts of quality are explained and then a comprehensive summary of two of the most widely used Quality Management standards is given (ISO 9001 and CMMI).

In the third chapter, the topic of Mobile Application Development is analyzed. First, the mobile industry characteristics are explained in order to provide a comprehensive background. Then,

the research illustrates some aspects of traditional methodologies contrasted with agile methodologies. The last section of the chapter is dedicated to cohesion and knowledge sharing between mobile development teams.

Section four of the thesis is dedicated to presenting and analyzing six main practices identified as a result of the interviews to the industry experts. This section is based on the interview questions that are elaborated according to the theory extracted in the previous section.

Chapter five of the document presents the conclusions and discussions of the research. Presenting also the topics for further research possibilities. Figure 1 shows the organization of the research in a linear way.

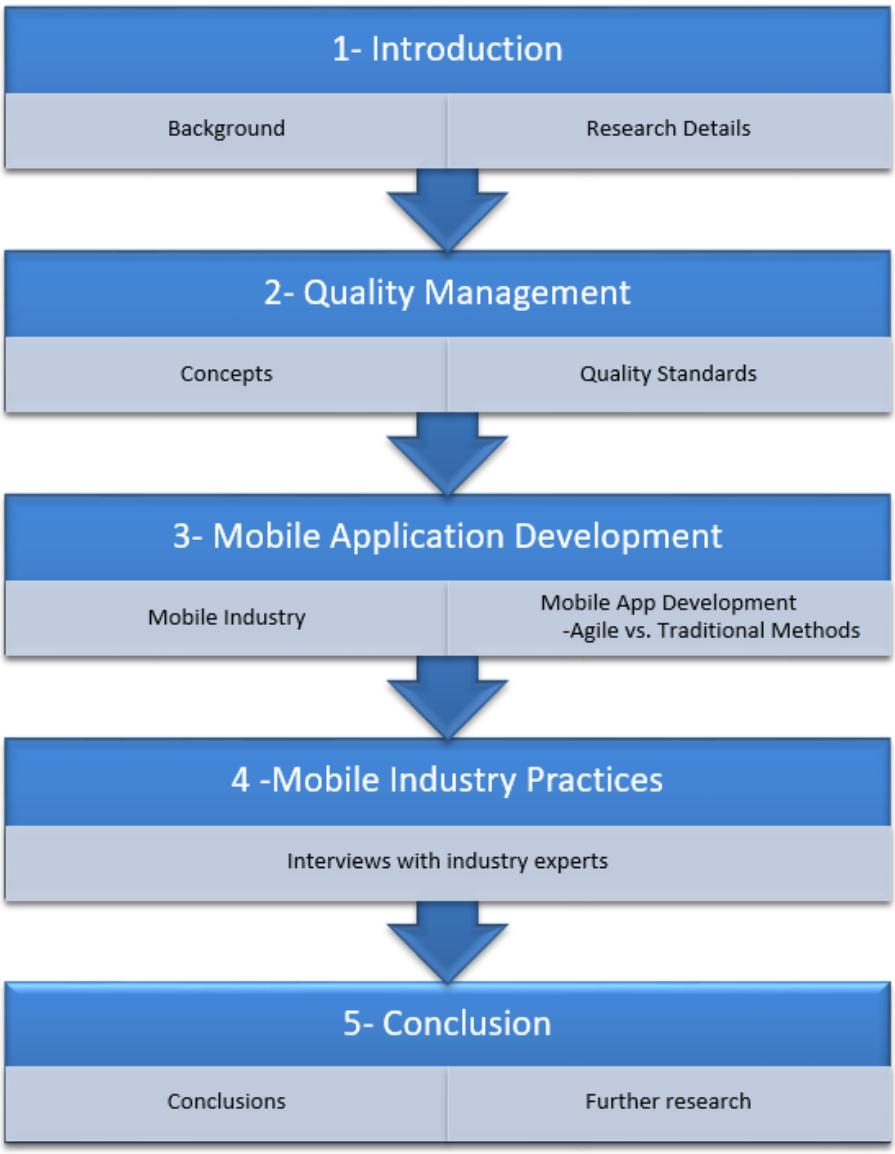

Figure 1. Organization of the study.

2. Quality Management in Software Development

Software Quality Management (SQM) is a set of processes that ensure that software products and services fulfill the quality objectives that have been set by an organization and achieve customer satisfaction (Bourque & Fairley, 2014). According to Sommerville (2004), SQM can be applied in two different levels. At the organizational level, it establishes a framework of processes and standards that will result in high quality software. At a project level, it ensures that project deliverables fulfill the quality standards applicable to the project. Consequently, SQM includes two high-level activities (Poth & Sunyaev, 2014):

1. Delivering a product with the expected quality (product/project level).
2. Making appropriate choices regarding activities and processes (organizational level).

Regarding the first activity, the Institute of Electrical and Electronics Engineers (IEEE) defines Software Quality as "the degree to which a software product meets established requirements". As stated by Bourque & Fairley (2014) the main goal of engineered products is to deliver maximum value to the stakeholders while managing budget and time constraints. And requirements are a way of expressing stakeholder value.

The second high-level activity refers to SQM as a set of processes integrated into the software development lifecycle. In order to have high-quality products, one must improve the quality of the software developing processes (Ijaz, Asghar & Ahsan, 2016). In this context, SQM relates directly to Software Process Improvement (SPI) and the Quality Life Cycle (QLC). One of the main goals of SQM is to have high-quality developing processes to produce high-quality end products.

Software testing is part of SQM but the two concepts should not be used interchangeably. SQM includes many processes that elevate the quality of a project and an organization. As stated by Tinnaluri (2016), quality cannot be added to a product in a specific stage of the project, quality is a result of many activities performed during the project's life cycle.

According to Inukollu et al. (2014), users who download a “bad quality” app will uninstall it after a few minutes and leave a negative review. Therefore, when developing a mobile app, quality must be perceived by the customer in the product. In a similar way, Tinnaluri (2016) mentions that deliverables must be “fit for use”, referring to the value that is delivered to the stakeholders. As with any other software product, high-quality processes must be in place when developing mobile applications.

A Quality Management System (QMS) is a management system put in place by an organization in order to accomplish its mission and objectives; There is only one system in an organization that includes all types of processes and procedures and their interactions, including quality (Hoyle, 2001). Standards like ISO 9001:2000 explain requirements for establishing, documenting, executing, supporting, and improving the processes of the QMS (Mutafelija & Stromberg, 2003).

2.1 SQM in the Software Development Lifecycle

Software Quality Management consists of several processes that are followed by a team to produce “fit for purpose” deliverables (Tinnaluri, 2016). Therefore, SQM can be seen as a cycle of activities that is followed along the development process. Software development and SQM are directly associated and outputs of SQM activities can be used as inputs for some development activities.

To identify the phases on a SQM cycle, one must consider four subcategories (Tinnaluri, 2016 and Bourque & Fairley, 2014):

1. Software Quality Planning: includes the identification of critical procedures and quality standards that will be used in the project. Also, it involves defining the quality objectives

and estimating the effort and time that will be needed in order to fulfill the quality activities.

2. Software Quality Assurance (SQA): defines and evaluates the adequacy of organizational processes to establish confidence that the project will produce software that fulfils the quality standards.

3. Software Quality Control (SQC): evaluates intermediate project artifacts and final products to determine if they are in adherence to the initial plan established (related to requirements, constraints, design). Also, it involves controlling that the processes and standards are implemented by the development team.

4. Software Process Improvement (SPI): this subcategory can also be merged with the previous ones. It consists on improving the efficiency and effectiveness of the current processes so that software quality is improved.

2.2 Quality standards and frameworks

Standards are an important part of quality assurance. An organization must define and select the standards that should be applied to products and processes (Sommerville, 2004). Ijaz et al. (2016) argue that when defined procedures and policies are in place, the human effort is reduced. Software Process Improvement is a good tool to achieve a well-defined and high-quality set of standards and processes. This processes and standards are not specific only to quality matters, they are related to the whole organization and its development cycle. Then, as part of the SQM effort, the established procedures will be monitored to guarantee that they have been followed.

According to Sommerville (2004), standards are important because of three main reasons:

1. They capture knowledge that is important for an organization.
2. They help define what "quality" means in a more objective way.
3. Standards support continuous operation since all engineers follow the same practices; Therefore, the organization will not be greatly disrupted if one person must continue work initiated by another.

"The purpose of most standards is to help its users achieve excellence by following the processes and activities adopted by the most successful enterprises" (Mutafelija & Stromberg, 2003 p.1). There are currently in the market some international set of standards that are used for developing quality management in organizations (Sommerville, 2004). Two of the most commonly used are ISO 9001 and CMMI which can be applied as a framework for the purpose to enhance software quality and reduce development costs (Agrawal & Chari, 2007).

2.2.1 Software Quality: definition and principles

Quality can be seen as a subjective field; However, there are many ways to define and measure it. Garvin (1984) gathered information regarding five different ways of defining quality according to the perspective:

1. Transcendent approach: states that quality is something that can be recognized universally and cannot be defined in a precise way.
2. Product-based approach: viewing quality as something that can be measured precisely and relating quality to quantitative variables such as price or number of features.
3. User-based approach: states that quality is related to user's requirements and overall satisfaction of use.
4. Manufacturing based approach: viewed from the perspective of the supplier. Design and requirements are given high importance and any deviation from them result in quality affectations.
5. Value-based approach: defining quality based on price and costs.

The International Software Testing Qualifications Board (ISTQB) definition of software quality falls under the user-based approached; it is described as the degree to which a software product and its features fulfill the explicit or implicit requirements specified by customers or users (ISTQB, 2015). It can be deducted from the definition that software quality is not something that can be added to a product in a specific phase of development, but rather a result of following a quality process throughout the development life cycle. Therefore, Quality is a field that must be managed efficiently.

