FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

An Agile and Lean Software Process Model for Mobile Application Development

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Lukas Wenzel
Matrikelnummer 0826915

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Thomas Grechenig
Mitwirkung: Raoul Vallon

Wien, 16. April 2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

An Agile and Lean Software Process Model for Mobile Application Development

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Business Informatics

by

Lukas Wenzel
Registration Number 0826915

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Thomas Grechenig
Assistance: Raoul Vallon

Vienna, April 16, 2014

(Signature of Author) (Signature of Advisor)

Statement by Author

Lukas Wenzel
Weinberggasse 28, 2100 Leobendorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

I hereby declare that I am the sole author of this thesis, that I have completely indicated all sources and help used, and that all parts of this work – including tables, maps and figures – if taken from other works or from the internet, whether copied literally or by sense, have been labelled including a citation of the source.

(Place, Date) (Signature of Author)

Acknowledgements

First and foremost I want to thank my parents, Helga and Wolfgang, for their ongoing financial and personal support during my studies. Without them this would not have been possible.

I also want to thank the employees of the INSO institute who helped to create this master thesis. In particular I want to thank Thomas Grechenig and Raoul Vallon for the scientific supervision and guidance in course of this master thesis.

Last but not least, I want to thank all people involved in the case study. Special thanks go to the experts who participated in the interviews and therefore provided valuable input for this research.

Kurzfassung

Die rasante Entwicklung von mobilen Technologien hat Geräte in diesem Gebiet von einem Luxusgegenstand zu einer Notwendigkeit in unserem Leben gemacht. Vor allem die Fähigkeit, jederzeit und überall online sein zu können, ist für die Mehrheit der Smartphone-Nutzer extrem wichtig geworden. Dies hat zur Folge, dass der Mobilfunkmarkt kontinuierlich wächst. Die Popularität von Smartphones und Tablets hat die Nachfrage von qualitativ hochwertiger Software in diesem Bereich stark erhöht. Um in diesem umkämpften Markt überleben zu können, wird der Einsatz von Softwareprozessmodellen empfohlen. In den letzten Jahren sind agile und lean Vorgehensweisen für die Entwicklung von Software sehr populär geworden, jedoch gibt es nur wenig Forschung in Bezug auf die Anwendbarkeit dieser Konzepte im Bereich von mobilen Applikationen. Ziel dieser Diplomarbeit ist es daher, die Auswirkungen von agilen und lean Methoden in diesem Gebiet der Softwareentwicklung zu untersuchen.

Um dies bewerkstelligen zu können, hat der Autor ein neues Prozessmodell, genannt ALP-Mobile (**A**gile and **L**ean **P**rocess for **Mobile** Application Development), entwickelt, welches eine Kombination aus Scrum, Kanban und Extreme Programming (XP) darstellt. Das eingeführte Konzept, bei dem auf fixe Iterationen in der Entwicklung verzichtet wird, erhöht die Flexibilität von ALP-Mobile und hilft darüber hinaus, sich an die Schnelllebigkeit des Mobilfunkmarktes anzupassen.

Im Rahmen einer Fallstudie, die im Zuge dieser Diplomarbeit durchgeführt wurde, konnte das entwickelte Prozessmodell evaluiert werden. Durch die Durchführung von qualitativen Interviews mit Experten in diesem Bereich ist der Autor in der Lage, Stärken und Schwächen von ALP-Mobile abzuleiten.

Das positive Feedback, welches im Zuge der Interviews erhalten wurde, ist ein Indikator dafür, dass ALP-Mobile einen Mehrwert für Organisationen, die sich mit der Entwicklung von mobilen Applikationen beschäftigen, liefert.

Schlüsselwörter

Agil, Lean, Software Prozesse, Entwicklung mobiler Applikationen

Abstract

With the recent advances in mobile technology, mobile devices have emerged from a luxury to a necessity in our lives. Especially the ability to be connected anytime and anywhere has become essential to a majority of smartphone users. As a consequence the mobile market is flourishing. The popularity of smartphones and tablets has increased the demand for high quality software in this area. In order to survive is this competitive market the use of software process models is advised. In recent years agile and lean methodologies have become very popular for the development of software, but there has only been little research about the applicability of these methods in the area of mobile applications. Therefore this master thesis aims to investigate the impact of agile and lean concepts in the field of mobile software development.

In order to achieve that, the author proposes a new process model called ALP-Mobile (**A**gile and **L**ean **P**rocess for **Mobile** Application Development) which is a combination of scrum, kanban and extreme programming (XP). The introduced concept, in which fixed iterations in the development are waived, increases the flexibility of ALP-Mobile and furthermore helps to adapt to the fast pace of the mobile market.

Practical experiences in the field of mobile application development are gathered in course of a case study executed within this research. Due to the conduct of qualitative interviews with experts in this area, the author is able to derive strengths and weaknesses of ALP-Mobile.

The positive feedback received during the interviews is a first indicator that ALP-Mobile can bring real value to organizations working in the area of mobile application development.

Keywords

Agile, Lean, Software Processes, Mobile Application Development

Contents

List of Figures

List of Tables

1 Introduction

This chapter introduces the reader to the topic of this master thesis. First of all the author describes the research problem in general. Subsequently, reasons for the selection of this specific topic are pointed out. After the formulation of the problem statement and the motivation, the author presents the aims of this thesis. Conclusively the structure of the thesis is displayed to the reader.

1.1 Problem Statement

In order to develop software successfully, the process of developing software needs to be standardized [1]. This can be accomplished through the use of software process models and the proper execution of software development methodologies.

This thesis will focus on agile and lean software development, which has become very popular in recent years [2, 3, 4]. The principles behind those two methodologies come from agile and lean manufacturing [5]. The term *agile* was used in the 1990s to refer to flexible production systems [5]. With the creation of the agile manifesto [6] in 2001 the principles of agile manufacturing have been adapted to software development [5]. It has few important values to keep in mind. Individuals and interactions are more important than processes and tools, working software is more valuable than comprehensive documentation, customer collaboration is preferred over contract negotiation and last but not least responding to change is better than strictly following a plan [6].

Lean manufacturing has its origins in the 1940s, when Toyota produced automobiles with roughly half the labor hours as automakers in the US [5]. The term *lean* was first introduced in the mid 1980s at MIT corresponding to production management processes and product development [5]. In the 1990s the idea of applying lean practices to software development arose and later on, it formed to seven principles of lean software development: 1) optimize the whole, 2) eliminate waste, 3) build quality in, 4) learn constantly, 5) deliver fast, 6) engage everyone, and 7) keep getting better [5]. Using agile methodologies can help software developers to lower costs, increase productivity as well as quality and create better business satisfaction [2].

Due to the ongoing globalization organizations around the world started to distribute their work to create better software in faster time [7]. Especially in agile software development distributed working can be a great opportunity, since development teams in an agile environment are usually small sized [8] and therefore the coordination overhead is not as big as in large sized teams. With all the advantages which come from such a distributed environment, there are also complications which arise from this situation, e.g. deteriorated communication. To ensure the quality of the products created via distributed software development organizations need methodologies to face the challenges in *global software development* [9]. Since global software development is not a part of this research, distributed working will only be discussed in a small scope.

The popularity of agile and lean methodologies in software development in general has made those approaches a valid topic for research. This thesis will therefore investigate the specific impact of agile and lean methods in the area of mobile software development and furthermore answer the following research question:

How can agile and lean processes be adapted to mobile app software development?

1.2 Motivation

In 2012 the amount of active smartphone users has topped one billion for the first time ever [10]. This huge number is only a small indicator for the rapid growth the mobile phone market is currently undergoing. Experts say that this number is likely to double within the next three years [10]. But not only the mobile phone market is expanding very fast at the moment. Tablets have become the new must have gadget. Only in the 4th quarter of 2012 more than 50 million tablets were sold worldwide [11]. With the high quantity of Internet-connected mobile devices the demand for suitable software is becoming bigger and bigger.

According to Canalys Google's Play Store and Apple's App Store both already provide more than 800.000 apps [12]. But it is not only Google and Apple who invest a lot in the potential of the mobile phone market. Especially Windows Phone from Microsoft is getting more and more popular. With an estimated market share of 10.2% in 2017, Windows Phone will solidify its position as the number three operating system behind Android (68.3%) and iOS (17.9%) [13]. Looking at the number of apps, Windows Phone is already targeting the big players in the business. With more than 145.000 apps, the Windows Phone Store still has not reached the amount of Google's Play Store or Apple's App Store, but it shows the direction Microsoft is heading [12]. From a software developing point of view those numbers, combined with the future outlook about the further growth of the mobile market, show the importance of mobile application development in the future.

The financial potential of the mobile market is also remarkable. An ABI Research report estimated that the mobile app market will be worth $27 billion in 2013 [14]. At the moment there is still a huge gap between smartphone app revenues and tablet app revenues. Nevertheless the ABI Research report predicts that tablet app revenues will overcome the smartphones by 2017 [14]. It is reasoned with the fact that people are more likely to spend money for apps on tablets compared to smartphones, because of the bigger screen size. Regardless of whether smartphones or tablets will create more revenue in the future, those predictions show that the mobile market is a very profitable business sector right now and will even be more important in the future. For the software development industry this means that more and more organizations will focus on this specific branch of industry, which will lead to a very competitive market. Software development methodologies and process models can help to survive in such markets.

1.3 Aims

The goal of this master thesis is to give insight into agile and lean software development specific to the context of mobile applications. In detail this means that the challenges in this field of software engineering will be discussed and analyzed. Possible solutions for those challenges will be presented and recommendations will be given. Especially the research question, how agile and lean processes are adaptable to mobile app software development, will be answered. In order to achieve this an agile and lean software process model will be created, which then later on gets used to evaluate the applicability of those techniques in this field of software development.

The increase of popularity of agile and lean methodologies has brought significant change to software development [15]. Over the last one or two decades there was a shift from traditional software engineering to agile methods like *extreme programming (XP)* and *scrum*. As a part of this thesis the differences between traditional software engineering and agile software development will be presented. By comparing those two methodologies the strengths and weaknesses of both approaches will be discussed.

Although agile and lean software development is widely used nowadays, there has been little research about the applicability of agile and lean specific in the area of mobile applications. By working out the characteristics of mobile apps the effectiveness of those two software development methodologies in this area can be illustrated. Furthermore the differences between software development in general and mobile application development will be displayed and described in detail.

Due to the rapid expansion of the mobile phone and tablet market, the technologies used in this area are very widespread. Basically there are two different approaches in the development of mobile applications. On the one hand there are native applications which run on the mobile device and can therefore interact very well with the hardware of the specific device. On the other hand there are cross-platform solutions which run independently from the operating system of the mobile device [16]. In the course of this thesis the pros and cons of both approaches will be displayed. Since none of those approaches change the setting of the software development process, they do not conflict either with agile or lean software development methodologies, but through the examination of these techniques the reader gets a better understanding of mobile application development in general.

Another aspect, which will be discussed in this thesis, is distributed working. Especially in the field of software development it is often the case that the development team is distributed over several sites [17]. The problems and challenges which arise from this situation will be explained and solutions respectively guidelines for dealing with this situation will be given.

1.4 Related Work

This section of the thesis presents related work of this research. As already mentioned, agile and lean methods have become very popular in recent years, but there has only been little research about the applicability of these methods in the area of mobile application development.

The first approach to apply agile thinking to mobile application development was proposed by Abrahamsson et al. in 2004. In [18] they introduced an agile development approach called *Mobile-D* which tries to overcome the challenges in mobile app development. Four years later Jeong et al. presented in [19] the *MASAM* (Mobile Application Software Development based on Agile Methodology) methodology which is very similar to Mobile-D. Also in 2008, Rahimian and Ramsin based in [20] their research on the combination of plan-based and agile methodologies for the production of mobile software systems. The outcome was the *Hybrid Methodology Design Process*. Another two years later in 2010, Scharff and Verma analyzed in [21] the use of scrum for the development of mobile applications in a scholar setting. In 2011, Cunha et al. presented in [22] *Scrum Lean Six Sigma (SLeSS)* which is an integration approach of scrum and lean six sigma. They claimed that this approach "enables the achievement of performance and quality targets, progressively improving the development process and the outcome of projects" [22, p. 283].

Corral et al. focus in [23] their research on the agile development models for mobile application development displayed within this section. They conclude that mobile software engineering still faces an extensive work load to determine which processes and practices are the best fit for this field of software development.

In section 6.3 the author compares the above presented approaches to the agile and lean software process model proposed within this research. Additionally to this comparison the current state of the art regarding the topics of this research is presented in section 3.6.

1.5 Structure of the Thesis

The thesis is divided into four main phases: 1) theoretical research, 2) creation of an agile and lean software process model for mobile application development, 3) practical experiences in the area of mobile app development, 4) discussion. Following each phase will be discussed in detail.

- Phase 1 - Theoretical Research
  In the beginning a literature research of relevant topics in the area of agile and lean software development, mobile application development and its processes is performed. After the research, the reader gets introduced into the main concepts of those areas. This includes a theoretical introduction to agile and lean software development in general. By comparing these approaches to traditional software engineering, strength and weaknesses of each methodology get pointed out and discussed.

  After the introduction to software process models, the thesis focuses on the topic of mobile application development. By explaining the different approaches and techniques which can be executed when dealing with software development in the area of mobile applications, the reader gets introduced into this specific field of software engineering. Furthermore the differences between mobile app and software development in general will be presented in this part of the thesis.

- Phase 2 - An Agile and Lean Software Process Model for Mobile App Development
  The main theoretical part of this thesis is the investigation of the applicability of agile and lean software development methodologies in the field of mobile application development. Hereby a comparison of characteristics of mobile applications and agile methods is performed. Difficulties which arise from differences in the characteristics get examined and solutions for those problems will be given. As a result of this comparison and the theoretical foundation gathered through the literature research an agile and lean process model for mobile application development is created.

- Phase 3 - Practical Experiences in the Area of Mobile Application Development
  Besides the theoretical part of the thesis there will also be a practical part in which the knowledge of the research is used in practice. In detail, an exploratory case study is conducted in which experiences of experts in the field of mobile application development are investigated. Interviews with different roles in this field of software development are the main source of information for the case study. The obtained information from the case study help to derive the strengths and weaknesses of the process model created in the previous phase of the thesis. Additionally to the execution of the interviews the author conducts a pilot interview which helps to integrate real life experiences into the created process model.

  The case study is divided into the following parts:

  - Case Study Focus
    The first phase of the case study includes various steps. First, the author proposes research propositions which are related to the defined research question. Next, the unit of analysis as well as the boundaries of the case study are defined. Subsequently the author introduces the selected cases to the reader and furthermore selects the cases for the pilot interview.
  - Design of a Detailed Plan
    In this phase all necessary steps to execute the case study are planned in detail. This includes the definition of the data collection strategy and the conduct of the pilot interview.

  - Data Collection
    The third phase of the case study is the actual collection of data as defined in the prior phase *Design of a Detailed Plan*. The collection of data is based on qualitative interviews with roles working in the selected organizations.
  - Data Analysis
    The last step in course of the case study is the analysis of the collected data.

- Phase 4 - Discussion
  The discussion of the results of the case study and the theoretical research is the last phase of the thesis. This includes the answering of the research question.

Figure 1.1 summarizes the methodological approach of this research.