According to the experts of ISO/TC 176, Quality management principles are "*a set of fundamental beliefs, norms, rules and values that are accepted as true and can be used as a basis for quality management."* (ISO, 2015 p.3). As displayed in Figure 2, the principles are interrelated to each other and an organization should aim to fulfill all of them and not just a specific set. The ISO 9001 standard is based on the eight Quality Management Principles, which are described as follows:

1. Customer Focus:

Customers are defined as any person or organization that receives a product or a service (ISO, 2015); an organization should focus on understanding the current and future customer needs to provide a service or product that aims to exceed customer expectations [(Hoyle, 2001).

2. Leadership:

Leaders are needed in every level of an organization. They provide vision and a unified direction so that people can come together to reach the organization's objectives. A strong leadership helps an organization to reach their desired objectives, instead of just going with the flow and adjusting along the way. For that, it is also vital to establish effective channels of communication with people (Mutafelija & Stromberg, 2003 & Hoyle, 2001).

3. Involvement of people:

An organization will benefit greatly from having fully involved staff at every level. This can be achieved by motivating and recognizing people for their contributions. Management should be open and encourage their staff to not hold back when discussing important matters whether is coming up with improvements or new ideas for the organization. (Mutafelija & Stromberg, 2003 & Hoyle, 2001).

4. Process approach:

The principle states that when the development activities are perceived as interrelated processes that are part of a system, the results are more efficient and effectively achieved (ISO, 2015). As part of Quality Management, those processes must be fully understood and planned to improve them. If a process is not providing any added value or its purpose is not understood, it must be changed or removed (Mutafelija & Stromberg, 2003).

5. System approach to management:

An organization should be managed as a system of interconnected and interacting processes. By taking this approach, each process and each interaction can be analyzed and corrected if the results are not efficient. The analysis also involves setting optimized roles and responsibilities associated with each process (Mutafelija & Stromberg, 2003).

6. Continual improvement:

The principle dictates that organizations should constantly look for ways to improve their performance by doing periodical evaluations and setting improvement objectives across the organization levels (Hoyle, 2001). Continual improvement involves regular assessments of the whole organization and its processes and evaluating the results against the objectives (Mutafelija & Stromberg, 2003).

7. Factual approach to decision making:

Organizations should have certain procedures to measure their processes accurately and reliably. Then, the data from those measurements can be analyzed and used to make decisions. Better results are achieved when important decisions are based on data and information (ISO, 2015).

8. Mutually beneficial supplier relationships:

The final principle states that efficient management of an organization's relationship with suppliers (being mutually beneficial) will enhance the organization's ability to create value (ISO, 2015). According to Hoyle (2001), organizations do not exist to generate profit. However, profit is needed to grow the organization's customer base and achieve high level of customer satisfaction. Therefore, profit and beneficial supplier relationships can be viewed as major tools to add value to an organization.

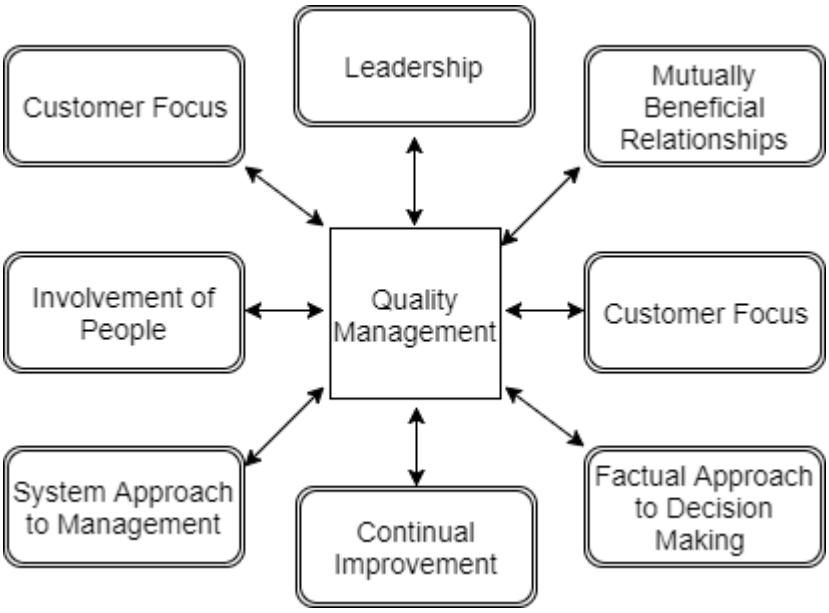

Figure 2 Eight QM principles interrelated and QM.

2.2.2 ISO 9001

ISO 9001 is one of the five documents that constitute ISO 9000, which is the most widely used quality management standard (Mutafelija & Stromberg, 2003). ISO 9001 can be applied to software; it is a framework for developing standards for quality management systems and focuses on organizations that design, develop, and support products (Sommerville, 2004). Organizations can be more efficient and improve their customer satisfaction by exploring the application of ISO 9001 (ISO 9001, 2015).

ISO 9001 defines quality principles and processes in general and establishes which standards and procedures should be identified by the organization (Sommerville, 2004). Once the processes are identified, quality management is accomplished by appropriate managing of them (Mutafelija & Stromberg, 2003). An organization that wants to be ISO 9001 compliant must document how their activities are related to the 9 core processes established by ISO, which are displayed in Figure 3.

Sommerville (2004) explains that the ISO 9001 standard is supposed to be general, it does not recommend any specific quality processes to be used by an organization; companies can

accommodate those processes to their needs and still follow the ISO standard regardless of their type, size or the product that they offer.

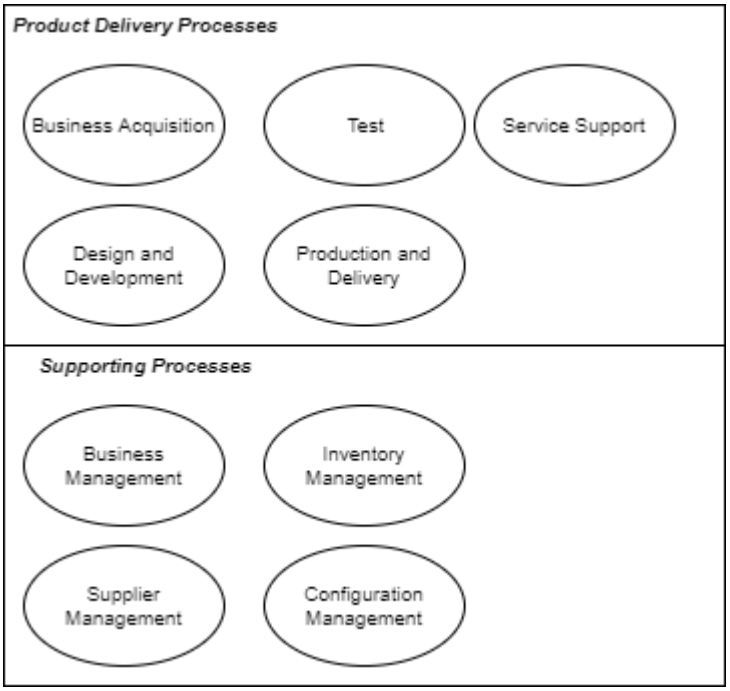

Figure 3 Nine core processes of ISO (Sommerville, 2004).

2.2.2.1 ISO 9001 General Requirements

To provide a better understanding of the ISO 9001 standard, the following are the general requirements for a quality management system established by ISO 9001 and are included in the IEEE's *Guidelines for the Application of ISO 9001:2008 to Computer Software*:

The organization shall:

1. Establish the processes needed for the QMS and their application in every level of the organization. IEEE mentions that processes for software development, operation or maintenance should also be identified.
2. Determine the order and interactions between those processes. IEEE adds that an organization should identify the sequence of processes inside their defined software development life cycle and in their quality planning, which also should be aligned to a life cycle model.

3. Identify criteria and methods that can ensure an effective operation and control of those processes.
4. Provide resources and information needed to monitor the processes and support the whole operation.
5. Analyze, monitor and measure the processes.
6. Implement measures to achieve the initially planned objectives and set the way for continual improvement of the processes.

2.2.3 Software Capability Maturity Model (CMMI)

In the 1980s the U.S. Software Engineering Institute (SEI) had the goal of finding a way to evaluate the capabilities of software contractor (Sommerville, 2004). As a result, in 1991 was published the Software Capability Maturity Model (CMM) version 1.0. Mutafelija and Stromberg (2003) define it as a framework describing fundamental aspects of efficient software processes. CMM is based on efficient practices of software development, it not a conceptual model based on theory (Mutafelija and Stromberg, 2003). After that, many models (CMMs) were created for different industries such as systems engineering, software acquisition, workforce management and development (Carnegie Mellon, S.E.I., 2006).

CMMI stands for CMM Integration and it was created to solve the problem of applying several CMMs (Carnegie Mellon, S.E.I., 2006). It consists on a set of best practices which will lead to performance improvements if followed by organizations (CMMI Institute, 2017). CMMI is a framework for process improvement that can be applied by many organizations on myriad industries. Sommerville (2004) notes that the CMMI staged version and the Software CMM are compatible and can be used to evaluate an organization.

Sommerville (2004) summarizes the model by highlighting its three main components:

1. Process Areas: CMMI defines 22 process areas which are a set of interrelated practices put in place to fulfill a common goal. Those process areas can be grouped into 4 main categories (Process and Project Management, Engineering, Support).

2. Goals: a description of desirable states that an organization should reach. CMMI includes generic and specific goals. Generic are applied to all process areas and specifics are related to a given process area.

3. Good Practices: CMMI includes a set of helpful actions to achieve a goal, those practices are also categorized as generic and specific. However, CMMI allows freedom of approach, recognizing that achieving the goal is the important part of the model.

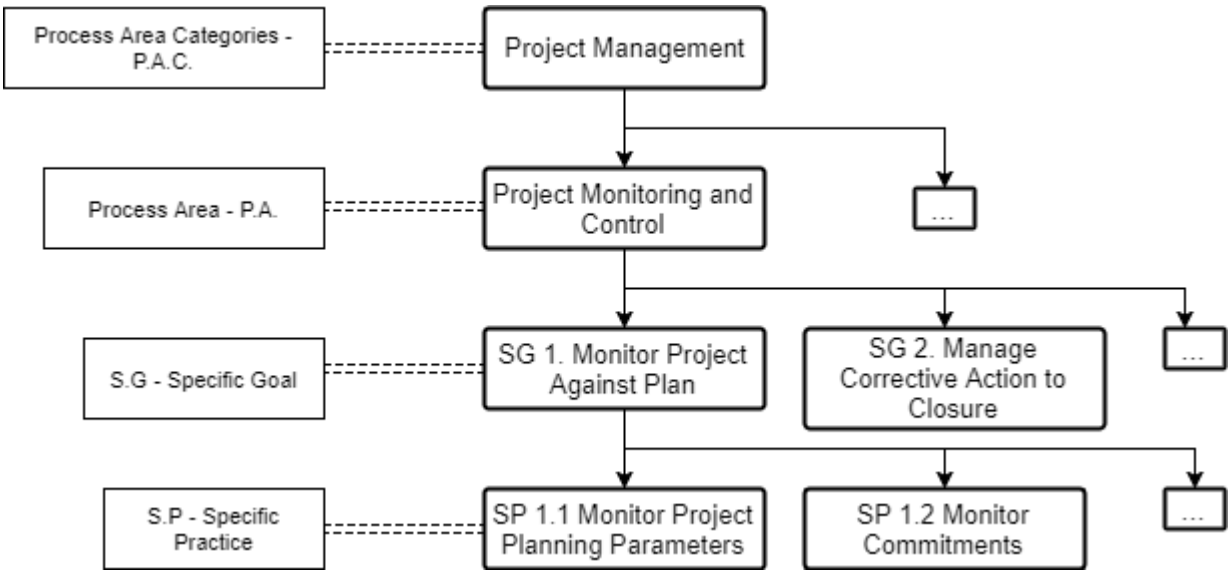

Figure 4 Main components of the CMMI model. (Summerville, 2004)

CMMI is a staged model that examines processes of an organization and rates them according to the level of process maturity; understanding that a process is better if it is more mature (Mutafelija and Stromberg, 2003; Summerville, 2004). The levels of maturity are defined as follows:

1. Initial: software processes are described as ad hoc and disorganized. The organization does not have many defined processes and it relies on high capacity individuals to succeed (Mutafelija and Stromberg, 2003). Maturity level 1 organizations are capable to produce positive results, although frequently exceed the initial budgets or schedules (Carnegie Mellon, S.E.I., 2006).

2. Managed: In this level, the organizations have established basic project management processes which are executed according to the policies. There are practices for controlling costs, budget and functions. This allows an organization to repeat previous success on similar areas and to keep adherence to their practices during high pressure situations (Carnegie Mellon, S.E.I., 2006).

3. Defined: The organization defines a standard software process of its own which includes the activities for management and engineering that have been documented and standardized. All projects use a variation of the organization's standard software process for developing and maintaining software that has been approved (Mutafelija and Stromberg, 2003).

4. Quantitatively Defined: The main distinction of level 4 is that an organization has established quantitative goals for performance and quality. Detailed metrics are gathered and utilized for managing the processes and project quality. Also, the control of process performance switches from qualitative to quantitative (Carnegie Mellon, S.E.I., 2006).

5. Optimizing: The final level is characterized by continuous process improvement. The quantitative data gathered from the processes is used to improve constantly (Mutafelija and Stromberg, 2003). The organization is concerned with the causes of process variation and there is room for innovative processes and changing processes to improve performance (Carnegie Mellon, S.E.I., 2006).

CMMI provides to organizations a viable path to improve from maturity level 1 to 5. To do so, an organization must focus on the process areas associated to the maturity level that it wishes to

reach and achieve all the goals in them (Carnegie Mellon, S.E.I., 2006). Asides from the level of maturity, CMMI establishes 4 categories:

Project Management: for project related practices.

Engineering: related to development and product.

Process Management: related to processes inside the organization.

Support: related to resolutions and process analysis.

Table 1 shows the Key Process Areas (KPA) grouped by maturity level and category. Note that there are no process areas in maturity level 1 given that there are no defined processes.

Table 1 Key process Areas (Carnegie Mellon, S.E.I., 2006).

Maturity Level	Key Process Area	Category
2	Configuration Management	Support
	Measurement and Analysis	Support
	Project Monitoring and Control	Project Management
	Project Planning	Project Management
	Process and Product Quality Assurance	Support
	Requirements Management	Engineering
	Supplier Agreement Management	Project Management
	Decision Analysis and Resolution	Support
	Integrated Project Management +IPPD	Project Management

3	Organizational Process Definition +IPPD	Process Management
	Organization Process Focus	Process Management
	Organizational Training	Process Management
	Product Integration	Engineering
	Requirements Development	Engineering
	Risk Management	Project Management
	Technical Solution	Engineering
	Validation	Engineering
	Verification	Engineering
4	Organizational Process Performance	Process Management
	Quantitative Project Management	Project Management
5	Causal Analysis and Resolution	Support
	Organizational Innovation and Deployment	Process Management

3. Mobile Software Development

'Mobile devices' is a term used for any computer device that can be handheld, usually it refers to smartphones, tablets, e-readers and portable music players (Viswanathan, 2017). The main characteristics of a mobile device are displayed in Figure 5.

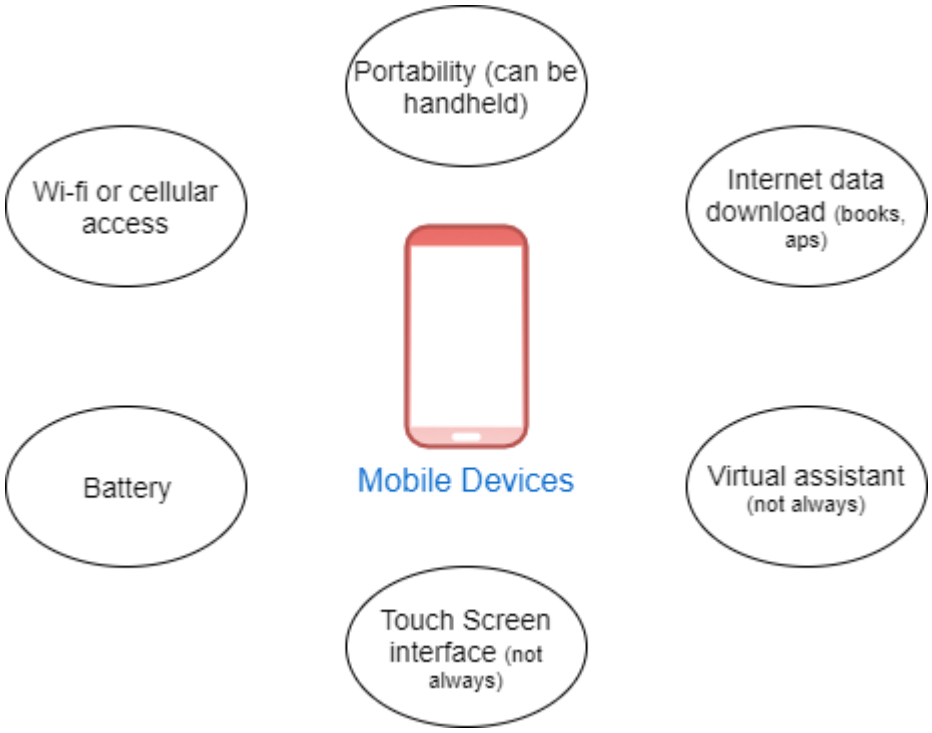

Figure 5 Characteristics of mobile devices (Viswanathan, 2017)

The mobile device industry has seen a significant growth in the last decade. According to Boren (2017) there are now 7.2 billion devices in the world, which means the number of people and devices are close to being equal. However, he points out that the number of devices is growing at a rate 5 times faster than people.

Smartphones and other mobile devices are becoming the primary tool for online interactions. As reported by Chaffey (2017), mobile devices are now more widely used than desktop computers; People in United States (71%), United Kingdom (61%), Mexico (75%) and Indonesia (91%) spend most of their time online in a mobile device. In addition, Danova (2017) reports that 86% of the time spent in mobile devices is dedicated to applications, with Gaming (32%) and Facebook (17%) being the most widely used.

The incremental growth in the mobile device market has created an opportunity for many software companies (Joorabchi, Mesbah and Kruchten, 2013). This part of the thesis explores the main characteristics of mobile development and examines several methodologies and practices used to develop applications for the mobile industry.

3.1 Characteristics of mobile development

Organizations in the mobile development industry face some specific characteristics that differ from standard software development. There are challenging market and technical characteristics that must be taken into consideration by management. Some of the market characteristics include:

1. **Short time to market:** high number of competitors create a need to launch products as early as possible; therefore, creating a challenge for the development process (Corral, Sillitti, and Succi, 2013).

2. **Fast paced industry:** a constant change in requirements and emerging trends in technology create a volatile industry that provokes some complications in development (Corral, Sillitti, and Succi, 2013).

3. **Early feedback:** automation testing is not common in the mobile market and occasionally the developers do their own tests (Joorabchi, Mesbah and Kruchten, 2013);

Therefore, early feedback from customers is highly valued and can cause major changes in the final product.

In addition to the market challenges, mobile development organizations also encounter technical difficulties due to the characteristics of the industry:

1. **Fragmentation:** Joorabchi et al. (2013) explain that developers perceive as a challenge the high level of fragmentation in the mobile industry, referring to the different platforms available (such as iOS and Android) and different characteristics (CPU, memory, etc.) of the devices in the same platform.

2. **Automation testing:** there are several obstacles to implementing automation testing in mobile development, making testing dependent mostly on manual testing. Some of the challenges are linked to the platforms itself and others to the lack of appropriate tools like a robust simulation environment (Joorabchi et al., 2013).

3. **Data intensive apps:** dealing with data transfers in mobile applications can be an arduous job, given that the devices have limited storage and network connections to data sources are challenging (Joorabchi et al., 2013).