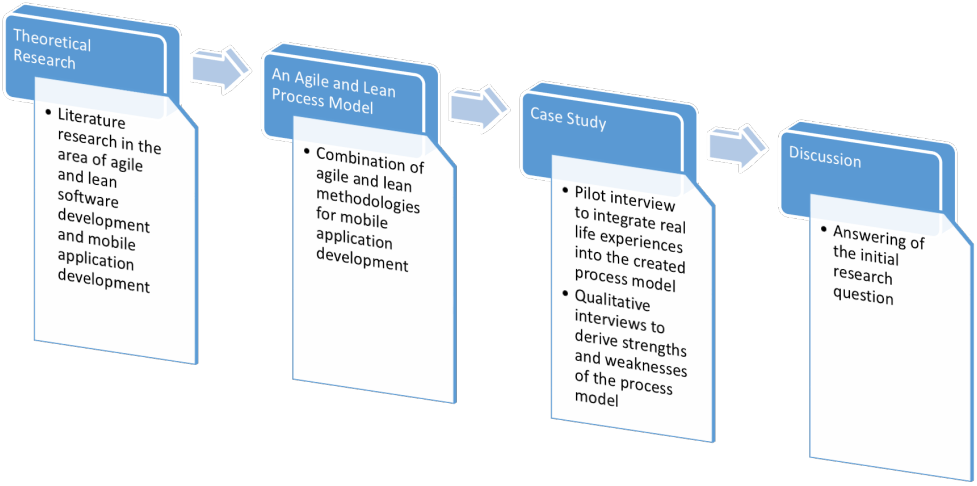

Figure 1.1: Methodological approach of the research

In the following the author maps the different topics to the chapters of this thesis. Chapter 2 introduces the reader to software process models in general. Moreover the author examines the agile and lean methodology in this part of the thesis. Additionally this chapter focuses on distributed working. The third chapter analyses the topic of mobile application development. The main theoretical part of this thesis is then presented in chapter 4 in which the author proposes an agile and lean software process model for mobile application development. Chapter 5 focuses on the case study, in which practical experiences in the area of mobile application development are observed. The information gathered through the theoretical research and the practical part of this thesis are discussed in chapter 6. After the discussion, chapter 7 sums up the main results of the thesis and conclusively gives an outlook into the future.

2 Theoretical Background

This chapter focuses on the basic concepts of software process models. At first, the author concentrates on traditional software engineering. Subsequently this part of the thesis examines the topic of agile and lean software development in detail. By presenting a typical process model for traditional software engineering and agile respectively lean software development the author illustrates the differences between those approaches. After that, the author focuses on the topic of distributed working.

2.1 Software Process Models

For more than over half a century academics and software related industry personnel have created frameworks and process models to support the development of software. This support was especially addressed to structure, manage and control software processes as a centered work. Those frameworks have their primary focus on the management of process models in order to achieve higher productivity, decrease development time and provide cheaper development services. [24]

Pressman defines in his book *Software Engineering: A Practitioner's Approach* a software process as "a framework for the activities, actions, and tasks that are required to build high-quality software" [25, p. 31]. As stated in chapter 1, a standardized process is needed to develop software successfully. Tyrrell defines the purpose of a process in more detail. He lists seven key goals what a process should achieve: [26]

- Effectiveness
  A software can be perfectly well written and developed in a minimum of time, but if it fails to be the software the customer demanded, it is just not good. Therefore effectiveness of a process helps to produce the right and required product the customer wanted.
- Maintainability
  It doesn't matter how good the programmer was who coded the software - problems will occur. Updates need to be established to keep up with changing requirements, patches need to be installed to ensure the security of the software, and so forth. Thus the maintainability of the software needs to be guaranteed.
- Predictability
  Especially from a economical point of view this goal is very important. A process helps to calculate the resources a new product is going to allocate. Without a proper plan about the effort of a production, predictions can't be made, which could lead to huge issues in the overall calculation of a software project.
- Repeatability
  If a process has been successful, it helps a lot in future projects to repeat this process, since the overhead of planning the process is not necessary.
- Quality
  Tyrrell also lists quality as one of the key goals a software process should achieve. He also points out that one of the objectives of a software process is to ensure the high quality of its

product. Moreover he states: “The process should provide a clear link between a customer’s desires and a developer’s product” [26, p. 4].

- Improvement
  No process can be perfect from the beginning on. Upcoming changes will require to adapt the process to the new demands. Thus a process should be capable of improving itself.

- Tracking
  It is important to keep track about the status of a project. This is the counterpart to the *predictability* of a process, since it can measure how good the made predictions are.

Most of the new software development processes can be categorized according to their principles they follow. On the one hand there are traditional software process models and on the other hand there are agile processes [24]. In the next part of this thesis a comparison between traditional software engineering and agile respectively lean software development is performed. In order to achieve this, each approach gets illustrated in detail and then further explained by introducing a typical process model in the respective field of software development.

2.2 Traditional Software Engineering

The topic of developing software does not have a very long history. Nevertheless the thesis will focus on the early years of this branch of industry before getting into more detail about specific process models in traditional software engineering.

2.2.1 The Early Years of Software Development

When the first electronic digital computers arrived in the 1940s software development was not seen as this important industry it is today. Software development was thought as a kind of engineering - more like a craft which was learned and then executed by craftsmen and craftswomen. There were no processes or methodologies involved in the act of producing software. The craft was just passed from the master to newcomers who eventually also became skilled and experienced in this craft [15].

With the increase of computing power and the availability of large computers to scientific institutions and universities in the late 1950s the craft of developing software became more and more complex. As a result of the emergence of computers from closed laboratories to the pubic domain the use of computers became an activity of the many. With the arrival of high-level programming languages the complexity of software development increased even further. This complexity led to software projects which were often delivered late, over budget and not fitting all requirements. This situation is known as the *software crisis* and was first addressed at the first NATO Software Engineering Conference in 1968 at Garmisch, Germany. [15, 27]

Dijkstra wrote in his article *The Humble Programmer* from the year 1972 the following about the software crisis: “The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.” [28, p. 861].

The outcome of the before mentioned NATO Software Engineering Conference in 1968 was that experts realized that the current techniques used in software development were inadequate to meet

the new upcoming challenges in this sector of industry and that new processes and methodologies were necessary to overcome the software crisis. The term *software engineering* was first coined in the course of this conference. [27]

Software development was no longer seen as a craft but more as a form of engineering. Experts aimed for a framework with which it was possible to develop complex and large software projects which meet the requirements a customer demanded, on-time, within the defined budget and with fewer bugs [15].

The first traditional software engineering process models emerged in the late 1960s. Classical software engineering methodologies are often referred as *plan-driven* or *heavy-weight*. Upfront they require detailed plans, documentation and requirements definition. One famous representative is the *waterfall model*, which will be discussed in more detail below. [29, 30]

2.2.2 The Waterfall Process Model

The waterfall model, sometimes also referred as the *classic life cycle*, is the oldest paradigm of software engineering [25]. It expresses a systematic, linear approach in software development. The term *waterfall* was derived from the idea that the progress in this process model is seen as steadily flowing downwards. It was first introduced in 1970 by Winston Royce in his article *Managing the development of large software systems* [31]. In this article he proposed a model which included feedback loops between the different steps in the process. Those feedback loops where removed later on by the vast majority of organizations which applied the process as strictly linear. [25]

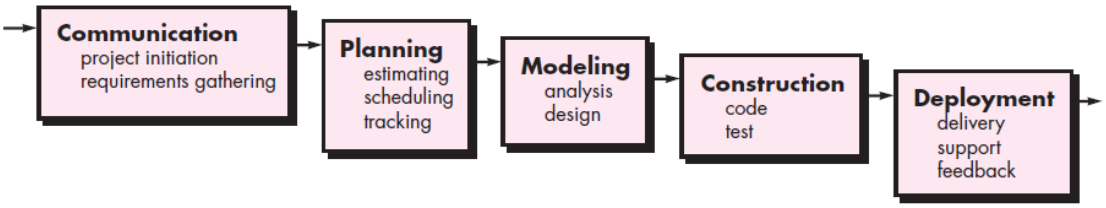

Figure 2.1: The waterfall process model [25]

The process was mainly addressed to management issues which arise in the development phases of a software project [29]. Figure 2.1 illustrates the basic concept of the waterfall model. The model features five main steps, which are executed sequentially: [31]

1. Communication
   In the first phase of the process the requirements of the software project are gathered. This requires intensive communication with the customer.

2. Planning
   After the first phase, the software project gets planned. This involves risk management, scheduling of tasks and further estimations related to the project.

3. Modeling
   In the modeling phase the architecture of the system is designed. This is one of the most important phases which also includes the documentation of the design. In all traditional software engineering processes documentation is essential.

4. Construction
   The actual coding is performed in the second last step of the process. Additionally to coding, testing is executed.

5. Deployment
   As a last step the software gets delivered. The original waterfall model proposes an ongoing support of the software after the completion of the process.

Through the linearity of the process every stage has to be completed before the next stage is executed. This fact is often criticized by proponents of agile development methodologies, since it constrains software project members in their flexibility. Due to the linearity of the process, the model fails to detect problems in early stages, which leads to huge refactoring tasks later on. Another problem which is encountered when applying the waterfall model is the circumstance that it is hard for customers to state all needed requirements in the very beginning of a software project. A further issue, which critics of the waterfall model complain about, is that customers have to be patient to see a first version of their product, because the phases before the real implementation take very long in this process model and in traditional software process models in general. [25, 32]

Today software development is very fast paced and developers often have to adapt their work to changing requirements. The waterfall model is therefore not suitable for such projects, nevertheless it can support software projects in which requirements are fixed and not likely to change. [25]

In general plan-driven process models are recommended for risky and large-scale software projects [15] which promise predictability, stability and high assurance [33].

2.3 Agile and Lean Software Development

In his article *Software Engineering: An Idea Whose Time Has Come and Gone?* DeMarco questions the fact that the idea of engineering software is a relict of the past [34]. He writes about his early metrics book from 1982 *Controlling Software Projects: Management, Measurement & Estimation* [35] in which he emphasizes that control is the key to successful software projects. The quote "You can't control what you can't measure." stands for his perception in that time. 40 years after the NATO conference in Garmisch he changed his view about controlling software projects. He still thinks that consistency and predictability play a decisive role in software development, but he also states that over the past decades those values have never been the most important ones. The key to success is creating software that "changes the world" and not being stuck with unfinished projects. DeMarco himself states that he is too far away from actual building software to give advice which methods should be used in software development, but his perception about the management of a software process is very similar to agile and lean software development methodologies.

The next part of this research will deal with agile and lean software development. At first the author introduces the reader into the basics of those approaches. Subsequently a typical representative of agile and lean software development is presented in order to give a more practical insight into those methodologies.

2.3.1 The Idea behind the Agile and Lean Approach

As mentioned in the previous section of this thesis, it is hard to predict how software will evolve as time passes. There are different variables which change during the development time of a software project, e.g. the rapid change of market conditions or changing demands of customers. It is often impossible to define all requirements before the implementation phase in a software process starts, as traditional software engineering processes suggest. There is a need for agility to react to changing business environments. [25]

Changing business environments bring one big disadvantage to software development in general. Adapting to change is always very expensive, especially if changes occur late in software projects. One of the most tempting characteristics of agile software development is its ability to reduce the cost of change [25]. Figure 2.2 illustrates the cost of change in traditional software processes compared to agile software processes.

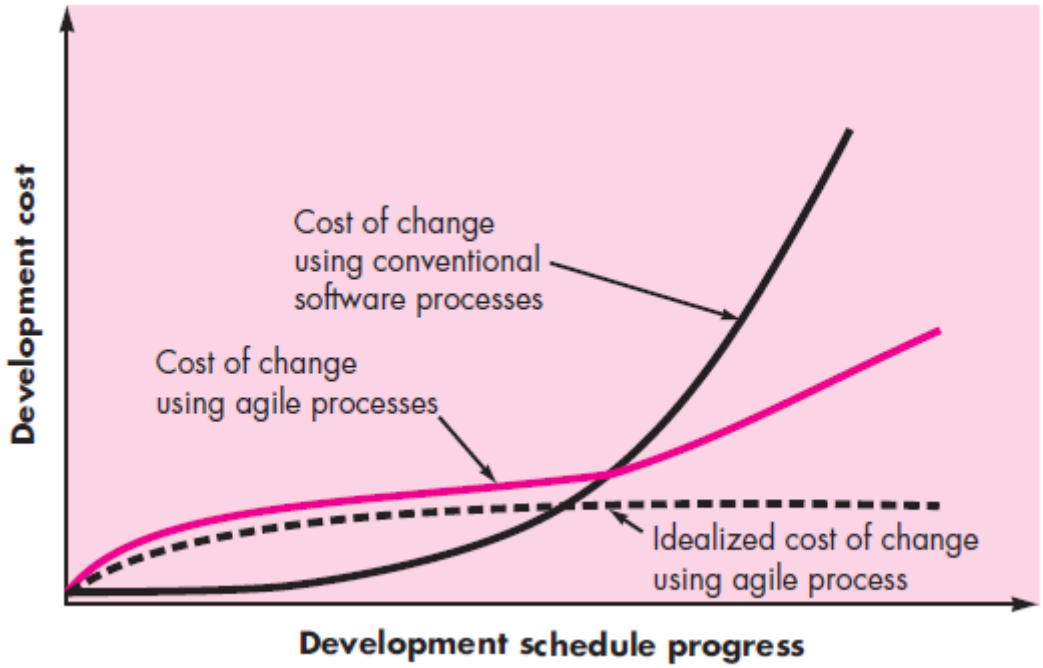

Figure 2.2: Change costs as a function of time in development [25]

In plan-driven software processes the costs increase non-linearly as the project progresses. The reason for the non-linear increase is very clear. A change in a late phase of a software project may require changes of earlier phases, like adapting the architectural design of the software. Such modifications are very expensive, because the stability of the implementation is threatened. To ensure the quality of the software, testing of the new modules must be performed, which increases the costs even further. In agile software development this curve can be flattened, which makes late changes in software projects not as time consuming and as expensive as in traditional software processes. [25]

As stated in chapter 1, agile and lean software development has become very popular in recent years. With the creation of the agile manifesto in 2001 the principles of agile manufacturing have been adapted to software development. Besides the important values the agile manifesto stands for, there are twelve principles of agile software [6]:

1. “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”
2. “Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.”
3. “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”
4. “Business people and developers must work together daily throughout the project.”
5. “Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.”
6. “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”
7. “Working software is the primary measure of progress.”
8. “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”
9. “Continuous attention to technical excellence and good design enhances agility.”
10. “Simplicity–the art of maximizing the amount of work not done–is essential.”
11. “The best architectures, requirements, and designs emerge from self-organizing teams.”
12. “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

Not all agile processes weight these principles equally and also not all processes implement every single principle, but agile processes in general use these twelve principles as a guideline. [25]

The main idea behind agile development is incremental and iterative development. This means that the phases in the software development life cycle are repeated again and again. The software, which should be developed rapidly, gets improved continuously by adapting to customer feedback. [30]

Proponents of traditional software engineering and critics of agile methods often compare agile methodologies to code hacking without any documentation. This is not true, but it is correct that more value is placed to planning than to documentation, which makes those processes look less plan orientated as they really are [36]. In traditional software processes a very high amount of documentation is produced, which mostly is not used anymore in later stages of the software project, because requirements changed during the development [37].

In the first chapter of this thesis the reader got briefly introduced into the history of lean software development. Following a more detailed insight into lean software development is given.

As stated, lean manufacturing has it origins in the 1940s long time before the principles of lean got adapted to software development. After the second world war the Japanese automobile industry struggled to compete with its opponents in America. In the US, mass production was invented by Henry Ford in 1913 and its goal was to produce large amounts of the same product to a very low price. A disadvantage of mass production is its inflexibility. Nevertheless this system worked well for the American automobile industry, but the Japanese market was too small for mass production. Furthermore there was a need for variety in automobiles in Japan, which made mass production

even more unfit. As a result of this situation experts came up with many ideas and techniques which later on evolved into the Toyota Production System. Through the use of this system Toyota was in the early 1990s 60% more productive and having 50% fewer defects. The system was based on the total elimination of waste in the course of the production. [38]

The first thorough mapping of lean principles to software development was published by Mary and Tom Poppendieck in 2003 in their book *Lean Software Development: An Agile Toolkit* [39]. Through this publication the term lean got established in the software world and furthermore associated with agile principles. The authors formed the seven principles of lean software development which will be examined in detail in the next section of the thesis. [5]

As pointed out in the previous paragraph, lean software development is associated with agile practices nowadays. The reason for that is the fact that they share the same goals. Both methodologies try to increase the productivity of the software development process as well as the quality of the product itself. A common misunderstanding is that agile and lean are the same, but this is not the case. Both follow similar goals, but the perspective is different. Lean is not as narrow as agile, it takes a wider view over the whole business context. Agile methods can be seen as supporting practices to lean software development. [38]

2.3.2 Agile and Lean in Detail

The previous section illustrated the main idea behind agile approaches. A good description of this idea was stated by Stober and Hansmann in their book *Agile Software Development: Best Practices for Large Software Development Projects*: "Agile thinking is an attempt to simplify things by reducing complexity of planning, by focusing on customer value, and by shaping a fruitful climate of participation and collaboration." [40, p. 35]

In this section of the thesis the author discusses the agile and lean methodology in more detail. Chapter 1 presented the values of agile software development. These values are listed again below: [6]

- "**Individuals and Interactions** over Processes and Tools"
- "**Working Software** over Comprehensive Documentation"
- "**Customer Collaboration** over Contract Negotiation"
- "**Responding to Change** over Following a Plan"

An important sentence in the agile manifesto which is often not recognized or ignored is directly following these values: "That is, while there is value in the items on the right, we value the items on the left more." [6] A common misinterpretation of the agile manifesto is that only the values on the left are relevant, but as stated in the manifesto the values on the right are also important in agile approaches.

In the following each of the values proposed by the agile manifesto gets examined: [41]

- Individuals and Interactions over Processes and Tools
  The first principle of the agile manifesto clarifies the significance of individuals in the software development process. Especially in traditional software engineering approaches processes and tools were seen as one of the most important parts in software development. This principle encourages software developers to appraise the individuals which are part of the development process. This also includes the interactions of those people and furthermore

reminds everybody who is a part of an agile process to take care of other people involved in this process. In essence this means that before an action is taken related to the software development process the influence of this action on other people working on this project should be considered. This influence includes the communication, environment and relationship of the persons. Furthermore Hazzan and Dubinsky name an example in which organizations often stick to the use of tools which are hard to understand and processes which are hard to follow instead of creating an environment for each of the participants in which they can be an integral part of the process itself and which enables the ability to collaborate with all other stakeholders in the project.

- Working Software over Comprehensive Documentation
  This principle states the fact that the main objective in software development is to create quality software. Hazzan and Dubinsky list three implications which come with this principle:
  - The focus in an agile approach is set on the development itself rather on the documentation of it. A result of this behavior is the fact that only the most necessary and essential documentation is produced. Based on the importance and characteristics of this documentation some of it can be posted on a wall to make it available for all stakeholders anytime.
  - Agile methods have the goal to create executable software as soon as possible in the software development process. The outcome of this goal is that people involved in the software process (e.g. management, customers, developers,...) get an early understanding of the vision of the product. Changes can be addressed sooner, because issues, which would not have been visible, can get noticed and examined.
  - Through the focus on working software in early stages of the development process, the delivered software will feature a higher quality and fewer bugs respectively problems in the implementation.

  Regarding to Hazzan and Dubinsky the importance of this principle can be shown when comparing the implications of early working software to processes in which the development stages or activities related to quality are postponed in the overall development process. Methodologies in which the actual coding only starts after the creation of a lot of documentation, which includes a very detailed requirements analysis, struggle with the fact that in reality user requirements change during the development process. Due to the fact that the documentation is based on the initial perceptions of the customer, the finished product will not match the underlying documentation of the product, because the product got adapted to changing requirements. Regarding the postponement of quality activities, developers will face a probably unmanageable challenge to fully test the product, because the effort to test everything after coding is too high. The proposed agile solution is to test on every iteration of the development process.

- Customer Collaboration over Contract Negotiation
  The idea behind this principle is to change the way people see the customer's role within a software project. This value should inspire developers to build up a closer relationship with customers which is based on ongoing daily contact. Due to this close relationship it is possible for customers to keep track over the progress in the software project. Furthermore a higher communication between customers and the management is provided. Hazzan and Dubinsky state further: "These interrelations in turn have direct implications on the development team, which should employ specific practices to ensure this kind of relationship and communication. Such practices, when employed on a daily basis, directly influence the culture of agile organizations." [41, p. 7] They also state that the idea of a close relationship

with customers also helps in contract and communication related issues, because due to this fact the chances that the customer gets his desired product are increased.

- Responding to Change over Following a Plan
  The last principle faces the ability of a process to react to change. It encourages businesses to come up with a methodology which enables the team within a software project to respond to changes without jeopardizing the quality of the product. This principle accepts the fact that customers are not able to define all needed requirements before the development process starts. Therefore the process must be able to adapt to new inputs the customer is providing the project team with. This principle highly correlates with the decreased costs of change in agile processes, which was addressed in the previous section of this thesis.

After the focus on the agile principles, a deeper insight into the lean methodology is given. The first part of the thesis already introduced the reader into the origins of lean manufacturing and its adaption to software development. Furthermore the author named the seven principles of lean software development. These principles get investigated in more detail below:

- Optimize the Whole
  In order to develop a product, which fits the customers needs, a deeper understanding of the whole production process is needed. It is necessary to have a look on the complete value stream. This means that an investigation of values a customer will care about is needed. It is important to understand what factors are necessary to create a successful and profitable product. Through this deeper understanding it is then possible to optimize the whole process of developing the product as well as the product itself. This optimization can only be performed by reviewing the process and the product continuously. [5, 40]

- Eliminate Waste
  Poppendieck and Cusumano define in [5] waste as "anything that doesn't either add customer value directly or add knowledge about how to deliver that value more effectively."[5, p. 28]

  There are multiple ways to introduce such waste to a software project. Next, some of these ways are listed and grouped to a specific category: [40]

  - Technology
    It is often the case that a developer focuses too much on fancy technologies which bring no real value to the project. This should be avoided by only concentrating on features respectively use cases which help to improve the product the customer demands. Regarding this issue Stober and Hansmann name the 80-20 rule as an indicator. The 80-20 rule says that only 20% of the features of a product bring 80% of the product's value. A further example of waste is the fact that software engineers are often introducing new technologies into a software project, because they have a personal interest into this specific technology. The problem is that in most of the cases this new feature does not add any business value to the project. Even if the developer learns to use this technology in his free time and therefore spends no working time on getting familiar with this specific technology it brings new potential bugs into the software. Furthermore it raises the maintenance costs of the product.
  - Project Management
    In many software projects excessive efforts are invested in administration and project management. Unfortunately these efforts are sometimes useless, because they miss the intended target. One could for example waste time and resources on the planning

of just too many requirements respectively features which are not possible to be completed in the given time frame. Another possibility could be the specification of future functionalities for which the requirements are likely to change in the course of the process and therefore need to be adapted to the upcoming modifications.

  - Organizational Boundaries
    Waste can also be introduced to a project due to organizational boundaries. These boundaries slow down the complete process and furthermore increase the costs of the project. Such boundaries usually occur when different organizations are involved in a project. Stober and Hansmann mention different organizations for development and testing, instead of having one team for both tasks as an example.

- Build Quality In
  Through the use of test driven development a proper working code is created from the beginning on. This prevents the situation that the test team finds too many defects at a time which is a clear indicator for a process which is not working properly. Due to the creation of tested code directly from the start of the development on, it is possible to integrate small parts continuously into the overall product. Through continuous integration high efforts to merge different code bases are avoided. To build quality in also refers to the fact that the expectations of the customers must be addressed. An error free source code does not do the job as long as it does not represent the visions of the customer. [40]

- Learn Constantly
  Software development is the process of embedding knowledge into a product. Depending on the context lean development recommends two different ways on how to embed this knowledge into a product. [5]

  - "Expensive-to-change" decisions should be postponed as much as possible to be in a position to base the decision on the best knowledge available. These decisions include for example fundamental architectural design decisions or the right choice of the used programming language. The reason for the postponement is that by exploring multiple options, the appropriate method, which optimizes the whole system the most, can be chosen.
  - The other approach to embed knowledge into a product is to start with a minimum subset of capabilities of a product and to improve this product continuously. This improvement involves feedback of customers to make content decisions of the product. This way decreases the efforts spent on features customers do not even want to have in their product.

  The main idea behind those two approaches is the fact that the content of software systems changes constantly and therefore it is important to be able to adapt to these changes.

- Deliver Fast
  Lean development recommends to deliver software fast. Automated testing and mistake proofing mechanism have dramatically increased the quality of high paced software releases. Furthermore the overhead which originally resulted in software releases could be decreased through the use of such methodologies. Another reason for the fast delivery of software is the fact that development teams which are overloaded with work slow down the progress. The advantage of small changes per release is that the productivity of teams can be increased, because the workload is reduced to a realistic capacity. Furthermore lean development suggests that development teams deliver end-to-end use cases, which further increases the productivity, because the progress in the development does not depend on other people or teams. The hand-overs within an organization are reduced, while the collaboration

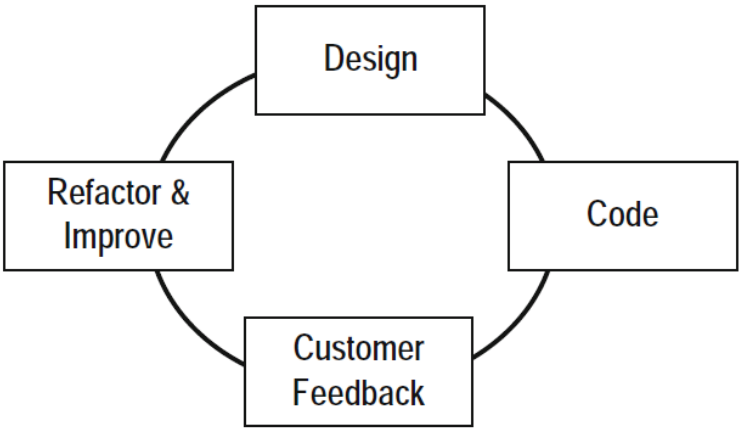

Figure 2.3: Lean development process [40]

with the customer is increased. Due to the increased number of releases the customer can sooner see his product and therefore help to meet his expectations of the final product. The software development in such fast paced environments should not be seen as a project which is completed when the final product is created. Instead it should be seen as a steady flow where software is designed, developed and delivered in small pieces. Figure 2.3 illustrates the development cycle in lean development. It goes from the design over to code which is then delivered and feedback of the customer is gathered. With the feedback the product gets further improved. [5, 40]

- Engage Everyone
  Poppendieck and Cusumano write in [5] that software development should not be seen as a separate department in an organization. Because the development of a product includes more than just writing source code, a software development department should represent a miniature version of the business. As declared in the first principle *Optimize the Whole*, it is necessary for a development team to see the whole value stream, beginning by discovery, over creation to delivery of value. Furthermore Poppendieck and Cusumano state: "Even when software development occurs in a separate organization, lean practices encourage teamwork among engaged people who are empowered to make decisions appropriate to their level." [5, p. 29] In lean practices it is suggested to move decision-making to the lowest possible level within an organization to further encourage self empowered personnel.

- Keep Getting Better
  Lean thinking recommends to improve processes constantly. Methods may work very well in other environments, but it is essential to find the best possible solution for a specific situation. Organizations should not use practices like scrum without thinking about chances to get better. These practices should more function as a good starting point for a process. Every process then needs to be adapted to the characteristics which occur in particular settings. [5]

As stated in the first chapter agile and lean methodologies follow very similar principles. The combination of those two approaches is performed in chapter 4.

A common practice when implementing lean software development is to take a typical agile process model as a basis for the software development and then to start applying lean principles and

tools to it [38]. In the next part of the thesis such a typical agile process model is presented to the reader.

2.3.3 The Scrum Framework

In 1986 *scrum* was first introduced to the world. Takeuchi and Nonaka wrote in their article *The New New Product Development Game* [42] about companies in Japan and the United States which have taken a new approach to manage their product development process. With this new approach these companies were able to produce world-class products in a fast and flexible way. One of the key aspects in this new development process are empowered, self-organizing project teams. In 2001 Ken Schwaber and Mike Beedle published the first book about using scrum in software development [43].

Rubin states in his book *Essential Scrum: A Practical Guide to the Most Popular Agile Process* that "scrum is an agile approach for developing innovative products and services" [44, p. 1]. It is not a standardized process which guarantees businesses that they finish their software project on time and within budget. "It is a framework for organizing and managing work." [44, p. 13]

Before the core scrum framework gets presented to the reader, the different roles in this approach get illustrated. In a software project supported by scrum there are one or more *scrum teams* which all consist of three scrum roles[1]: 1) *product owner*, 2) *scrum master* and 3) the *development team*. Following each role gets presented in more detail: [44]

- Product owner
  The product owner represents the stakeholders in a software project. He is the central point of product leadership and he ensures that the team delivers value to the business. Furthermore he decides which features and components get implemented in the product. To guarantee the success of the project the product owner has to work very closely with the scrum master and the development team.
- Scrum master
  The scrum master functions as a servant leader in scrum teams. The main task of a scrum master is to make sure that the values and principles of scrum are implemented correctly in the scrum team. In contrast to classic project managers scrum masters do not have the authority of control within the team.
- Development team
  The development team is responsible for delivering working software to the customer. Therefore development teams consist of all necessary members to develop high quality software, e.g. software architect, programmer, tester etc. The development team should work self-organized and consists usually out of five to nine people.

After the introduction of the roles within the framework, the focus is now being set on the *scrum activities* and *scrum artifacts*. The basic concept of the framework is illustrated in figure 2.4. The scrum activities presented in the diagram are the so called *sprints* and *daily scrums*. The artifacts consist of the *product backlog* and the *sprint backlog*. To explain these terms in more detail, the course of action is described below.

In the beginning the product owner has a vision of a product and therefore creates the product backlog in which he defines a set of features. These features are prioritized within the product

[1] The scrum framework requires these three roles, but there are no restrictions to have more roles.