4. **Limited capabilities of devices:** mobile devices have limited capabilities in battery, storage and connectivity. In addition, there are devices and operating systems inside a platform that have different specifications (Corral et al., 2013).

Due to the characteristics explained above, it is considered that traditional methodologies for software development are not suitable for mobile development (Baker, 2014). Corral et al. (2013) argue that agile methodologies are more appropriate for mobile development because factors such as changes and innovations are facilitated by lightweight processes of development. This chapter will explore the characteristics of both types of methodologies.

3.1.1 Traditional development methodologies

Traditional methods of development are focused primarily on planning; they are considered by some as bureaucratic and slow paced, given their aim at predictable and efficient development (Fowler, 2005).

Traditional methodologies also have the characteristic that people inside a team is seen as a resource with specifically defined roles (Fowler, 2005), this helps with planning and makes it easier to transfer or include people in the team given any special circumstance.

To understand what is considered "traditional methodologies" of software development the thesis will explain the methods of Waterfall and Incremental model.

3.1.1.1 Waterfall methodology

The Waterfall model, also referred to as cascade model, is a linear software development lifecycle (SDLC) introduced in 1956 and modified in 1970 (Ruparelia, 2010). There are five different stages in the model and they should be executed in a sequential form, with the output of one phase usually working as an input for the next one (Huo et al., 2004).

The main characteristic of the model is that it focuses in planning and definition before starting the design and development of software (Huo et al., 2004). The five cycle stages are displayed in Figure 6.

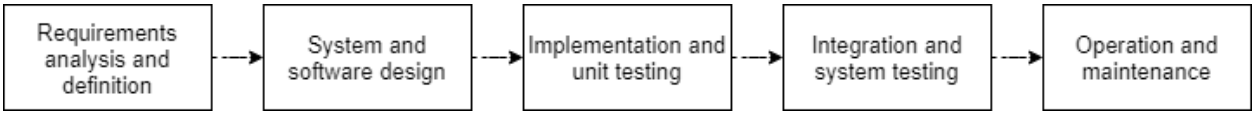

Figure 6 Waterfall cycle stages (Sommerville, 2004).

Stoica, Mircea and Ghilic-Micu (2013) find more disadvantages than advantages when using the Waterfall model. Some of the advantages and disadvantages are:

Advantages:

1. New members of the team benefit from the documentation.
2. The model is simple and easy to understand.
3. It is easy to coordinate due to the sequential approach of the model.

Disadvantages:

1. New requirements emerge after the definition phase is finished.
2. The model does not allow the project to be partitioned into stages.
3. Difficult to estimate budget and schedule for the stages.
4. It is hard to return to the design stage if testing throws negative findings.

The waterfall model is mostly suitable for short projects where the requirements are defined and understood by the team and the product description is not expected to undergo any major changes during the development (Stoica, Mircea and Ghilic-Micu, 2013).

Formal methodologies such as Waterfall have been used in the industry for many decades and they provide a clear cycle that can be followed from stage to stage. Huo et al. (2004) identified the Waterfall cycle phases and the SQM supporting processes that must be followed along. Both set of activities are presented in Figure 7.

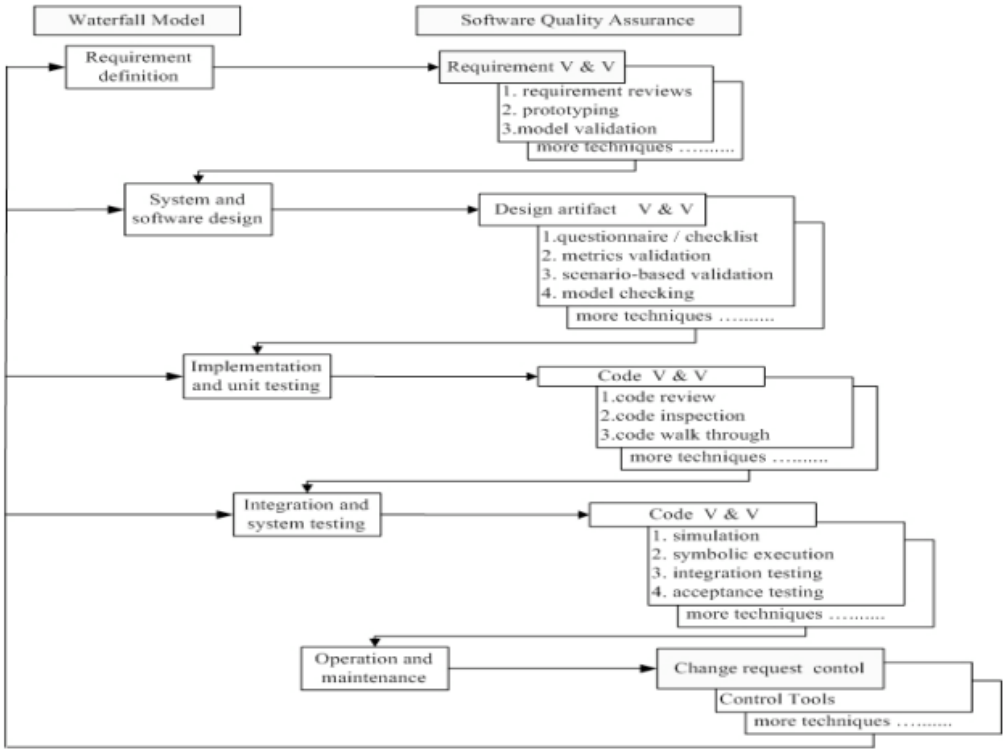

Figure 7 Waterfall model and supporting quality processes. (Huo et al., 2004, p.4)

3.1.1.2 Incremental model

The incremental model is characterized by dividing the requirements in smaller segments. The modules are developed in multiple cycles and each of them go through requirement analysis, design, implementation and testing (Stoica et al., 2013). The idea is that every module adds features to the previous one, finalizing with the completion of the system. The model (displayed in Figure 8) is sometimes viewed as a segmented variation of the waterfall model.

The main advantages and disadvantages of the model are (Stoica et al., 2013):

Advantages:

1. Each module produces a working section of the product, which can be presented to clients.
2. Prototypes are easy to deliver.

3. New requirements can be added to the next iteration.

Disadvantages:

1. The system definition must be complete in order to be divided into segments in an efficient way.
2. The model can become a way of constantly fixing and repairing previous code.
3. Design errors can be hard to fix.
4. The client may be tempted to add more functionalities to next modules.

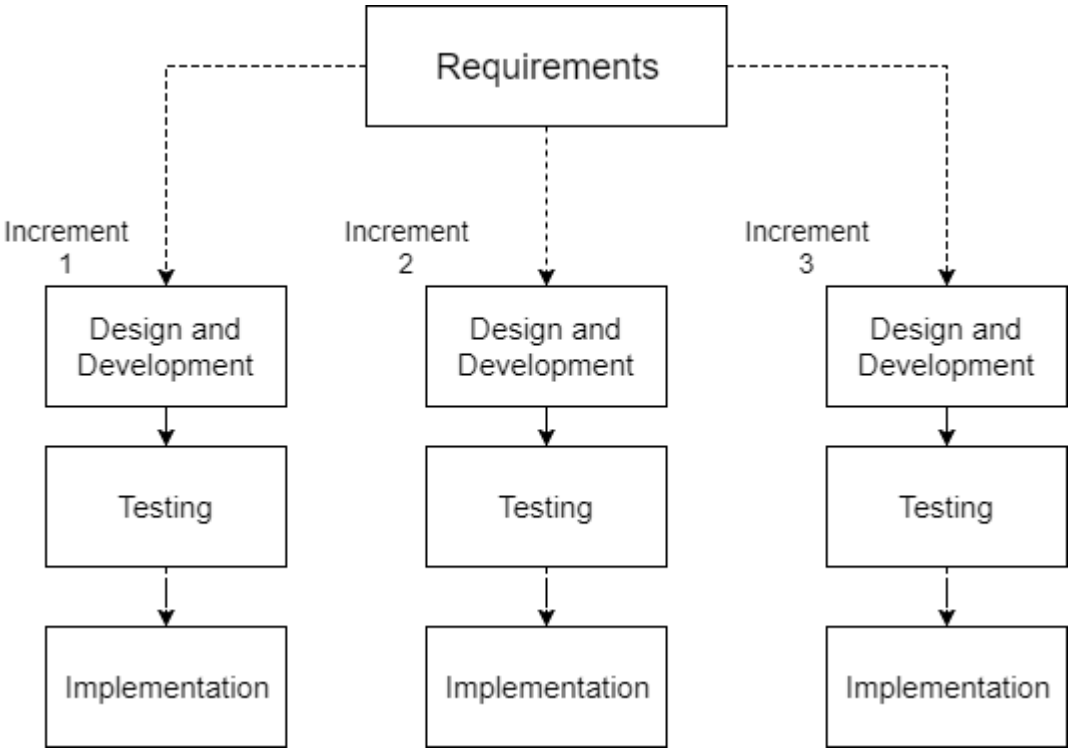

Figure 8 Incremental model diagram (Stoica et al., 2013).

3.1.2 Agile methodologies

Agile methodologies originated in the 1990s when a group of software developers proposed a new approach to software engineering, much less focused on design than traditional heavyweight models, which allows a rapid and efficient response to changes (Harb, Noteboom, and Sarnikar, 2015). Agile methods are simple and aim to deliver software in a short period of time by prioritizing the functions and collecting feedback from several sources (Abrahamsson, Salo, Ronkainen & Warsta, 2002).

When using Agile methods, the software specification, development and delivery are based on incremental cycles that deliver working software constantly (Sommerville, 2004). Working in an agile environment involves constant communication and an overall integration of the team (Stoica et al., 2013). Agile teams also aim to eliminate factors in the software process that are considered bureaucratic or inefficient (Sommerville, 2004).