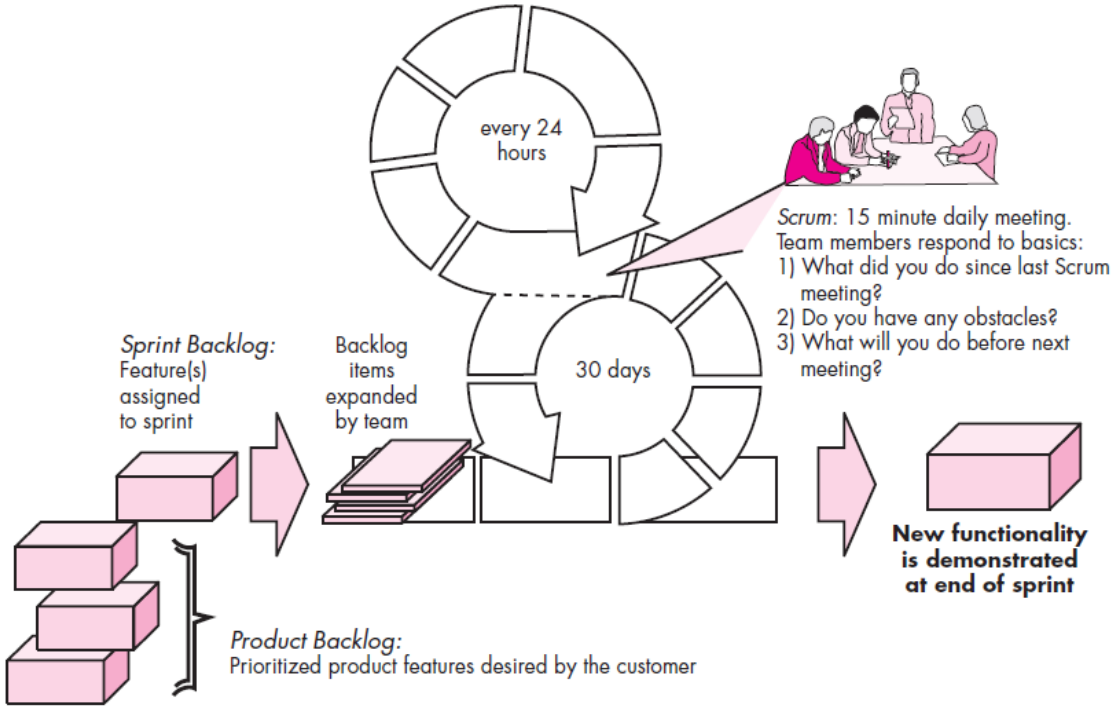

Figure 2.4: The scrum framework [25]

backlog. In scrum, the work is performed in sprints which are iterated and cycled. Sprints can last up to one month and each sprint should bring value to the customer. A sprint has a fixed start and end date. Subsequently to a finished sprint the next one starts. To create a stable working environment changes in the requirements are not allowed during a sprint. As figure 2.4 shows, a sprint backlog is defined in which the development team selects a subset of the product backlog which should be finished within this sprint. [44]

Next in the scrum framework is the sprint execution in which the development team implements the before selected features of the product. Daily scrums are meetings which are held every day. A meeting shouldn't last longer than fifteen minutes and should help the team to find problems as early as possible. Usually each team member should answer the following three questions within every daily scrum: [25, p. 84]

1. "What did you do since the last team meeting?"
2. "What obstacles are you encountering?"
3. "What do you plan to accomplish by the next team meeting?"

At the end of each sprint a potentially shippable product should be available which features a subset of the vision of the stakeholders. Furthermore the scrum team performs two further activities which are not displayed in the diagram. The first activity is the so called *sprint review* in which the stakeholders and the scrum team review the created product. The second activity is the so called *sprint retrospective* in which the scrum process itself is being evaluated. The outcome of these two activities is a possible change of features or the development process. [44]

The above explanation of scrum should function as a very brief description of the relation between scrum roles, activities and artifacts.

2.3.4 Kanban

In order to give a more detailed insight into the lean methodology the author presents *kanban*, a typical representative of this methodology.

Kanban is a practice which originally was used in lean manufacturing and has been transformed for the use in software development. Basically kanban is a simple and effective scheduling system which focuses on maximizing flow and minimizing work in progress. In lean software development kanban is usually implemented using colored sticky notes and a whiteboard. Ikonen et al. state about the benefits of kanban: "Kanban attempts to lower production costs, increase quality, and accelerate cycle time." [45, p. 306]

In [46] Kniberg and Skarin state the three major constituents of kanban:

- Visualize the workflow
  The visualization of the current workflow is one of the most important concepts in kanban. In kanban this is done by using a so called kanban board. This board contains columns which donate the current step in the workflow. Work itself is split into pieces, written on a card and then visualized on the board.

  Transparent information is necessary so that tasks can be completed in a self-organized manner. Every person in the process should benefit from this transparency. Among other things those information are: [47]

  - The phases in the complete workflow
  - The tasks which need to be executed during workflow
  - The people who work on a specific task
  - The limitation of work in progress
  - The metrics which define the work progress

- Limit Work In Progress (WIP)
  In kanban the work is limited by an explicit number of items at each working step.

- Measure the lead time
  The progress in kanban in measured via the so called *lead time*. This time denotes the average time for one work item to run through the whole process. It is often referred to as *cycle time*. By decreasing the lead time to a minimum the process gets optimized. Additionally the lead time should be predictable for future estimations.

Hibbs illustrates in [38] the functionality of kanban with the help of an example usage. In this example a process is used which consists out of five steps: 1) design, 2) implementation, 3) test, 4) build and 5) deploy. Each request will run through the complete lifecycle in order to be finished. The sticky notes are representing the various tasks in the process. Different colors can be used to mark different types of work items. Figure 2.5 visualizes the use of the kanban board.

As stated, the idea of kanban is to limit the work in progress in each step. In the example this limit is denoted by the number below the label of the step, e.g. implementation has a limit of three, whereas design and test have a limit of two. This limit is now the maximum of work items which are allowed to be in this step at any given time. Only if a step is below limit, a new work item can be pulled from the previous step. That means that if a step has reached its limit it must wait to get reduced by a pull of the next step in order to be able to pull a new item. This visualization should help to show the whole process respectively the current situation in the project at a glance. As many lean practices kanban is not very prescriptive, but it focuses on a steady improvement of the whole process.

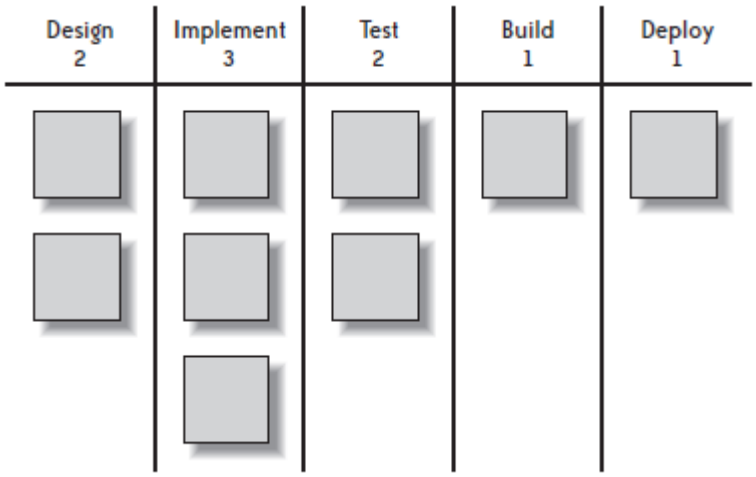

Figure 2.5: Example kanban board [38]

2.3.5 A Comparison To Traditional Software Engineering

By presenting one typical process model for traditional software engineering and agile respectively lean methodologies the reader got introduced into the basic concepts of these approaches. Following a direct comparison of agile and traditional approaches is performed in order to illustrate strength and weaknesses of both methods. Table 2.1 presents an overview over the main characteristics of each approach.

As already stated in previous sections, traditional software engineering is mainly used in large-scale software projects. That does not imply that agile or lean software development cannot be used for such projects, but the literature recommends agile methods for small- and mid-size software projects. Proponents of the agile approach often see the reduced costs of change as one of the biggest advantages compared to plan-driven methods. This fact is highly influenced by the ongoing testing performed in agile methods compared to testing in plan-driven methods which is performed after the coding phase is completed. Due to this fact errors can be found earlier in the development process.

Table 2.1 also mentions interpersonal skills which are necessary for developers in the agile approach. This is very important because not every employee in an organization is able to work

	Agile methods	**Traditional approaches**
User requirement	iterative acquisition	detailed user requirements are well defined before coding/implementation
Rework cost	low	high
Development direction	readily changeable	fixed
Testing	on every iteration	after coding phase completed
Customer involvement	high	low
Extra quality required for developers	interpersonal skills & basic business knowledge	nothing in particular
Suitable project scale	low to medium-scaled	large-scaled

Table 2.1: Comparison of agile methods and traditional approaches in software development [30]

successfully in teams. Businesses should keep that in mind when they decide which methodology to use for their projects.

2.4 Distributed Working

As briefly discussed in chapter 1, global software development is a common practice nowadays. With the emergence of high-speed Internet services and the improvement of distance collaboration tools organizations more likely expand their businesses to other countries. Those advances in technologies even bring the situation that colleagues working within the same city *telecommute*[2] in a distributed fashion at least part of their working time. [17]

Hazzan and Dubinsky name in [41] reasons for distributed working: "The motivation for global development usually stems from the need to use the organization's resources costcompetitively, and the need to shorten time to market by around-the-clock development." [41, p. 191-192]

Woodward et al. present in [17] different levels of distributed working, which are displayed in figure 2.6. Following each of the levels is discussed: [17]

- Collocated
  "To be collocated, a team must meet frequently." [17, p. 9] In essence, this citation defines what a team has to do in order to be collocated. It describes a normal working environment where team members have their own space, but they meet face-to-face in a regular manner. Due to global software development the number of such teams is decreasing.
- Collocated Part Time
  Teams in a collocated part time environment usually work all in the same physical location, but part time team members are working off-site. This may include home offices or working from another business location. Such teams need to face some of the challenges which arise from distributed working, but usually the team members are able to meet in person if necessary.
- Distributed with Overlapping Work Hours
  Situations where team members only share a specific amount of working time is described in this level. Such a situation occurs when coworkers are working in different time zones. Therefore there are only a few hours during a normal work day in which the team members can collaborate.
- Distributed with No Overlapping Work Hours
  Such teams have no chance to interact with each other during normal working hours. This situation is the most challenging in the context of distributed software development. Using asynchronous communication methods like email is one chance to overcome this problem.

According to Sangwan et al. teams which work in a distributed working environment have to face various challenges in this field of software development. Following some of these challenges are named and investigated in more detail: [49]

Due to the fact that global software development projects often come in different shapes and sizes it is not uncommon that those projects have to deal with different cultures and languages among the staff working on this project. Furthermore such projects can involve different organizations

[2] The term telecommuting means to work from a remote location and was first coined by Jack Nilles in 1973. [48]

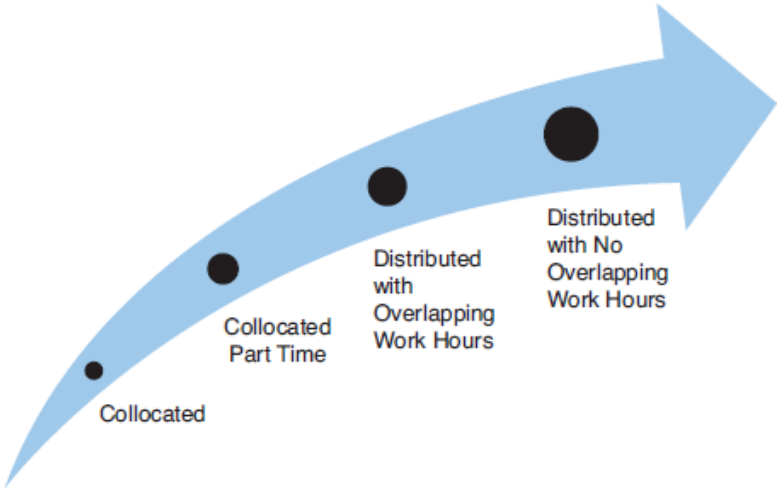

Figure 2.6: Levels of distribution [17]

and divisions, which makes the whole coordination overhead even bigger. It is often the case that participants in projects have not met before. Additionally the level of experience among the participants is very likely to be different. All these issues make it very hard to manage global software development projects. According to [49] studies have shown that tasks in a distributed setting take two and a half times as long to complete as in a collocated setting.

Sangwan et al. state further that people have often difficulties in coordinating complex tasks before they have met in person. It is hard to find the right person to talk to on a remote site, because the necessary insight into the structure respectively people of the remote site is not given. It is also mentioned that people tend to make assumptions rather than asking the remote site, if specific information is not available. This leads to problems, because the assumptions are very likely not to match the requirements of the product.

Another problem when dealing with people without knowing them personally is the fact that complains or improvement suggestions could be interpreted as criticism. Additionally a worker on a site maybe does not see the urgency of a task, but this task could be very important for a remote colleague. Problems in communication like local holidays which are not celebrated at the remote site and therefore people are unreachable add another problem to the whole management issue in global software development.

Cultural and language differences among different remote locations are another issue. People who are not native speakers of a specific language often struggle to participate in teleconferences. Email or other non verbal communication methods are preferred, which is often slower and sometimes more frustrating than talking about issues. Cultural differences in the way of communication are another issue which needs to be addressed in global software development. Some cultures are very open minding and direct when it comes to talking about problems. This directness can be intimidating or even disturbing to other cultures which brings up a bad relationship among the different sites.

Technical challenges in distributed working are of course another huge issue to take in global software development. It takes time to build up a technical infrastructure, to deal with connectivity issues or to implement a working development process for all organizations which participate in a project. Those tasks can be very time consuming and difficult to achieve, but they are necessary to be successful in distributed working.

The above mentioned problems are only a few issues which arise in distributed working. Nevertheless some of those issues will get addressed in the following part of this chapter in which

the author focuses on the communication in a distributed environment. As mentioned, distance, different time zones, language barriers and cultural differences increase the complexity of communication in global software development. Woodward et al. state further that although people with the same cultural background and no language barriers have significant problems in distributed communication. According to Edward T. Hall, a social anthropologist, more than 65 percent of social meanings in a communication occur in a non-verbal way. This means that distributed teams which communicate mostly via telephone only have 35 percent of the normal communication channel available. This issue can be compensated by the use of video conferences, but the technological infrastructure for video conferences is either quite expensive or just not available to distributed teams. Knowing a person personally helps in such cases too. Therefore a kick-off meeting in which the participants of a project meet in person can help to get to know each other's personalities and communication styles. [17]

In [49] Sangwan et al. write about communication drivers and barriers in globally distributed software projects. Furthermore they investigate coordination possibilities and ways to control such communication. In the next part of the thesis the main ideas will be listed and examined in detail: [49]

- Communication Drivers
  According to Sangwan et al. there are two main communication drivers in collaborating teams: 1) task interdependence and 2) organization bond. Furthermore they state: "The higher the degree of interdependence among tasks, the greater the need for communication; this need increases even further if such communication is fraught with uncertainty and ambiguity." [49, p. 170]

  Concerning the boundaries in an organization, communication occurs more frequently among members belonging to the same group in contrast to people belonging to different groups.

  Especially in global software development those drivers become even more significant, because the communication in geographically distributed organizations can become very challenging in respect to control and coordination.

- Communication Barriers
  Some barriers in communication among distributed teams have already been pointed out. Following an overview over the main challenges in this specific field of global software development is given:

  - Physical distance
    The problem of physical distance got already addressed in this section of the thesis. Sangwan et al. mention further that a drastic drop in communication appears, if the distance exceeds 50 meters. This is an interesting fact, because collaborating teams which operate in the same city will have to face similar problems like teams working on different continents.
  - Overlapping working time
    Concerning overlapping working time the communication decreases as the overlapping working time decreases. As a consequence face-to-face meetings become very rare and the telephone is used more frequently. If the overlapping working time decreases further, email is mainly used as the preferred communication channel.
  - Cultural and language differences
    As stated above cultural and language differences are a real problem in a distributed working environment. It is in the nature of humans to bond with similar groups of people. Therefore it is more likely that a good communication is established between

people with the same cultural background. The more different the culture in collaborating teams the more heavily email as an asynchronous communication channel is selected, which reduces the effectiveness in the communication massively.

  - Organizational motivation and trust
    Different interests in the outcome of a project are another barrier in collaborating teams. It is often the case that one side of a partnership is more interested in the success of a project than the other. In such cases the effectiveness of the collaboration is going to suffer.
  - Personal relationship
    Communication between teams in which the team members does not know each other have to face the fact that communication is less likely to occur compared to teams with people having a personal relationship.

- Communication and Coordination
  It is stated in [49] that coordinated teams have a higher level of performance. But coordination in global software development requires intense communication, which is a challenging task. To improve the communication and coordination Sangwan et al. recommend to use the following techniques from organizational theory:

  1. Coordination theory
     According to Malone and Crowston coordination theory is defined as "a body of principles about how activities can be coordinated, that is, about how actors can work together harmoniously." [50, p. 358]
     Sangwan et al. define coordination theory as follows: "This theory stipulates coordination as the act of managing dependencies among activities." [49, p. 173] A possible example is a prerequisite constraint dependency, e.g. an approval of a change request before an engineer can execute the request. A change management procedure can help to manage such a dependency.
  2. Communication genres
     Communication genres are specific tasks or actions in order to help to accomplish a certain goal. Daily scrums respectively team meetings are possible examples for such an action.
  3. Collective mind
     Especially in a distributed environment it is hard to create a situation in which all collaborating teams understand the necessary aspects to complete a common goal successfully, but a shared understanding is a critical success factor in projects.

  The above enumeration shows techniques to coordinate work in a distributed environment. The task of coordinating collaborating teams is not trivial at all, but necessary to be successful in global software development.

- Communication and Control
  Without control over the communication performed in a distributed environment a situation may arise in which collaborating teams communicate too much, which could affect the performance of a project negatively, e.g. an overload of information which could be misleading. The reduction of dependencies in collaborating teams can be one strategy to control such a situation.

As examined above, communication is essential to succeed in a distributed working environment. Nevertheless coordination and control of this communication is also very important and must not be ignored.

3 Analysis of Mobile App Development

In the first chapter the reader already got familiar with the potential the mobile market offers to organizations. This part of the thesis examines this valuable business sector in more detail. In order to achieve this, the thesis focuses first on the history of smartphones respectively tablets. After that, the author presents statistics concerning the growth as well as the financial potential of the mobile market. Subsequently this chapter goes into more detail about the characteristics of mobile software and how they relate to agile respectively lean thinking. Conclusively a comparison of the approaches in the field of mobile app development is performed and the current state of the art, which is related to the topic of this master thesis, gets presented.

3.1 Smartphone & Tablet - A Brief History

"You can think of a smartphone as a contemporary cellphone combined with a handheld computer you can use to develop apps that run on the underlying platform." [51, p. 6] Duffy defines in this citation a smartphone according to its ability to run native applications on it. People also often consider the feature of a touchscreen as indicator for a smartphone. Whether it is the capability of installing third party apps or the functionality to use the screen as an input device, fact is that smartphones have become an essential part of todays society. Following the author presents a recap of the history of smartphones and tablets. [51]

In 1994 IBM and BellSouth introduced the first smartphone to the world. This historical important product was named *IBM Simon*. Simon was the first mobile phone to include a touchscreen as input capability to mobile devices. Furthermore it featured a calendar, an address book, a world clock, a calculator, a note pad, e-mail and games. The following years mainly hybrid devices were produced which were a combination of a mobile phone and a personal digital assistant (PDA). Usually those devices were capable of surfing the Internet and furthermore third party apps could be installed on it. [51]

In 2007 Nokia released its N95 which included a lot of features people associate to smartphones today, e.g. GPS, camera, Wi-Fi, etc. In the same year Apple announced the first iPhone. Through its innovative handling and its unprecedented user experience the iPhone lifted the idea of a smartphone to a new level. With the release of Google's Android, other manufactures produced new mobile devices, which further increased the growth of the mobile market. Late in 2010 Microsoft presented the Windows Phone 7 after recognizing the huge success of Apple and Google in the smartphone sector. [51]

Another product which has a significant impact on the growth of the mobile market is the tablet computer. In [52] Cortimiglia et al. define a tablet as "a lightweight, highly-portable device with an unmistakable architecture: a single panel covered by a touchscreen, which serves as its main input device." [52, p. 21] Furthmore they state that "although similar to smartphones in concept and design, tablets are substantially larger." [52, p. 21]

In the mid-1990s first experiments with tablet computers were made. The first notable product in this area was introduced by Microsoft in 2001 - the tablet PC. Microsoft described this new device as "as a mobile computer for field work whose screen could receive input through a special pen." [52, p. 21] The sales numbers of this new product line were disappointing though. In the following

different organizations were developing products in tablet format, but it was once again Apple to launch a widely used innovative device. With the release of the iPad in 2010 the tablet market started to boom. Other manufactures like Samsung released their own devices, mostly running the operating system Android.

3.2 The Mobile Market

As mentioned in chapter 1, the number of active smartphone users has topped one billion for the first time ever in 2012. No one could imagine how the release of the iPhone would change the mobile market. In the first quarter of 2013 smartphones outshipped feature phones for the first time ever. 216.2 million smartphones were shipped in this time period worldwide, which marks a 51.6% share of the total phone shipments [53]. According to Canalys this percentage will increase further to an estimated share of 73% in 2017. Those 73% will represent 1.5 billion smartphone shipments worldwide. The report states further that more than 1 billion out of the 1.5 billion units will feature Google's operating system Android. Compared to a total of 470 million Android-based shipped units in 2012, this number has more than doubled in 2017. Altogether Canalys expects a compound annual growth rate of 18% for the smartphone market. [54]

The boom of the smartphone and tablet market has a direct impact on the sales numbers of desktop computers and notebooks. Due to the rapid growth of other technologies the manufacturers of these devices suffer a lot. Table 3.1 shows the worldwide device shipments per segment. The total shipments of traditional PCs is forecast to 303 million sold devices in 2013. This number decreased by 11.2% compared to 2012. The total PC market (including ultramobiles) will decline by 8.4%, nevertheless the future for ultramobiles alone looks promising with an expected double of sales to 18.6 million units. In the same year mobile phones are expected to increase their sales numbers to a total of 1.8 billion units (a growth of 3.7%). Altogether the combined shipments are estimated to reach 2.3 billion units, which is an increase by 4.5 percent compared to 2012. This increase is mainly associated to cheaper products in nearly all segments. [55]

Especially the tablet market has a very bright future. With an expected growth rate of more than 50 percent in 2013 the popularity of tablets constantly increases. Gartner predicts 263 million tablets to be shipped in 2014. According to an IDC press release tablets will overcome the shipping numbers of the entire PC market in 2015. Figure 3.1 shows the estimated forecast for the tablet market compared to the PC market over 7 years from 2010 to 2017.

After getting into more detail about the sales figures of this market, the thesis will now focus more on the financial potential of this emergent business sector. The total shipments of smartphones respectively tablets are very impressive, but even more impressive are the statistics concerning the various app stores. Gartner estimates the total number of app downloads in 2013 to be 103 billion apps, which is an increase of about 59.4% compared to 2012. Furthermore Gartner expects the

Device Type	**2012**	**2013**	**2014**
PC (Desk-Based and Notebook)	341.273	303.100	281.568
Ultramobile	9.787	18.598	39.896
Tablet	120.203	184.431	263.229
Mobile Phone	1.746.177	1.810.304	1.905.030
Total	**2.217.440**	**2.316.433**	**2.489.723**

Table 3.1: Worldwide device shipments by segment (thousands of units) [55]

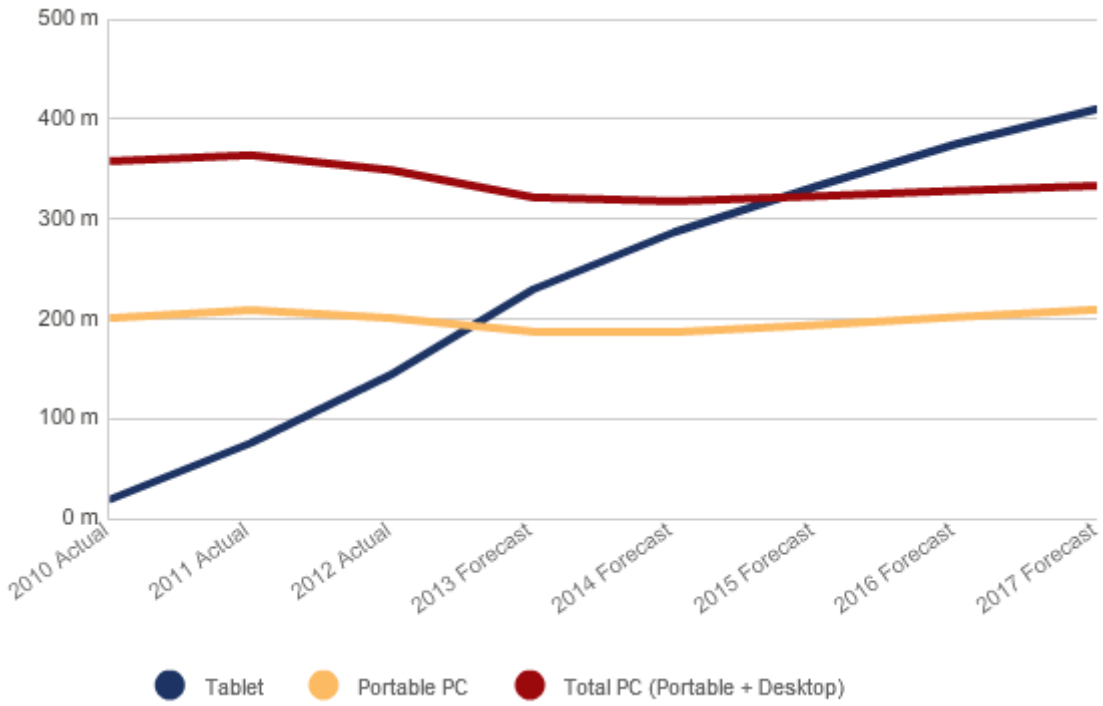

Figure 3.1: Worldwide tablet and PC forecast in million units [56]

revenue created by apps to be 26 billion dollars[1] in 2013. This amount increased by 44.4% from a total of $ 18 billion in 2012. From the total amount of downloads Gartner predicts 91% to be free of charge. A very profitable way to make money in the app business are the so called in-app purchases. According to Gartner 17% of the total revenue in 2013 was produced through in-app purchases. [57]

Futhermore Gartner predicts that in 2017 a total of 268.7 billion apps will be downloaded, from which 94.5% will be free. The in-app purchases will increase to a share of 48% of the total revenue. Additionally the report sees iOS and Android as the two leading operating systems for the next few years. By 2017 90% of all global app downloads will account either to iOS or to Android. The reason for this high share is the rich ecosystem both operating systems offer. The average monthly amount of downloaded apps will decline to 5.8 from 6.2 for Android devices and to 3.9 from 4.9 for iOS devices. This shows that people are more likely to use a good app more often than several apps with a lower quality. Additionally people will spend more money on high quality apps (e.g. through in-app purchases). This intensifies the fact that developers need to set a

	2012	**2013**	**2014**	**2015**	**2016**	**2017**
Free Downloads	57.331	92.876	127.704	167.054	211.313	253.914
Paid-for Downloads	6.654	9.186	11.105	12.574	13.488	14.778
Total Downloads	**63.985**	**102.062**	**138.809**	**179.628**	**224.801**	**268.692**
Free Downloads %	89.6	91.0	92.0	93.0	94.0	94.5

Table 3.2: Mobile app store downloads, Worldwide, 2012-2017 (Millions of downloads) [57]

[1] As mentioned in chapter 1 an ABI research estimates $27 billion revenue created by apps in 2013 [14]

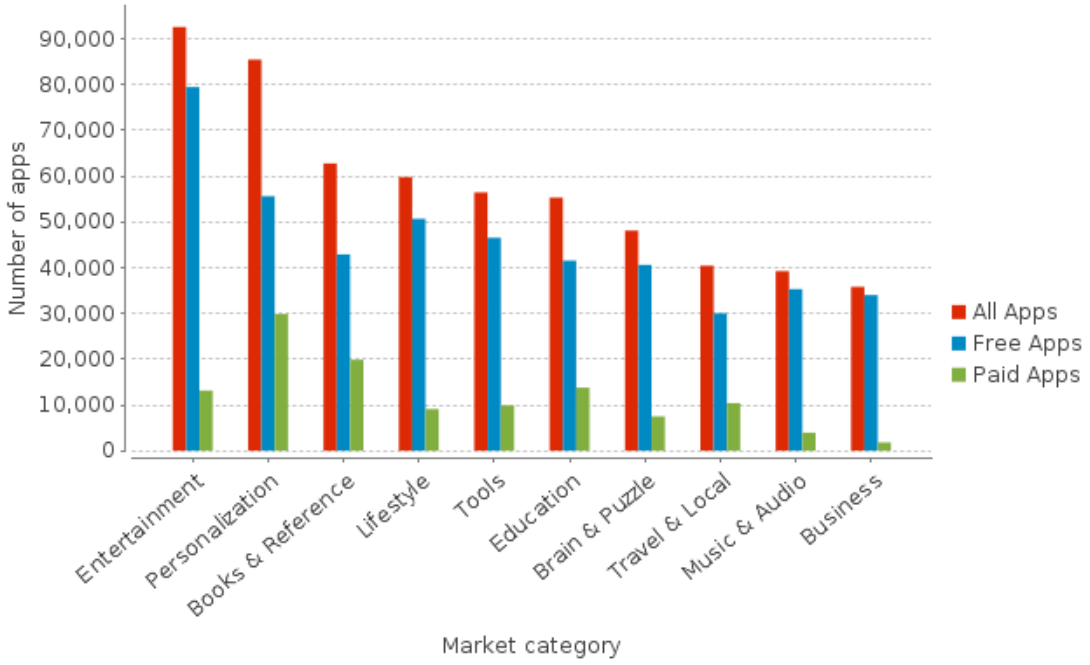

Figure 3.2: Top 10 Android market categories, October 22, 2013 [58]

focus on high quality apps to survive in this high competitive market. Table 3.2 gives an overview about the mobile app store downloads worldwide from the year 2012 to the year 2017. [57]

Another interesting statistic in displayed in figure 3.2. It shows the most popular categories in the Android market. According to AppBrain the most popular category in the Android market is Entertainment which contains more than 90.000 apps. The Entertainment category is closely followed by Personalization. Those two categories are followed by Books, Lifestyle, Tools and Education. Brain & Puzzle is the first category to feature less than 50.000 apps. The last three categories in the ranking are Travel & Local, Music & Audio and lastly Business with still more than 35.000 apps in the market.

This section of the thesis examined the mobile market in more detail. Due to the high competition in this business sector, it is very hard to succeed in this market. Therefore it is necessary to focus on high quality products to satisfy the customers using the application. A way to achieve such high quality is presented in chapter 4 of the thesis.

3.3 Characteristics of Mobile Software

With the recent advances in mobile technology mobile phones have emerged from a luxury to a necessity in our lives. There are a lot of features people use besides making phone calls, e.g. taking pictures, listen to music, using navigation, etc. Especially the ability to be connected anytime and anywhere has become essential to a majority of smartphone users. These facts have increased the demand for high quality software for mobile devices. [59, 60, 61]

The increase of popularity of tablets has a direct impact on the sales numbers of desktop computers. According to Gartner the worldwide PC shipments in the second quarter of 2013 has dropped by more than ten percent compared to the second quarter of 2012 [62]. One of the reasons for this

significant decrease is that more and more people use a tablet or a smartphone to access products or services nowadays [16].