One of the key elements of Agile methods is the Agile Manifesto published in 2001 which defines 12 principles. Figure 9 displays some of the main ideas of the manifesto. Beck et al, (2001) defined four focal points based on the principles:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

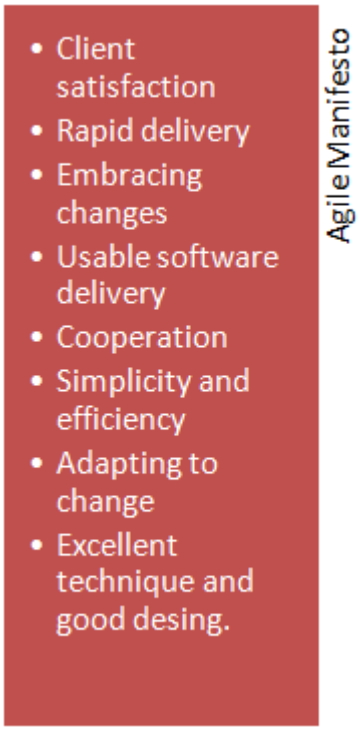

Figure 9 Main ideas of the Agile Manifesto (Stoica et al., 2013).

Extreme Programming (XP)

Extreme Programming (XP) is a software development method that aims at improving the project quality and the ability of a team to respond to requirement changes (Nawrocki et al., 2002) (Qureshi, 2012). To achieve those goals, teams work on short development cycles and are self-organizing and cross-functional (Shore & Warden, 2008).

Extreme programming has the following stages in its development process (Abrahamsson et al., 2002):

1. Exploration: customers select which story cards they want to include in the release (story cards describe features of the system). Additionally, the team gets acquainted with the tools and technology that will be used for the project and a prototype is built

2. Planning: in this phase the schedule is agreed upon by prioritizing story cards and developers estimate their efforts. Planning phase takes a couple of days only.

3. Iterations to release: in this phase the initial schedule is broken down in several iterations. There are several short iterations (one to four weeks) before the first release. The customer usually decides what goes in each iteration; However, the first iteration should define the architecture for the system to be built upon.

4. Productionizing phase: testing and performance reviews of the system take place on this phase. Requirement changes can emerge and new features or fixes can be included in the current release or postponed for later.

5. Maintenance phase: maintenance phase requires to keep the system in production while also adding new features through iterations. Development speed may drop on this phase and new people can be added to the team.

6. Death phase: death phase occurs when the customer has no more story cards to be included in releases or if the project will not be continued. Documentation is written in the death phase since the established system will not experience any major changes.

Extreme programming collects ideas and practices of already existing methodologies. One of its main characteristic is the list of best practices (Beck, 2000):

- Planning game: There is a close relationship between customers and software developers.

- Small releases: Keep releases short, at least once every month a new version should be available.

- Metaphor: The system is designed based on metaphors about how the system should work.

- Simple design: Aim to design the simplest solution possible, eliminate unnecessary complexities.

- Testing: Software development is test driven. Customer writes functional tests.

- Refactoring: Restructure the system by simplifying it and making it more flexible.

- Continuous integration: New pieces of code are added continuously. Tests must be passed in order to accept the new code.

- 40-hour week: The team works 40 hours a week.

- On-site customer: Customer or representative must be available full time for the team.

- Coding standard: Programmers must agree on a standard to be defined and used.

Abrahamsson et al. (2002) points out that Extreme programming is usually followed in a partial way, by implementing only a set of the practices described. Additionally, he states that XP is mostly recommended for small or medium sized teams.

Scrum

Scrum is an agile methodology based on the iterative and incremental development cycles, it consists in implementing customer requirements (in form of User Stories) by working in small cycles called Sprints, which usually last from 2 to 4 weeks (Schwaber, 1995). By using a process such as Scrum, teams can improve their capacity to deal with changes and deliver high quality software faster (Keenan, 2004). Figure 10 shows the Scrum development process.

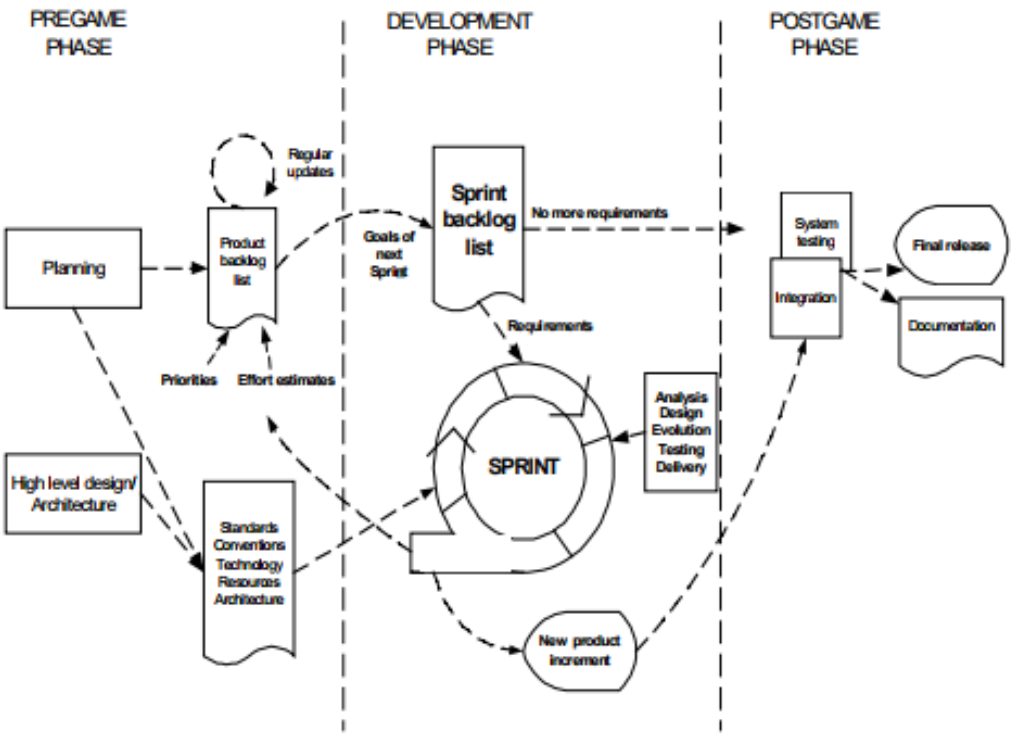

Figure 10 Scrum process (Abrahamson et al., 2002 - p.28).

There are several variables (technical and managerial) involved in software development. The main idea behind Scrum is that those variables are likely to change. Therefore, the development processes are complex and uncertain, which requires teams to be flexible and adapt to make corrections. (Abrahamson et al., 2002).

Scrum is a development method recommended for teams of few people. It focuses on the individuals and aims to deliver software constantly in short spans of work. Scrum promotes working as a unit and sharing responsibilities and tasks between members (Keenan, 2004).

Table 2 Scrum items (Abrahamsson et al., 2002; Fernandes and Almeida, 2010).

Name	Type	Description
Product Owner	Role	Officially responsible for the project and in charge of tasks associated to the Product Backlog.
Scrum Master	Role	In charge of making sure the project follows the scrum framework with its rules and best practices. Additionally, ensures that the team can work as efficiently as possible, removing roadblocks and clarifying issues.
Scrum Team	Role	Group of people responsible for achieving the goals of every sprint.
Customer	Role	Client needs to be fully involved in the decision-making tasks.
Product Backlog	Artifact	Complete list of required features of the system.
Effort estimation	Event	Process of assigning an estimation to the items in the product backlog.
Sprint	Event	Short increments (2 to 4 weeks) in which a sub-list of requirements is developed.
Daily meeting	Event	Meetings held every day to check progress.
Sprint Planning	Event	Meeting at the beginning of the sprint to prepare and evaluate the items to be included.
Sprint Backlog	Artifact	Sub-set of tasks from the product backlog that will be included during the sprint.

Sprint Review	Event	Review meeting at the end of the sprint.
Sprint Retrospective	Event	Meeting to analyze issues of the past sprint.
Burn Down Chart	Artifact	Graphic representation of the work pending on a sprint.

Mobile-D Approach

Mobile-D is an example of how Agile methodologies can be adapted to mobile development. It was created by a group of professionals in Finland, basing the development practices, scalability options and life-cycle coverage on Extreme Programming, Crystal methodologies and Rational Unified Process respectively (Abrahamsson et al., 2004).

Mobile-D is recommended for teams of less than 10 developers who are working in a shared space given that quick and easy communication is vital. Mobile-D aims to deliver fully functional mobile applications in less than 3 months. The methodology was created with cooperation from companies and was assessed in level 2 of the CMMI framework. (Abrahamsson et al., 2004).

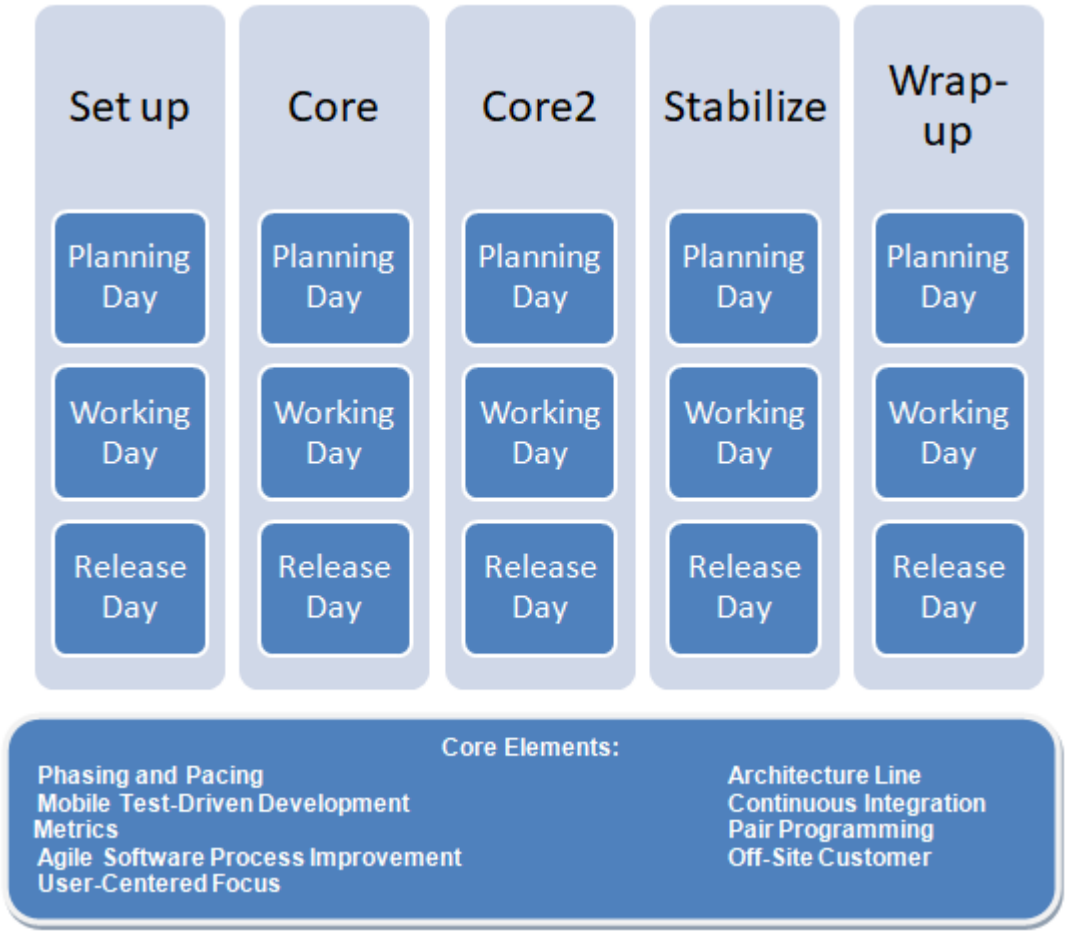

Figure 11 Mobile-D Development (Abrahamsson et al., 2004).

Figure 11 displays the organization details of the Mobile-D approach. There are five iterations, each consisting in three types of days (planning, working and release) and an additional type of day (Integration Day) is needed if there are concurrent teams developing a system. The practices of the Mobile-D approach consist on 9 different core elements that are commonly used in the agile environment. However, Architecture Line is a specific practice of Mobile-D and it consists on knowledge sharing across an organization by collecting patterns and solutions utilized in similar situations inside or outside the organization. (Abrahamsson et al., 2004)

Table 3 Comparison of development methods

Methods	Main Characteristics	Recommended for
Waterfall	Extensive Planning Formalization of roles Sequential phases	- Requirements are clear and understood. - Subject experts are available. - Teams are familiar with the technology.
Incremental	Development Iterations Prototyping	- System definition is complete. - Early launch needed. - New technology.
Extreme Programming	Short cycles Continuous integration Customer and teams work closely	- Customer unsure of requirements. - Constant changes.
Scrum	Flexibility Short iterations Communication Team cohesion Early feedback	- Small teams. - Constant change. - Involved customers. - Frequent demonstrations.
Mobile-D Approach	Short time delivery Communication Iterations	- Teams of less than 10 people. - Mobile Application Development. - Co-located teams.

3.2 Mobile Development Teams

Considering the special characteristics and challenges that the mobile software industry presents (mentioned in Section 3), companies are adopting agile methodologies and variations of them instead of traditional methods of development. As mentioned before, Agile switches the focus from planning and documenting to communications and individuals; This has proven advantageous in an industry that requires a lot of exploration and flexibility. (Jeon, Lee & Shin, 2008)

So far, the literature has clearly presented a contrast between the main characteristics of traditional development and the needs of the mobile software development industry. The sequentiality of the phases and the lack of adaptation to changes being some of the main issues for mobile teams (Abrahamsson et al.,2002). Agile methodologies, on the contrary, offer an environment that can adapt and is suitable for small teams that want to develop software in an exploratory and efficient way. However, mobile teams should adopt agile in a way that allow them to deliver high-quality software and that could be achieved by following some of the quality management principles.

Team Cohesion

The development methodology that a team follows can have an impact on the team cohesion, making it easier or harder to achieve. Cohesion refers to the degree of attraction of group members (Shaw, 1981). In a cohesive group, members participate actively in activities and help each other to fulfill the group's goals (Henry, Arrow, and Carini, 1999).

In traditional software development, there is a hierarchical approach to tasks; Managers assign work and set deadlines and processes that must be followed (Melnik and Maurer, 2006). On the contrary, Agile teams are usually self-organized and possess some degree of autonomy (Moe, Dingsøyr, and Dybå, 2009). The Agile Manifesto states in one of the values that individuals and their interactions must have high priority within an organization. Agile teams develop work

through participation of the members, reorganizing and assigning work when needed. Also, there are shared tasks and a freedom to choose the approach needed to solve them (Nerur, Mahapatra, and Mangalraj, 2005).

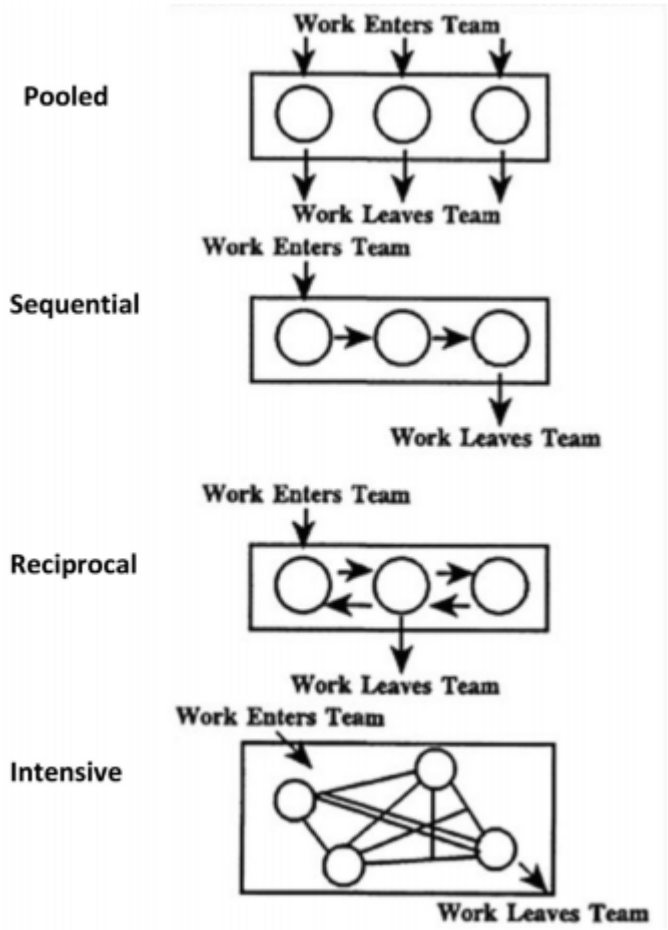

Figure 12 Typologies of interdependence (Kakar, 2018).

Agile methodologies give more responsibility to members of a team. When members perceive that the project depends on their actions, they tend to participate more and strive to achieve quality. Figure 12 illustrates the typologies of interdependence on a team. In an agile environment, tasks should follow the intensive typology.

Cram, W. and Marabelli, M (2017) mention that knowledge sharing differs substantially between teams who adopt agile methodologies and teams who work using traditional methods of development. Knowledge can be shared using an "object" approach, which can be exchanged

from person to person by documentation; or can be viewed as “relationship” knowledge, which is transferred through interactions of team members while working. These two types of approaches can be associated to the agile and traditional software development methodologies (Alavi and Leidner, 2001).

4. Practices of the industry

In order to evaluate the literature and establish correlations between the theory and the practice, a series of interviews were conducted to professionals in the mobile development industry. Two mobile developers, one user-experience (UX) designer and an agile coach collaborated with their knowledge on this research project. The results of those interviews are classified in six main areas that are described in this section.

4.1 Flexible Agile Environment

The research finds agile development to be the most utilized methodology by teams creating mobile applications. However, there is not a unique, specific methodology used across teams. Given the flexibility of the methodologies, organizations use a variation of Scrum or Kanban that is adjusted to their needs.

Bruno Sarmiento is an iOS developer involved in outsourcing development and tailored software solutions. He mentions that agile methodologies are by far the best option given how adaptable they are and the fluidity of interactions needed when developing a mobile application. In his experience, when developing mobile applications, there is a constant process of discovering and learning. And agile methodologies such as Scrum provide the perfect framework for the uncertainty and adaptability needed from both client and development team.

Mikkel Toudal is an agile coach with over 10 years of experience in diverse industries. He points out that the agile manifesto and its principles must be used only as a guide for software development. He comments that, when working in an agile environment, teams must adapt and define processes that make sense to them and their specific factors. That is why, in his opinion, the current agile methodologies should cover the needs of the mobile industry.

4.2 UX Driven Development

Sarmiento mentions that a main element of mobile development is the use of a set of illustrations created by the User Experience (UX) designer to demonstrate to the client how the system will look. That prototype is approved by the client and then transferred to a Product Owner and the rest of the team, which divides it into user stories and requirements. The prototype is taken as a basis during the whole development process.

This type of development is characterized by being very agile and dynamic; it focuses on the user's perspective on the system, which results in a better experience for them. In his opinion, Scrum facilitates this type of development and two-week sprints are particularly helpful for analyzing current results. However, there are two major disadvantages that must be managed:

1. UX designers may lack technical skills, which causes their designs to be hard to achieve given the development framework and the time constraints. To overcome this, he proposes to involve developers and testers as early in the process as possible, so that there is a technical base.

2. The development team must guide their coding mainly by the prototype, without specific functionalities or requirements involved. This may cause that other implicit quality features are overlooked. A way to mitigate this risk is to have a separate list of non-functional requirements that the system must meet.

Hans Sandberg, who works as a UX designer in the development industry, considers that the two disadvantages presented above can be avoided by including developers, the client and the product owner in the creation of the prototype. Resulting in an agile and collaborative process of defining the product.