Besides the potential the mobile market offers, there are various reasons why businesses are interested in mobile application development. Competition is one major reason why organizations need to develop their own app. A competitor on the market who offers an app for his services could draw away potential customers. An improved app with further functionality could add vital value to a business. Nevertheless it is important to focus on high quality in mobile app development. An app needs to be planned and developed in detail. Otherwise the app could bring bad reputation, which decreases the reach in the market. [63]

Taking the above mentioned facts into account it shows that mobile application development has become very important in recent years. A look on the predicted sales numbers of mobile devices for the next couple of years implies that mobile application development will become even more important in the future. In general there are three factors which characterize mobile app development: [59, p. 74]

1. "Maturity of the mobile network infrastructures"
2. "Advanced mobile hardware"
3. "Increasing demand for mobile applications/services"

As already stated in this chapter, the ability to be connected anytime and anywhere is a very crucial fact for a lot of people. This ability has only become possible with the expansion of the mobile network infrastructure. New communication protocols for mobile phones, like 3G and 4G, have increased the bandwidth for mobile devices significantly. The combination of faster network connection and the increasing usability of smartphones through the use of larger screens with a higher resolution and more powerful hardware has made using the internet via a mobile device more comfortable than ever. Developers of mobile apps have recognized this evolution and as a result innovative software is produced to fully make use of the new capabilities these devices offer. [59]

Mobile phone manufactures constantly try to improve the specs of the hardware they build into their devices. The main focus is hereby set on installing new displays with a higher resolution, increasing the computing power and making lighter devices along with a better battery life. The limitations they face is the decreased battery life when increasing processing power respectively screen resolution. Manufactures are still able to stretch those limitations and furthermore improve other features like the camera. Besides the improvement of already implemented features, new features are added to mobile devices to make them even more functional, e.g. *radio frequency identification (RFID)*. [59]

The third factor mentioned in the enumeration above is the increasing demand for mobile applications respectively services. In the first chapter of this research the reader was introduced into the enormous potential the mobile market offers. A more detailed insight into this topic was given in section 3.2 of this thesis.

With the advances in technology the mobile application landscape is evolving rapidly. Scharff and Verma state in [21] the following about the mobile application industry: "The industry follows a strict time to market requirement orchestrated by a fierce competition where standards are not stable." [21, p. 25]

To survive in the mobile market a suitable process is needed. In section 1.4, frameworks specific for the development of mobile applications were presented. These approaches are based on agile

Ideal agile characteristic	Mobile software
High environment volatility	High uncertainty, dynamic environment: Hundreds of new mobile phones published each year.
Small development teams	Majority of mobile software is developed in micro or SME companies, or development teams.
Identifiable customer	Potentially unlimited number of end-users. Business customer easier to identify, e.g. distributor.
Object-oriented development environment	E.g., Java and C++ used. Some problems in proper tooling e.g. for refactoring and test-first approach.
Non-safety critical software	Majority of existing mobile software is for entertainment purposes. Mobile terminals are not reliable.
Application level software	While mobile systems are complex and highly dependent, mobile applications can be stand-alone applications.
Small systems	Size of mobile applications vary, but generally they are less than 10000 lines of code.
Short development cycles	Development cycles vary. Generally mobile applications and services can be developed within 1-6 month time frame.

Table 3.3: Mapping agile home ground themes with mobile software development. Taken and adapted from [8]

principles, which matches with the paper [8] published by Abrahamsson in 2005 in which he considers agile methods as the best fit for mobile applications. Abrahamsson maps in his research agile home grounds with characteristics of mobile software. Table 3.3 gives an overview about the mapping. According to Abrahamsson only the agile characteristic *identifiable customer* does not match with the corresponding characteristic of mobile software, since the end-users in the distribution of mobile applications are potentially unlimited. Nevertheless there are also areas in mobile application development in which the developers are able to identify their customer, e.g. a netbanking app will only be interesting for customers of the respective bank.

In [23] Corral et al. focus on changed characteristics in mobile software compared to the characteristics Abrahamsson stated in his publication nearly a decade ago. They identify the following changes:

- Abrahamsson writes about the high uncertainty of mobile software which comes from the fact that hundreds of new mobile phones are published each year. Today the amount of new smartphones each year is still very high, but the uncertainty which comes with this fact is decreased due to the fact that three main operating systems have taken over the market, e.g. Android, iOS and Windows Phone. Those systems all have a solid development kit, which makes it easier to develop software for new devices. Nevertheless in contrast to the opinion of Corral et al. the author sees the high heterogeneity of mobile devices as a problem in mobile app development nowadays. Especially Android features different versions of their operating system, which run on all kind of different devices manufactured by different

companies. All these factors still bring an uncertainty to mobile app development which cannot be ignored.

- Regarding the team size in mobile application development not that much has changed. Mobile software is still produced by small- to mid-sized teams, but today mobile software is often part of big developments which include large companies and organizations.
- Mobile applications have become more complex over the years. This results in the fact that mobile software is not necessarily a stand-alone application any more. Mobile applications nowadays interact with other systems, use network and hardware resources more heavily, apply collaboration tools, and so forth. This makes mobile applications by definition not small any more.
- The agile characteristic for non-safety critical software collides with mobile software produced these days. According to [23, p. 103] there are applications like "healthcare monitors, mobile banking or earthquake alerts that are required to meet strict standards to enter into service and cannot be categorized as noncritical software." In general, due to the fact that mobile applications have become more and more complex the software produced for mobile devices face new requirements like safety issues.

New processes for mobile systems must therefore adapt to the changes in this field of software development.

3.4 Differences to Software Development in General

The next part of this thesis focuses on the differences between mobile app development and traditional software development for desktops or web applications. Following, characteristics of mobile software, which make it different from desktop or web-based software, get displayed: [64]

- Technological differences
  First of all, mobile hardware is different to hardware used in desktops. Manufacturers of mobile devices try to fit more and more functionality into smaller and smaller physical packages. The consequence is that storage, display and computing power are different to PCs. When developing software for mobile devices, a developer has to understand how these physical limitations change the design of the software. Another big difference is the fact that mobile devices often come with sensors respectively functionalities a desktop PC does not offer. One prominent example is the ability to track the current position via GPS and the inherent functionalities which come with this ability. Other examples are accelerometers, electronic compasses, light sensors, and so forth. A huge drawback of mobile devices is the energy consumption and the dependence on battery life. For apps which drain the energy out of the mobile device it is very likely that they get bad recommendations from users, which leads to a bad reputation of the software. Therefore developers have to be careful when dealing with the resources of the hardware.
- Differences related to usability and user experiences
  The before mentioned advances in mobile technology such as sensors and touch screens created the possibility to provide information in a more user-friendly way. Nevertheless there is a wide range of devices on the market using different display sizes or aspect ratios. This fact makes it hard for developers to optimize the software for all available products, which could lead to frustrating user experiences. Therefore developers should follow interface guidelines which allow to create applications with the same look and feel. There are already initiatives

which try to somehow standardize respectively ease the development and deployment of mobile application, but developers still have to deal with incompatible platforms for quite some time in the future. Another difference in the usage of mobile applications compared to the desktops equivalents is the limited attention span when using a mobile app. For example if a user searches information with his smartphone, he wants to have the information as fast and easy as possible. Therefore the presentation of the data has to be optimized for such situations.

- Differences in the ecosystem
  The access to software is very different from mobile devices to desktop systems. Desktop users are used to struggle with long installation requirements if they want to install a software. This is not the case for mobile users. They want to get the desired software without any inconveniences and this ideally on the go. The process is simple: The user surfs the designated app store, downloads the app and installs it - that's all. The ease of use is very important for mobile applications. Another difference to desktop computers is the lifetime and the time to market of software. The change in the mobile market is very high paced and developers have to adapt to this situation.

3.5 Approaches

There are various decisions to make before developing a mobile app. The probably most fundamental decision is which type of development approach to choose for the app. Figure 3.3 gives an overview over the three basic types of mobile applications which are native apps, web apps and hybrid apps. The basics difference between these types is the way how they access the API of the device. Native apps run directly on the device and can therefore interact very well with the hardware. The downside to native apps is that they only run on a specific operating system. Hybrid apps take a bit different approach. They run independently of the operating system and they can use basic functionalities of the device. Mobile web apps are different in the way how they are accessed. Instead of installing the app directly on the device, these apps are viewed via a browser of the specific device. The decision which type to choose also depends on various other parameters such as for example budget, time frame of the project, target audience and functionality of the app. Each of the different approaches has its own characteristics and requirements. [65]

Figure 3.3: Different approaches in app development [65]

3.5.1 Native Apps

Native applications are executable files which are downloaded and then installed on the device. The easiest way to get a desired app is to visit a specific app store such as Google's Play Store or Apple's App Store, search the app and download it. After the installation, the app can be launched as any other service. Due to the fact that the app is directly developed for this specific operating system it can access and use all the functionalities the API provides for this specific operating system. [65]

Basically there are three major development targets[2] native apps get developed for: [63]

- Apple iOS
  The operating system for apple products such as iPad and iPhone has some limitations specific when it comes to the development of software for these devices. The only way to develop an app for these devices is to use Apple's XCode which only runs on a Mac. The programming language is Objective-C. In order to start developing an iOS application a developers needs to have an apple developer account which is free. If the software should be deployed, a membership of the iOS developer program is necessary. Currently the basic program costs 99$ per year [3].

- Google Android
  Unlike iOS, Android apps can be developed on PC, Mac or Linux. The programming language is Java and for the development Google recommends the Android toolkit in combination with the IDE Eclipse. One of the advantages of Android compared to iOS is the fact that Android devices are quite reasonable priced compared to Apple's equivalents. Furthermore there are a lot of devices in different sizes available, which increases the flexibility in the development of an app for a specific market. The downside to a very open operating system like Android is that device manufactures add their own software on top of the underlying operating system, which could bring up compatibility issues within different Android devices. Developers should keep that in mind when producing software for this operating system.

- Microsoft Windows Phone
  As already stated in chapter 1, Windows Phone will solidify its position as the number three operating system behind Android and iOS. The operating system is based on the .NET framework from Microsoft. A positive aspect in the development is that in contrast to the other development targets, it is possible to use different programming languages for the development of the application, e.g. JavaScript and HTML, C#, Visual Basic, C++, XAML or DirectX. It also has limitations when it comes to the development process, because similar to iOS Microsoft Windows Phone tools only run on Windows.

Figure 3.4 shows the market share of operating systems of smartphones from the first quarter of 2012 to the first quarter of 2013. With a share of more than 60 percent Android is the undisputed leader in operating systems. Second place goes to Apple's iOS and for the first time Windows Phone is ranked as the third most used provider in the first quarter of 2013. Remarkable 92.3% off all smartphone shipments in this time frame were either featuring Google's Android or Apple's iOS. [66]

[2] The cited literature also mentions BlackBerryOS as one major development target. The author of this thesis just focuses on iOS, Android and WindowsPhone though, because BlackBerryOS has lost and will further lose market share to under 2% by 2017. [13]

[3] Detailed pricing information can be found here: https://developer.apple.com/programs/start/ios/

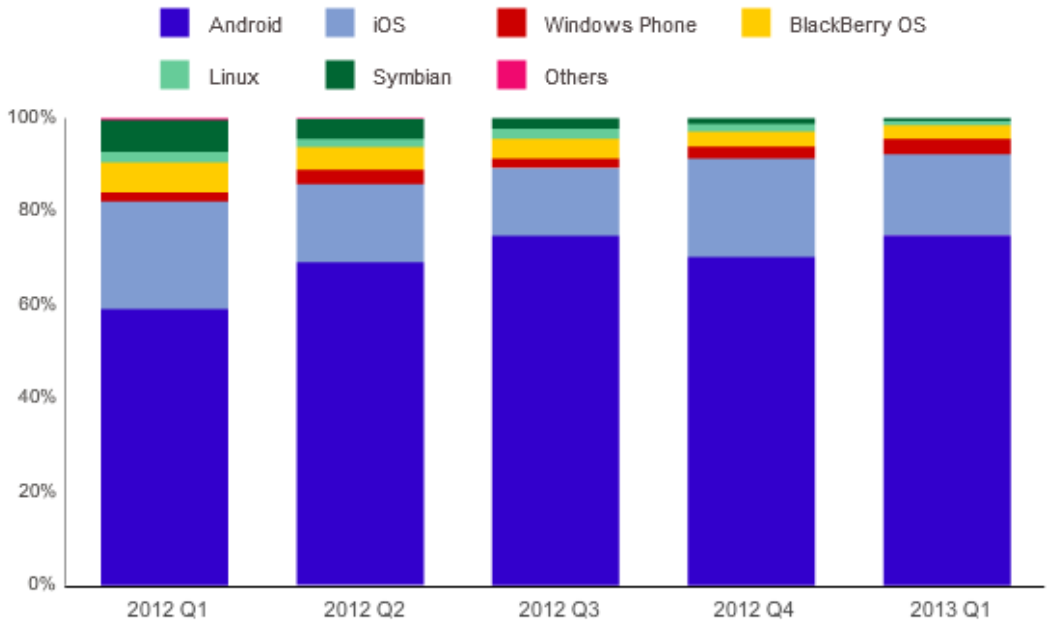

Figure 3.4: Worldwide smartphone OS share, 2012 Q1 - 2013 Q1 [66]

One of the biggest disadvantages of the native approach is the fact that it is very expensive compared to the other approaches. Mobile application development in general is not cheap, but especially the native approach brings further costs, because the development time is higher compared to cross-platform solutions, especially if the same app should be available for different operating systems. [63, 65]

Of course there are also arguments or situations where a native app is the right choice. If performance is a crucial requirement for an app, then a native app is the way to go, because the performance is better compared to the other approaches. Another advantage is the ability to use the app offline, which is not possible for web apps. Nevertheless the most important feature of native apps is the possibility to access the hardware of the device directly. The following list shows possible scenarios for the use of native apps: [65]

- Existing native skills
  The before mentioned high costs for creating native apps for different operating systems can be reduced if the knowledge of a specific operating system already exists within a company. Otherwise organizations would need to hire a new developer for this specific operating system in case they need a native app with multiplatform support.

- A single mobile OS
  In some cases a multiplatform support is not necessary. A reason for that could be the fact that an app is targeted to specific market which features just one operating system. In that case a native app is the best solution, because the costs for creating the app for different providers are non existing. Furthermore this app can be optimized for this specific operating system.

- Native functionality
  Often an app needs to access native functionality of the phone, e.g. using the device's camera, address book or media library just to name a few. Another scenario could be the requirement to use the app market as an option for payment or if the app needs high graphics and processing power. In these cases only a native app can be used.

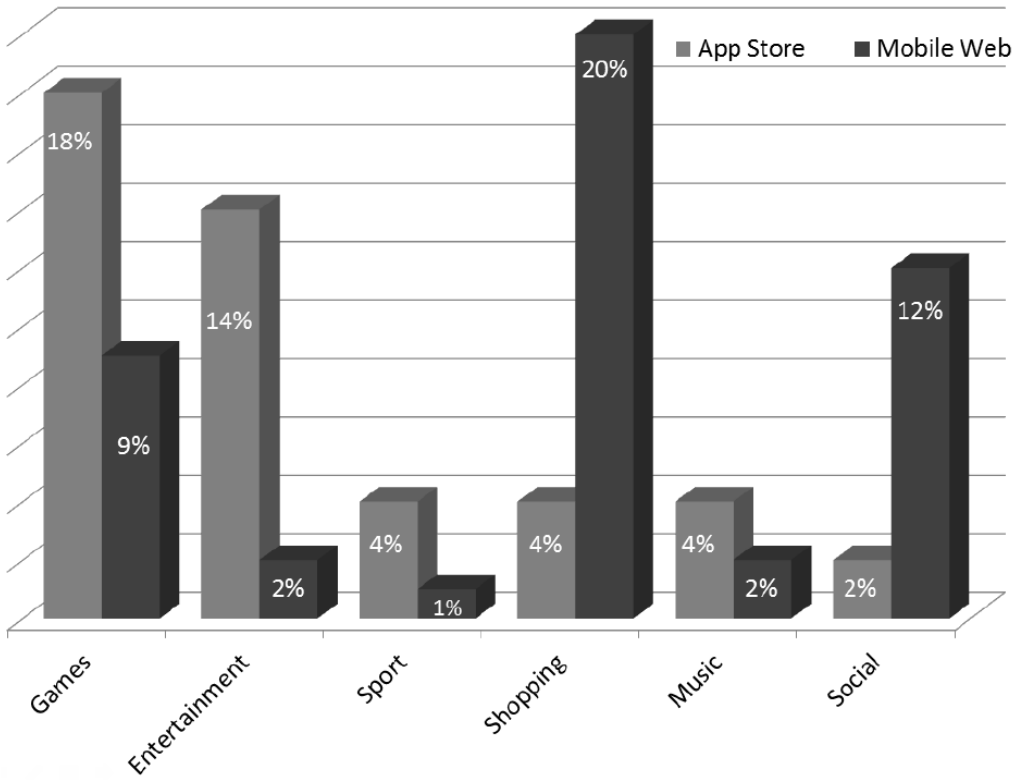

Figure 3.5: Mobile web app vs. App Store categories [63]

- Rich UI requirements
  For game-like applications which need a rich UI and real-time responsiveness a native app is also the right choice.