Sandberg mentions that even though the prototype is considered a point of reference for the product, it is constantly adjusted by meetings and interactions between the team during development.

4.3 Flexibility and constant feedback

Constant feedback is also a characteristic of agile methodologies that is present in the mobile development industry. Sarmiento mentions that in his projects clients evaluate progress regularly, which provides a direct channel of communications and feedback. Teams perceive this situation as highly helpful as it allows them to correct any misunderstandings quickly, resulting in fewer surprises for the client in the end.

Jason Gamboa is an Android developer with over 5 years of experience for multinational companies. He argues that even though having a direct channel of communication is beneficial, it must be regulated with a clear scope and expectations. A defined scope limits the client from making extreme changes that vary too much from the original agreement. He mentions that given the flexibility provided by Scrum, clients sometimes can change their mind in the middle of the development phase, and ask for a dramatic change in the project's objectives.

The adaptability of the agile methodologies should not be misinterpreted as freedom to introduce major changes in the final product, he says, as these might have an impact on the deadline and the price negotiated initially. Therefore, there is a need of managing expectations and other management factors such as budget and schedule.

4.4 Efficient development

The experts also mention that given the small size of some teams and the urgency of the project, the client expects the development to be as efficient as possible. Therefore, meetings for process establishment are sometimes deemed not relevant by managers and clients. When developing software for small clients or with small teams, quality standards such as ISO 9001 are seen as a

complex, time-consuming task which will not impact the final product to a degree that justifies the effort necessary to adhere to them.

Gamboa, who is currently working for an outsourcing software company, considers efficient development to be a major requirement for clients. He mentions that Agile methodologies contribute to achieve efficiency by encouraging interactions and constant problem solving. In his opinion, methodologies such as Scrum are suitable frameworks for mobile development. However, he mentions how faulty implementations of those methods may cause problems. For example, in an effort to reduce development times, a client might ask to exclude unit testing from development; which is perceived by him as a violation of the coding best practices.

For Gamboa, the concept of efficiency is closely linked to quality. He argues that to produce a high-quality product the team should work efficiently as there are always budget and schedule constraints. If there is no efficiency in development, those constraints will end up hurting the product quality.

4.5 Two different platforms

According to the experts, another characteristic in the industry is the presence of two teams of development when creating mobile applications. The market is dominated by Android and iOS; and developing for each platform involves radically different characteristics. In the teams interviewed, Android teams use Android Studio as an Integrated Development Environment (IDE) and develop software using Java programming language. iOS teams use Xcode and code in the Swift language. Sandberg mentions that Xamarin is a platform that provides an option for teams to develop mobile applications in both platforms at the same time. However, all the interviewees agreed that most of their projects have been developed natively both in iOS and Android.

Aside from the highly different framework of development, there are substantial differences when designing user interfaces for the two platforms. There are general styling differences that create the need for separate designs, even though the mobile applications are expected to

perform equally. Sandberg mentions that is possible to utilize the same designs for both platforms, but the client must be informed that there will be visual differences in the result. Gamboa and Sarmiento mention that the differences between platforms are not restricted to visuals. They each work in a different platform, Gamboa develops for Android apps and Sarmiento for iOS. They comment that, even though the same applications perform similar functionalities in both platforms, they are built entirely different in the back-end.

The disparity of platforms and their development environments create a management problem because the two teams are expected to progress at similar speed. In some cases, the client might expect both applications to be delivered at the same time, since essentially the functionality is identical. However, there are many peculiar circumstances around development that make it challenging to keep the same pace in both teams, and the client must be informed about that fact. Both Sarmiento and Gamboa agree that Android development is usually more time consuming and takes longer than iOS development. Table 3 shows differences between the two platforms mentioned in the research.

Table 4 Differences between Android and iOS.

Characteristic	Android	iOS
Programming language	Java	Swift
IDE	Android Studio	XCode
Development time	Longer periods	Quicker development
Store payment	One time pay	Yearly
Review period	Quick review	Longer review
Monetization strategy	In-app advertisement	Premium and payed apps

4.6 Quality in the organizations

Regarding what quality means for their organization, the interviewees answers resemble Garvin's (1984) user-based definition exposed in section 2 of the literature review. Quality in mobile industry is closely linked to user's perception and ease of use; And the development process is guided by requirements and expectations from the client. Additionally, Gamboa considers that the code should follow standards about maintainability, naming conventions and design patterns to be considered as high quality.

In the mobile industry, opinions and experiences of final users are highly valued. Sarmiento explains that in a competitive market, where users have many options available and uninstalling an application is a simple process, first impressions are important and any defect that disrupts the user experience must be fixed expeditiously.

Even though the companies of the interviewees do not follow any specific QA standards, they follow an agile development process. Companies define their objectives, in which they include quality metrics and requirements, and then shape their development process in order to achieve those goals according to the schedule set.

Both developers acknowledge that standards such as CMMI and ISO are not taken into consideration by small companies or projects. These standards are regarded by them as nice-to-know theory but a formal implementation of them is not in line with the scope and dimension of the projects they are involved. However, once an organization adheres to the standard, they consider necessary to comply with the guidelines, in the interest of following the processes defined organizationally.

Gamboa states that quality standards are important for customer attraction. In his experience, however, it is easier to certify people instead of a whole organization. And certified workers have also a great impact in attracting customers as they confirm the high level of professionalism and skills in the team.

The experts mentioned 6 key factors that they consider to be related to quality in software and mobile applications. They are explained below:

1. Minimal errors: the perception of quality is related to the amount of errors that people encounters while using the application. If there are a small number of errors in an application, the experts consider it to be a sign a good quality.

2. Requirement fulfilment: if an application fulfills all the requirements that were initially planned, it can be considered to have at least some degree of quality.

3. Amount of people using the application: a high number of downloads and active users of an application can be an indication of high quality according to the experts.

4. Non-functional requirements are under normal ranges: even though a mobile app covers the initial requirements and functions without encountering errors, there are still other factors such as loading times, privacy settings, data usage and visual aesthetics that can affect quality and the user experience. According to the experts, these other aspects should be under reasonable ranges to consider an application to be high quality.

5. Quality of the team: the concept of product quality was linked by the interviewees to the quality of the individuals inside the development team. In their opinion, individuals who are certified in coding, testing or other fields are more capable of delivering a quality product.

6. Well-structured and documented code: the experts discuss how a quality application must have good coding practices behind it. And the observance of those practices can be considered an aspect of quality. This is an example of how quality is not only linked to user experience but can also be related to back-end practices that are not noticeable for users.

Figure 13 shows the main ideas that Sarmiento and Gamboa link to the quality of a product in software.

Figure 13 Ideas linked to high quality apps by developers.

5. Conclusions

Quality management provides a set of theoretical practices and standards that help teams achieve quality not only on the final software product, but also along the whole development process. Organizations on the mobile industry can apply quality management frameworks in their processes and generate high-quality products. However, for smaller teams, those processes are not formally defined. Teams with limited budgets and scopes do not consider adhering formally to a standard because of the complex task it presents.

There are several methods of developing software with high quality processes. Standards such as ISO and CMMI provide only a set of practices that have been backed up by theory and have been tested by other teams. However, a team can produce great software by many other ways. Some organizations rely heavily on skilled individuals and agile development to achieve quality. Considering that agile teams produce results in short spans, it is possible to iterate many times until a satisfactory degree of quality is achieved. In some organizations, this makes more sense than establishing an entire framework.

Quality is a critical factor in the mobile industry. In mobile applications it is closely linked to user perception and satisfaction. The competitive market also drives the user's expectations up and makes it vital for teams to present a stable product from the first release. For example, searching for a calculator app in the Android store displays hundreds of results. Not only the application has to catch the eye of users but also keep them satisfied with the functionality.

Traditional methodologies for software development have been used for many years and continue to be applied by many organizations. They have proved to be useful and teams have achieved quality products during many years by their application. However, agile development has taken over the software industry during the last decades. The mobile industry is no exception and the theory and results of this research point towards agile development being the chosen methodology by mobile development teams.

Agile methods provide a better alignment with the needs of mobile teams than traditional methods. However, there is not a specific agile method tailored for the industry, at least not one that is being widely used. Because of the flexibility that the agile framework allows, each team has its own implementation for their specific purposes. Dealing with time constraints and customer involvement, mobile teams rely on agile methods in order to work more efficiently and produce software.

Methodologies such as Scrum aim to guide the processes of a team to provide better results and a more efficient development. Quality Management standards introduce practices and processes with the intention to achieve quality. Therefore, Scrum can be aligned with QM by adjusting the Scrum processes with the standards as a base.

A clear example of compatibility between Agile and QM is the main role given to individuals. One of the principles of Quality Management mentioned in section 2 is the involvement of people. That principle can be directly associated with the agile principle mentioned in the manifesto (Beck et al, 2001) of *Focusing on individuals and interactions*. Customer focus and continual improvement are other principles of QM that are described in section 2 that can also be associated to agile, considering that the manifesto mentions the importance of *Customer Collaboration* and *Responding to Change.*

The mobile industry presents some specific challenges when building software. Those challenges are not commonly present in other types of development and they increase the level of complexity in the delivery of a high-quality product. There are time constraints and a very competitive industry that depends in customer satisfaction. Also, there is a need for two development teams if the application is to be launched in the two main platforms. All these factors affect the software development process and they need to be taken into consideration when defining methods of working and standards.

The topics covered in the research aim to provide an answer to the initial question: Can software development teams work in an agile environment and follow the Quality Management standards while overcoming the challenges presented by the mobile industry? Based on the findings, it

can be inferred that mobile teams are able to deliver high quality software in the industry and agile methodologies are the predominant method of development. However, adhering to standards of quality such as ISO 9001 and CMMI is not widely practiced. For reasons of efficiency, scope or budget, mobile development teams are not used to adjusting their processes in conformance of a quality management framework.

Quality is not a characteristic that can be provided or added to a product in a single phase or stage in development. In order to achieve a high-quality product, there should be a development environment that embrace the concept of quality and applies it along the way. QM standards show how to do that by providing guidelines and practices that apply to every role and phase of development. Agile methodologies have not been formally linked to QM standards and there is a lack of theory that combines both topics. However, agile methods incorporates quality in its values and objectives.

Agile methodologies are suitable for Quality Management standards as they share basic principles and both concepts aim to introduce quality in several aspects of the development process. Based on the research, it can be interpreted that mobile teams can comply with quality standards without drastically changing their methods of working, supported by the flexibility that agile methods provide. However, the development teams interviewed believe that quality can be introduced in different ways and they are confident in their current methods.

Future Research

Further research based on this project could be oriented towards two directions. First, creating an agile method that incorporates quality management and its associated processes in a more integrated way. This research presents the correlation between the agile framework and the quality management standards. Therefore, combining both concepts can result beneficial for many teams because it will create an agile way of development that also ensures a high-quality delivery. Next steps need to include the application of that framework in a practical way (using a development team) and evaluating the results.

Other topic to explore further is how development teams achieve quality in their products. In this project it becomes clear how quality can be managed subjectively, with individuals defining it according to their perception. By identifying which points, processes or roles provide quality to a final product, it will be possible to potentiate them and make the development cycle more efficient.

6. References

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P. and Salo, O., 2004, October. Mobile-D: an agile approach for mobile application development. In Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (pp. 174-175). ACM.

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J., 2002. Agile software development methods: Review and analysis.

Agrawal, M. and Chari, K., 2007. Software effort, quality, and cycle time: A study of CMM level 5 projects. IEEE Transactions on software engineering, 33(3).

Alavi, M. and Leidner, D.E., 2001. Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS quarterly, pp.107-136.

Alsabi, E. (2016). A Methodology Engineering For Mobile Application Development. Masters Thesis. Prince Sultan University.

Baker, V. (2014). Gartner Says Traditional Development Practices Will Fail for Mobile Apps. [online] Gartner.com. Available at: http://www.gartner.com/newsroom/id/2823619.

Beck, K., 2000. Extreme programming explained: embrace change. addison-wesley professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J., 2001. Manifesto for agile software development.

Boren, Z. (2017). There are officially more mobile devices than people in the world. [online] The Independent. Available at : http://www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html [Accessed 26 Oct. 2017].

Bourque, P. and Fairley, R.E., 2014. Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.

Carnegie Mellon, S.E.I., 2006. CMMI for Development, Version 1.2 (pp. 29-45). CMU/SEI-2006-TR-008.

Chaffey, D. (2017). Mobile Marketing Statistics compilation. [online] Smart Insights. Available at: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/ [Accessed 26 Oct. 2017].

CMMI Institute, (2017). What Is Capability Maturity Model Integration (CMMI)®? | CMMI Institute. [online] Cmmiinstitute.com. Available at: http://cmmiinstitute.com/capability-maturity-model-integration [Accessed 23 Oct. 2017].

Corral, L., Sillitti, A. and Succi, G., 2013, May. Software development processes for mobile systems: Is agile really taking over the business?. In Engineering of Mobile-Enabled Systems (MOBS), 2013 1st International Workshop on the (pp. 19-24). IEEE.

Cram, W.A. and Marabelli, M., 2017. Have your cake and eat it too? Simultaneously pursuing the knowledge-sharing benefits of agile and traditional development approaches. Information & Management.

Danova, T. (2017). THE FUTURE OF MOBILE [SLIDE DECK]. [online] Business Insider. Available at: http://www.businessinsider.com/the-future-of-the-mobile-industry-2014-11?r=US&IR=T&IR=T [Accessed 26 Oct. 2017].

Fernandes, J.M. and Almeida, M., 2010, September. Classification and comparison of agile methods. In Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the (pp. 391-396). IEEE.

Fowler, M. (2017). The New Methodology. [online] martinfowler.com. Available at: http://martinfowler.com/articles/newMethodology.html [Accessed 5 Oct. 2017].

Gartner (2014). Gartner Says Traditional Development Practices Will Fail for Mobile Apps. [online] Available at: http://www.gartner.com/newsroom/id/ 2823619 [Accessed 5 Oct. 2017].

Harb, Y., Noteboom, C. and Sarnikar, S., 2015. Evaluating Project Characteristics for Selecting the Best-fit Agile Software Development Methodology: A Teaching Case. Journal of the Midwest Association for Information Systems, (1), p.33.

Henry, K.B., Arrow, H. and Carini, B., 1999. A tripartite model of group identification: Theory and measurement. Small Group Research, 30(5), pp.558-581.

Hoyle, D., 2006. ISO 9000 quality systems handbook. Elsevier.

Huo, M., Verner, J., Zhu, L. and Babar, M.A., 2004, September. Software quality and agile methods. In Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International (pp. 520-525). IEEE.

IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.

IEEE Std. 12207-2008 (a.k.a. ISO/IEC 12207:2008) Standard for Systems and Software Engineering—Software Life Cycle Processes, IEEE, 2008.

Ijaz, Q., Asghar, H. and Ahsan, A., 2016, August. Exploratory study to investigate the correlation and contrast between ISO 9001 and CMMI framework: Context of software quality management. In Innovative Computing Technology (INTECH), 2016 Sixth International Conference on (pp. 388-391). IEEE.

Inukollu, V.N., Keshamoni, D.D., Kang, T. and Inukollu, M., 2014. Factors influencing quality of mobile apps: Role of mobile app development life cycle. arXiv preprint arXiv:1410.4537.

ISO 9000:2005 Quality Management Systems—Fundamentals and Vocabulary, ISO, 2005.

ISO 9001. (2015). Moving from ISO 9001:2008 to ISO 9001:2015. [online] Available at: https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/iso_9001_-_moving_from_2008_to_2015.pdf [Accessed 12 Oct. 2017].

ISO, I.S.O., 2011. IEC25010: 2011 Systems and software engineering–Systems and software Quality Requirements and Evaluation (SQuaRE)–System and software quality models. International Organization for Standardization, 34, p.2910.

ISO. (2015). Quality management principles. [online] Available at: https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/qmp_2012.pdf [Accessed 10 Oct. 2017].

ISTQB, I., 2015. Glossary of Testing Terms. ISTQB Glossary http://www. istqb. org/downloads/finish/20/193. html.

Joorabchi, M.E., Mesbah, A. and Kruchten, P., 2013, October. Real challenges in mobile app development. In Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on (pp. 15-24). IEEE.

K.Flora, H., V. Chande, S. and Wang, X. (2014). Adopting an Agile Approach for the Development of Mobile Applications. International Journal of Computer Applications, 94(17), pp.43-50.

Kakar, A. (2018). Engendering cohesive software development teams: Should we focus on interdependence or autonomy?. International Journal of Human-Computer Studies, 111, pp.1-11.

Krylova, S. and Leuchter, S., 2017. From ISO 9001: 2008 to ISO 9001: 2015–Challenges for Software Engineering in Small and Medium Sized Enterprises. Projektmanagement und Vorgehensmodelle 2017-Die Spannung zwischen dem Prozess und den Mensch im Projekt.

McMichael, B. and Lombardi, M., 2007, August. ISO 9001 and Agile development. In Agile Conference (AGILE), 2007 (pp. 262-265). IEEE.

Melnik, G. and Maurer, F., 2006, June. Comparative analysis of job satisfaction in agile and non-agile software development teams. In International Conference on Extreme Programming and Agile Processes in Software Engineering (pp. 32-42). Springer, Berlin, Heidelberg.

Moe, N.B., Dingsøyr, T. and Dybå, T., 2009. Overcoming barriers to self-management in software teams. IEEE software, 26(6).

Mutafelija, B. and Stromberg, H., 2003. Systematic process improvement using ISO 9001: 2000 and CMMI. Artech House.

N Inukollu, V., Keshamon, D., Kang, T. and Inukollu, M. (2014). Factors Influencing Quality of Mobile Apps: Role of Mobile App Development Life Cycle. International Journal of Software Engineering & Applications, 5(5), pp.15-34.

Nerur, S., Mahapatra, R. and Mangalaraj, G., 2005. Challenges of migrating to agile methodologies. Communications of the ACM, 48(5), pp.72-78.

Oberscheven, F.M., 2013. Software Quality Assessment in an Agile Environment (Doctoral dissertation, Master's thesis. Radboud University Nijmegen).

Poth, A. and Sunyaev, A., 2014. Effective quality management: value-and risk-based software quality management. IEEE Software, 31(6), pp.79-85.

Ruparelia, N.B., 2010. Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, 35(3), pp.8-13.

Shaw, M.E., 1971. Group dynamics: The psychology of small group behavior.

Sommerville, I., 2004. Software Engineering. International computer science series. ed: Addison Wesley.

Stoica, M., Mircea, M. and Ghilic-Micu, B., 2013. Software development: Agile vs. traditional. Informatica Economica, 17(4), p.64.

Tinnaluri, V.S.N., 2016. An Approach of Software Quality Management. Imperial Journal of Interdisciplinary Research, 2(7).

Viswanathan, P. (2017). How many mobile devices you have now?. [online] Lifewire. Available at: https://www.lifewire.com/what-is-a-mobile-device-2373355 [Accessed 26 Oct. 2017].

7. Appendices

Appendix 1 Interview Questions

1. How is the organization aiming to fulfill (or exceed) the customer's expectations? Is agile helpful or harmful for achieving that? Why?

2. How involved are employees in the organization? Are there channels to communicate important issues, offer feedback or propose ideas?

3. Are the processes inside the company constantly followed and evaluated?

4. What kind of data does the company gather to make decisions regarding practices and processes?

5. Is there a map of development processes and their interactions?

6. How attached to established processes are the employees when there are stressful situations?

7. Describe the development cycle from requirements analysis to delivery/maintenance.

8. What does the organization understands by quality of software and how is it achieved?

9. Are there negatives aspects of using an agile approach for mobile development?

10. How important are QM Standards (such as ISO or CMMI) for the organization. Has it ever been discussed to get certified?