3.5.2 Web Apps

Mobile web apps are applications which run on a server like normal web sites and are accessed via a browser on the mobile device. Due to the use of HTML5 in combination with CSS3 and JavaScript web apps are capable of generating a comparable user experience as native applications do. Furthermore web apps can be designed to support multiple platforms, which brings a great benefit compared to native applications. With the ability to create a shortcut on the homescreen of the smartphone the user is even more encouraged to use the web app repeatedly. [67]

In *Professional Mobile Application Development* a web app is defined as follows: "Mobile web apps, in a nutshell, are mobile apps created using HTML and CSS, viewed in mobile web browsers. Mobile web apps differ from mobile websites by having a focused application purpose, like native mobile apps do." [63, p. 33]

In some business sectors mobile web apps are more popular than native applications. Figure 3.5 shows a comparison of the usage between web apps and native apps grouped by categories in which they are used. In areas where a high performance, e.g. gaming, is needed, native apps are being used more often, but in some areas like shopping web apps are the common choice for users. [63]

In [63] McWherter and Gowell present some benefits of web apps:

- Easier to get started
  A lot of developers have at least some minor experiences with HTML. This fact offers the chance to have an easy start into development, because there are no new technologies to learn. Furthermore there are no licensing costs which can arise when developing native applications.

- Easier cross-platform development
  Like already mentioned in this section, mobile web apps feature a multiplatform support, which is a big advantage.

- HTML5 offers rich features
  With the upcoming HTML5 standard, mobile web apps can make use of new features, which creates a real value to developers. By the use of HTML5 web apps can compete with native applications on a user experience level. Due to the fact that HTML5 supports offline storage, it is also possible to use web apps without a constant Internet connection.

- Easier updates
  This is probably one of the key features which distinguishes this approach to other mobile application development approaches. Due to the fact that the app is accessed via a browser the executables don't need to be installed on the user's device. This makes updates very easy, because developers can change the sources of the application directly on the server.

- No approval process
  This advantage can be directly associated to the before mentioned benefit. Since updates can be established directly on the server, developers don't have to deal with approval processes which need to be passed when publishing an app to a specific app store.

Responsive web design, a new trend which can be linked to the approach of creating mobile web apps, could increase the popularity of mobile applications which run directly on a server. The main idea behind this new approach in web design is to enhance the flexibility of a website to fit to any device. This is accomplished by the use of a flexible grid system which consists of columns with a relative width in proportion to its containing element. The grid automatically resizes as the viewport [4] changes. Another key feature of a responsive design is the use of CSS3 media queries. These media queries allow building different layouts using the same HTML document by defining conditional statements in the style sheet. When applying responsive web design techniques there are two different approaches: 1) mobile first and 2) desktop first. In each approach the designing starts at a specific reference resolution and is then adapted via the use of media queries. The newest version of the famous *Bootstrap* [68] front-end framework goes completely mobile first and follows therefore the trend of responsive web design. [69]

3.5.3 Hybrid Apps

As mentioned in the previous sections, probably the biggest downside to the native approach is the lack of multiplatform support and the high costs which come with this circumstance. Hybrid apps feature cross-platform support and overcome therefore this issue. Setiabudi and Tjahyana define a hybrid application as an application "that is written with the same technology used for websites and mobile web implementations, and that is hosted or runs inside a native container on a mobile device" [70, p. 148].

[4] The viewport is the available area for showing the web page.

Hybrid applications use a native rendering engine to present HTML and JavaScript files in full-screen format on the mobile device. Furthermore an abstraction layer needs to be implemented which exposes the device's capabilities to the hybrid application in form of a JavaScript API. The big advantage of this approach is that the native part of the application can access parts of the device's API, which makes it possibly to use functionalities of the device directly. Developers can write their own application layer or they use so called cross-platform tools for their implementation. A prominent tool for such purposes is PhoneGap. [65, 70]

The deployment of hybrid apps is very similar to the deployment of native applications which means that the app is distributed via dedicated app stores. Therefore users just need to download the app and use it like any other native application. Concerning the web portion of the hybrid app there are two possibilities on how to store the data. The first option is that the web portion is stored on a server like a normal web page. This brings the benefit that developers are able to deploy minor updates to the app without having to go through the review processes of the app stores. The disadvantage of this approach is that it eliminates the offline availability of the application. The other option is that the set of HTML, CSS, JavaScript and media files is packaged into the app and therefore stored locally on the device. This enhances the performance of the application and does not require a constant network connection. The downside to this approach is that remote updates are no longer possible. Nevertheless it is also possible to combine both approaches by hosting the HTML resources on a server to increase the flexibility and to cache them locally on the device to increase the performance and the availability. [65]

Palmieri et al. present in their article *Comparison of Cross-Platform Mobile Development Tools* different tools which help developers to create hybrid applications. Furthermore they list benefits of such tools: [61]

- Reduction of required skills
  Due to the fact that the code for the app is written just once developers don't need to learn different programming languages respectively adapt to the specific restrictions of each mobile operating system.
- Reduction of coding
  This advantage highly correlates with the above benefit, because like already mentioned the application is written only once and therefore the coding effort is less compared to the native approach.
- Reduction of development time and long term maintenance costs
  Obviously the development time in this approach is also less and furthermore the maintenance costs decrease, because there is just one application a developer has to take care of.
- Decrement of API knowledge
  Because the same application can be used for different operating systems, there is no need to grapple with different APIs of various devices.
- Greater ease of development
  This fact is a combination of the above benefits, since the development of just one application is easier than to develop an app for each operating system.
- Increment of market share
  Since the application runs on multiple platforms, there is no risk to lose potential customers who maybe would not be able to make use of the application, since their operating system is not supported.

Hybrid apps combine the best of two worlds. Due to the ability to access the device's API capabilities of the device can be used for the applications like in a native approach. Furthermore it is possible to use the same code for different operating systems, which decreases the maintenance effort, because it is a centered process and additionally the development is shorter and cost effective. Those benefits make hybrid apps very interesting to businesses in the future. According to Gartner [71] by 2016 more than 50 percent of the deployed apps will be hybrid. [65]

3.5.4 Choosing the Right Approach

In the above sections the reader got introduced into the different approaches in mobile application development. Figure 3.6 summarizes the characteristics of each approach and furthermore positions each approach regarding their platform affinity and their ability to access the device's capabilities.

As shown in previous sections, there are different scenarios for each type of mobile application. Therefore organizations need to spend time in finding out which approach is most suitable for their business needs. A wrong decision made can bring high costs to the business and can be harmful to the reputation of the organization. There is no arguing in the fact that a mobile web presence, in any form whatsoever, can bring great value to businesses respectively has already become a necessity nowadays. [63]

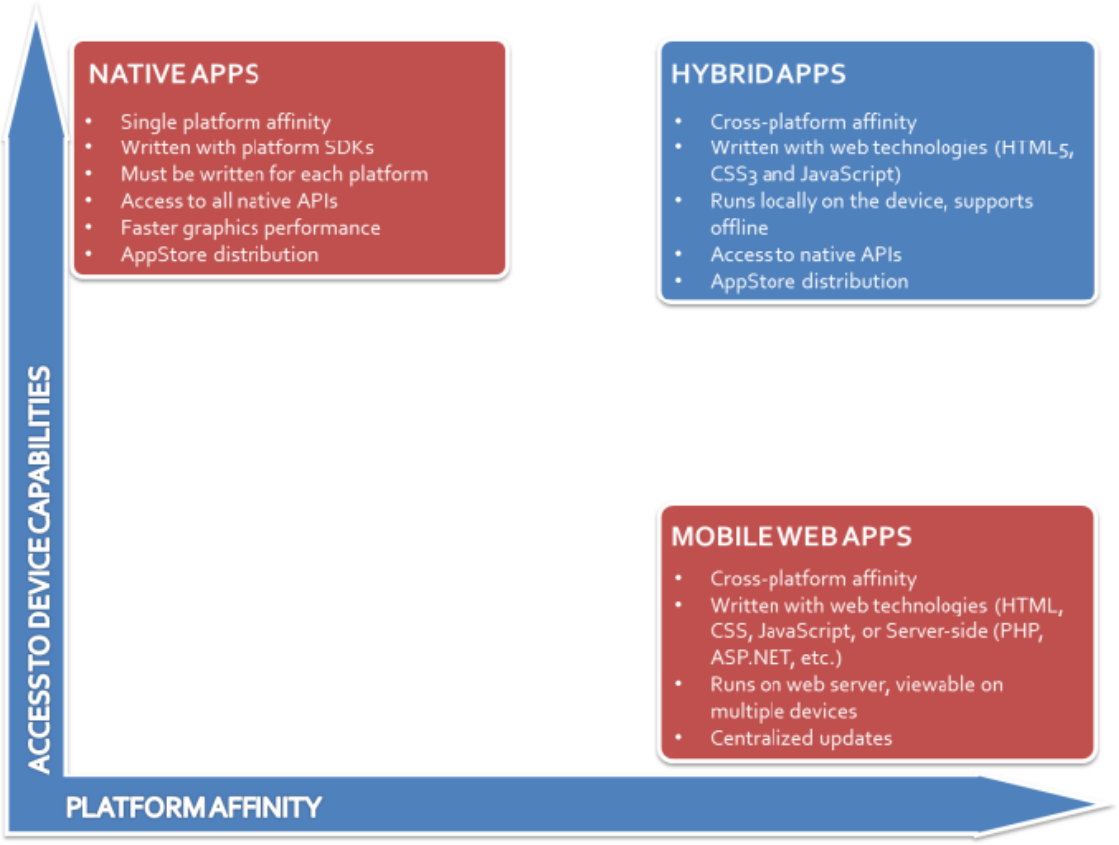

Figure 3.6: Characteristics of different types of mobile apps [72]

3.6 State of the Art

The main goal of this thesis is to find out how suitable the agile and lean software development methodologies are in the area of mobile application development. Perera and Fernando focus in [37] on the combination of those two process models in general. As stated in previous sections agile software development is very popular nowadays, but of course there are also downsides to this approach. By using lean practices Perera et al. tried to overcome possible weaknesses in the agile method. They call this approach the hybrid paradigm. In [3] Wang investigates how agile and lean approaches have been combined in software development. For his study Wang conducted 23 experience reports containing real-world experiences of those two methodologies. His conclusion was that there is no "one-type-fits-all solution" as he calls it. So businesses have to combine agile and lean software development appropriate to their needs. One goal of this thesis is to help with that challenge.

As mentioned in chapter 1, a part of this thesis is to display the differences between traditional software engineering and agile software development. Aitken and Ilango write in [15] about this exact topic. They reviewed a number of agile software development methodologies, methods and techniques against a backdrop of traditional software engineering. They found out that the two approaches are not incompatible which leads to a future possibility of an Agile Software Engineering. In [73] Trimble and Webster also focus on the differences between traditional software engineering and agile software development. In their research they document a two year process where they moved from traditional to lean and later on to agile development. For their needs the agile methodology was the best fit, but they conclude that the decision, which approach is the best, always depends on the team culture, goals and context of the work.

This thesis mainly concentrates on software project management and processes for the development of mobile applications. Gasimov et al. focus in [59] on mobile application development in general. They investigate applications on the market, present how those applications are developed and show the potential future directions of this specific field of software engineering.

Due to the rapid expansion of the mobile market new technologies have emerged. There are different ways to implement software for each of the technologies. Smutný gives in [16] insight into different mobile development tools. He also shows the strengths and weaknesses of native apps and cross-platform solutions. Sin et al. concentrate their research in [67] on the topic of mobile web apps. Trends have shown that web apps have become very popular, due to the fact that they are platform independent and therefore the resources needed in the development are less compared to the development of native applications. Mohorovičić bases his research in [69] on the new trend of responsive web design.

Abrahamsson focuses in [74] on the applicability of agile software development in the area of mobile information systems. Due to the rapid growth of the mobile market and the constant advance of technologies in this field, the mobile application market is highly competitive, uncertain and dynamic. Restrictions in screen size, memory and battery power are not that big any more and the processing power of smartphones and tablets is already as good as the power of regular computers. So the limitations in the mobile market are getting smaller, which opens the door for new advances in mobile application development.

In [75] Wasserman writes about software engineering issues related to the development of mobile applications. For that he analyzes the characteristics of mobile applications in general and further raises possible research topics related to mobile app development. He concludes that there are still a large number of complex issues in this area of software engineering which need to be addressed in further work.

4 ALP-Mobile

This chapter presents the main theoretical part of this thesis. Based on extensive literature research in the field of mobile software development and its processes the author proposes a process model called ALP-Mobile (**A**gile and **L**ean **P**rocess for **Mobile** Application Development) which is a combination of agile and lean methodologies to overcome the challenges in mobile application development.

Some of these challenges are presented in [76]. Conder and Darcey list the following hurdles which a mobile development team needs to overcome: [76, p. 551]

- "Choosing an appropriate software methodology for your mobile project"
- "Understanding how target devices dictate the functionality of your application"
- "Performing thorough, accurate, and ongoing feasibility analyses"
- "Mitigating the risks associated with preproduction devices"
- "Keeping track of device functionality through configuration management"
- "Designing a responsive, stable application on a memory restrictive system"
- "Designing user interfaces for a variety of devices with different user experiences"
- "Testing the application thoroughly on the target devices"
- "Incorporating third-party requirements that affect where you can sell your application"
- "Deploying and maintaining a mobile application"

In [18] Abrahamsson et al. tried to overcome the challenges in mobile application development and therefore they proposed Mobile-D, an agile development approach which is a combination of Extreme Programming, Crystal and the Rational Unified Process. It consists out of the following phases: 1) Explore, 2) Initialize, 3) Productionize, 4) Stabilize and 5) System Fix and was proposed in 2004. Figure 4.1 illustrates the approach.

Abrahamsson et al. state further: "The Mobile-D approach is optimized for a team of less than ten developers working in a co-located office space aiming at delivering a fully functional mobile application in a short time frame." [18, p. 175]

The idea of Mobile-D sounds very promising, but as stated in the previous chapter of the thesis, characteristics of mobile software have changed over the years. Mobile software must not be seen as simple stand-alone applications nowadays, but rather as complex software systems which

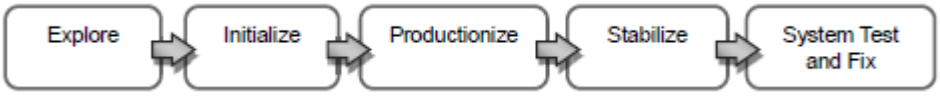

Figure 4.1: Phases of the Mobile-D software development process [23]

need standardized processes in order to be successful. Through the combination of agile and lean methodologies ALP-Mobile can bring great benefit to organizations working in this field of software development.

4.1 A Basic Overview

Hibbs et al. write in their book *The Art of Lean Software Development* [38] about practices which can be found in agile and lean software development. Following these practices are listed and mapped to mobile application development:

- Automated Testing
  Testing is a fundamental concept of lean software development. Due to ongoing tests during the complete project life cycle waste can be reduced because the necessary rework is eliminated and therefore the quality of the software can be increased. In agile software development testing is at least as fundamental as in lean software development. Through methodologies like *test driven development* the focus on testing got increased even further. Automated testing includes various types of testing, e.g. performance testing, unit testing, acceptance testing, etc. Concerning lean software development in particular automated testing supports three lean principles which are 1) Eliminate waste, 2) Build quality in and 3) Keep getting better.

  The mapping of automated testing to mobile application development is not easy. There are still issues related to testing on mobile devices. In [75] Wasserman points out that testing is an important area for mobile software engineering research. He advises that it is insufficient to test apps only on emulators. Apps needs to be tested across different versions of operating systems. Satoh comes up with another issue regarding testing on mobile devices. In [77] he states that tasks involved in mobile testing "are often tedious and susceptible to errors, because changes in network connectivity and locations may lead to sudden and unpredictable changes in contextual information." [77, p. 1112]

- Continuous Integration
  Integration in software development is where all components involved in the development come together and get tested against the underlying architecture. In traditional software engineering integration was seen as a separated phase after the development of the product. Due to the late integration in the project life cycle many issues came up late and therefore the rework tasks increased. Through continuous integration this phase is directly put into the development and can be seen as an ongoing activity during the implementation of the software. The idea behind this practice is that the application grows incrementally by integrating small changes into the product. As a result an extra integration phase can be eliminated.

  In [78] Stolberg describes the typical chain of events in a continuous integration framework as follows. First of all a source code repository with an automated change detection is needed. This detection recognizes the fact when a team member checks in new code into the repository. After the detection the new source code is built on a dedicated build server. Following unit tests are executed, acceptance tests are performed and the software is made available to all team members on a public server.

  This practice is very closely coupled to automated testing, but the difference hereby is that the application is tested on every check in of all team members. Additionally to the presented obstacles developers face the problem that testing tools often do not have interfaces to test mobile specific functionalities and sensors like GPS or rotation [79].

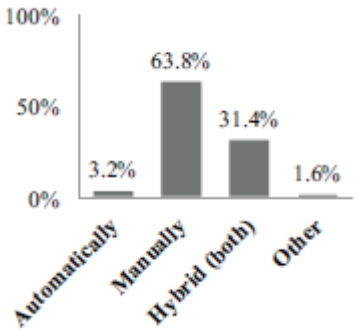

Figure 4.2: Testing in mobile application development [79]

In [79] Joorabchi et al. conducted a survey of 188 respondents in the field of native mobile development. One of their questions focused on the testing behavior of the respondents. The result illustrated in figure 4.2 displays the current situation in mobile application testing. Only 3.2 percent of the people test their applications fully automatically. With a percentage of more than 63% teams test their apps only manually.

- Less Code
  The third lean practice presented in the book concerns the codebase itself. Hibbs et al. recommend to focus on writing software with fewer lines of code. This can be done through eliminating unnecessary code and making necessary code more efficient. They point out that a larger codebase brings the following effects:
  - More code typically means more components, which brings more integration effort.
  - Through the larger codebase the chances for errors and bugs are increased.
  - The increase in lines of code makes it harder to understand the code for new developers.
  - Due to the fact that more code is written, the agility in the response to change is decreased.

  Concerning lean thinking Hibbs et al. state: "The definition of waste in Lean development is anything that increases costs without providing value, so the costs resulting from developing and maintaining unnecessarily large codebases can be viewed as waste." [38, p. 72]

  This practice can be mapped to mobile application development without further adjustments. Like many lean principles this one functions as a guideline which helps developers to focus on essential tasks during the development.

- Short Iterations
  Short iterations are an important practice in lean and in agile software development. With the iterative development functional software is provided to the customer on specific times during the development process, e.g. in scrum at the end of every sprint. As pointed out in the introduction to the scrum framework, the short iterations with the creation of a possible shippable product help customers to understand how the final application will fulfill its needs. Additionally through the high collaboration with the customer including customer feedback the software is developed under supervision of the customer, which brings the advantage that the development team can early adapt to changing requirements.

  This practice especially supports the lean principle *deliver fast*. By working in the feedback of the customer on every iteration the quality of the product is increased incrementally.

Furthermore the vision of the customer is met early in the development, which reduces the rework tasks at the end of the development.

Short iterations are a key success factor in mobile application development. In section 3.3 the author examined the characteristics of mobile software. Concerning the time frame of mobile app projects it was stated that project in this field of software engineering are usually developed in one to six month. Scrum as an example defines iterations for sprints from two to four weeks. Based on the short project lifecycle, those proposed iterations limit development teams in their flexibility. Therefore processes in mobile application development must adapt to the decreased time to market of mobile apps compared to software development in general.

- Customer Participation
  In case the software is produced for a customer, active customer participation is an integral part of the software development process. As mentioned in the previous practice, customer feedback helps to gain essential information about the product. Hibbs et al. state about customers involved in a software project: "They know the tasks that have to be done, they know the conditions in which the solution must perform, and they know the goals they are trying to achieve with the solution." [38, p. 96]

  In traditional software engineering customer feedback was gathered in course of the requirements analysis phase and after the implementation of the product, where change requests were generated. In lean software development customers are not only part of the development process but they also write and prioritize requirements of the application. Furthermore they help to define the acceptance tests and decide when specific requirements are met. Another important fact in the customer participation is that customers are available to answer questions regarding the development and to resolve ambiguities. Furthermore customers should be informed about the current progress and about goals of each iteration. Early problem reporting also helps to increase the trust relationship between the team and the customer.

  In scrum, customer participation is performed via the role of the product owner. The product owner represents the needs and expectations of the customer during the development phase and functions as a point of contact for the development team.

  Customer participation is a very important part of mobile application development because typically this kind of software development faces a lot of decisions the customer needs to make, e.g. design decisions. Therefore organizations need to integrate the customer into their development cycles. This is usually achieved through the usage of dedicated review meetings which the customer attends. In cases the customer is not available in person, the development team should provide the customer with a current version of the app. This way valuable feedback is gathered.

  When it comes to feedback from the customer in mobile application development early prototypes of the software are very advisable. The use of mock ups helps to illustrate the functionalities of the future app even before the actual implementation started. Through the use of this technique the customer can make change requests very early in the project lifecycle.

  Especially in mobile applications the end user of the application should be in the focus of the development team. It is not sufficient to test applications in a laboratory setting, but rather in real word scenarios. User acceptance tests are a possible solution for this issue. Through the ability of end users to rate apps in the dedicated app stores the importance of user satisfaction is increased in this field of software development.

ALP-Mobile is designed to implement the above practices as well as possible. This is achieved by the use of practices and principles from scrum, XP and kanban.

4.2 Roles in ALP-Mobile

ALP-Mobile utilizes the role of the product owner as well as the role of the development team presented for the scrum framework. Additionally the author introduces the role of an agile coach in ALP-Mobile. In the following section an insight into the functionalities of each role is given:

4.2.1 Product Owner

Rubin defines the product owner as follows: "The product owner is the empowered central point of product leadership." [44, p. 165] A product owner needs to look in two directions simultaneously. The first direction concerns external stakeholders like customers, users and organizational stakeholders. The product owner needs to understand the wishes and the demands of those stakeholders in order to ensure the development of the desired product. The other direction concerns the development team with which he communicates to make sure that the product is implemented properly. This is for example achieved through the determination of high level tests which ensure the functionality of the product. [44]

To make sure that the product is developed as expected high collaboration with both the development team and external stakeholders is needed. This close collaboration must not be underestimated. It is an ongoing tasks which brings necessary feedback to the development team. In traditional software engineering collaboration with external stakeholders is very high in the beginning of the project but decreases early after the definition of the requirements to almost nonexistent engagement of the customer. Only at the very end of the project the collaboration with the customer increases again which brings almost no feedback to the development team during implementation [44]. Such situations should be avoided.

In agile and lean approaches this issue is addressed through high customer collaboration during the development of the software. High customer collaboration is especially important in mobile application development, since there is a huge focus on usability and design in mobile software. In ALP-Mobile the product owner is responsible to keep up the collaboration with the stakeholders and to ensure the engagement of the customer during the whole project lifecyle.

Additionally to the already mentioned responsibilities the product owner needs to have good interpersonal skills to be successful in this role. A good relationship to all stakeholders in the project is the essential basis for the tasks a product owner needs to perform. With a lot of people involved in the project conflicts will come up. Therefore a product owner needs to be a negotiator as well as a good decision maker. This implies the importance that this role is empowered to make decisions in the project. A product owner needs to find the right balance between business needs and technical realities. [44]

4.2.2 Development Team

"Traditional software development approaches define various job types, such as architect, programmer, tester, database administrator, UI designer, and so on. Scrum defines the role of development team, which is simply a cross-functional collection of these types of people." [44, p. 195] The quote stands for the ability of the team to be able to deliver business value to the customer. Especially in traditional software engineering teams are grouped by their job types, e.g. one team of

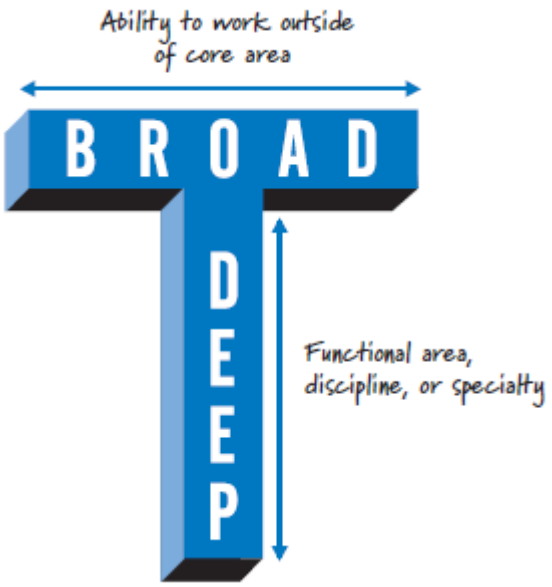

Figure 4.3: T-shaped skills [44]

developers and one team of testers. The development team in scrum should be able to provide all necessary skills without having external dependencies to other teams. In mobile application development such a development team usually consists out of the following types: designer, developer and tester. Those types of course vary and they highly depend on the size and the complexity of the project. ALP-Mobile requires a team that is capable of meeting the requirements which are made to the software, but in this point the model is not very prescriptive.

Agile and lean software development puts strong focus on self-organizing teams. Self-organization is nothing that should be forced by the management, teams should become self-organized in a bottom-up approach. Stober and Hansmann state about organizations which follow such a bottom-up approach: "Successful organizations will become highly dynamic organisms that are built from independent, self-organizing units that collaborate towards a common set of goals." [40, p. 8] Self-organization is also one of the core characteristics the development team in ALP-Mobile should have.

Another important aspect of this role is that the development team should be diverse in their skills. In detail this means that the team should for example not only consist out of developers who then need to do testing or perform design tasks. The development team in ALP-Mobile should be able to produce software in the best quality possible and therefore different types of people with different backgrounds need to be in the team. Rubin refers to that as "Cross-functionally diverse teams" [44, p. 200]

Concerning the people working in the team directly they should have so called *T-shaped skills* in order to have as flexible teams as possible. The term T-shaped skill comes from the fact that a person has deep understanding in a specific area, but also a broad understanding in other areas. Such skills help to overcome bottlenecks in a team by helping other team members with their tasks [44]. Figure 4.3 illustrates this ability.

4.2.3 Agile Coach

In [80] *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition* Adkins states the following about agile coaching: "Agile itself is sufficient;

coaching deepens it." [80, p. 5] and further "Agile is easy to get going yet hard to do well" [80, p. 6]. These two quotes perfectly sum up the necessity of a coach when applying agile or lean concepts in an organization. It is often the case that organizations want to switch from traditional software engineering to agile or lean methodologies. But for example just using scrum in software development does not make the organization agile. Agile and lean principles must be applied in a proper way to really profit from the benefits those approaches offer. The author therefore introduces the role of the agile coach in this process model.

Adkins defines an agile coach as the following: [80, p. 176] "An agile coach is a...

- **Bulldozer**: Helps the team bulldoze impediments to get them out of the way
- **Shepherd**: Guides the team back to agile practices and principles when they stray
- **Servant leader**: Serves the team rather than the team serving you
- **Guardian of quality and performance**: Examines both what the team produces and how they produce it to offer observations that help them tune the human system they are"

The above definition expresses the tasks the agile coach should perform. Scrum and XP both have very similar coaching roles. In scrum there is the scrum master [44] and in XP there is the so called coach [81]. In ALP-Mobile those roles are replaced by a more generic role which is not directly targeted at either the scrum framework or XP. In a nutshell, the agile coach helps everyone involved in the process to understand the values, principles and practices applied in ALP-Mobile. Therefore this role cannot be reduced to a scrum master who only concentrates on scrum itself. ALP-Mobile needs a coach for XP, scrum and kanban.

In more detail the agile coach supports the development team and the product owner. Through coaching both other roles in ALP-Mobile, the agile coach can remove barriers between those roles. This has the affect that the product owner does not have to face obstacles when managing the development.

The agile coach in ALP-Mobile can be compared to a coach of a sports team. Through the observation of the team performing in ALP-Mobile, the coach is able to identify possible weaknesses in the process. By eliminating those weaknesses the performance of the whole team can be lifted to another level. The agile coach achieves this by helping people in the team to solve their own problems rather than solving the problems for them. Only in cases that problems can not be solved by an individual, the agile coach actively participates in the problem solving process. Additionally to the above mentioned activities, the agile coach helps new members to integrate smoothly into the team. As an example a new product owner is supported by the agile coach in performing his responsibilities as a product owner.

Concerning the process itself the agile coach focuses on the maximization of the delivered business value. Note that the agile coach does not have the authority of a typical project manager. Instead of taking action by himself, the agile coach helps all team members in the process of making decisions on their own.

4.3 Communication in ALP-Mobile

In this section the author presents techniques and tasks to ensure good communication in ALP-Mobile. Therefore three important concepts are presented which are 1) the kanban board, 2) the work item card and 3) meetings. The thesis focuses on those concepts below.

4.3.1 Kanban board

In the second chapter of this thesis the author presented the concept of kanban to the reader. As a reminder the following overview states again the three main constituents of kanban: [46]

- Visualize the workflow
- Limit Work In Progress (WIP)
- Measure the lead time

The kanban board functions as the main source of information in ALP-Mobile. With the help of figure 4.4 the author explains the main concept of the kanban board. Due to the use of kanban as a tool for the process execution the usage of fixed iterations like sprints in scrum is not prescribed in ALP-Mobile. Instead the work in progress is directly limited via the workflow states in the kanban board.

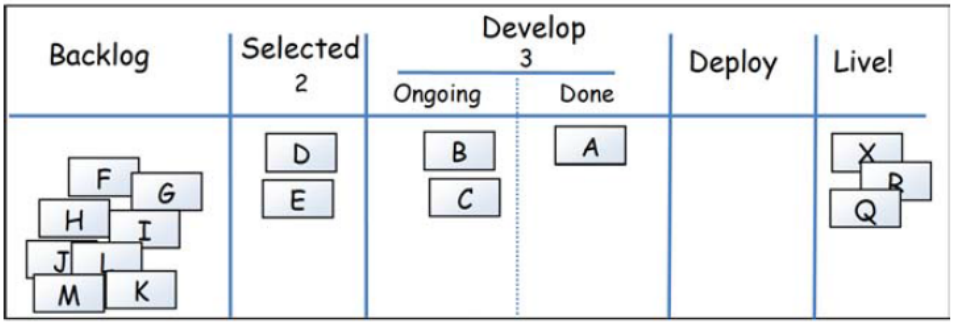

Figure 4.4: Kanban board example [46]

On the very left hand side of the kanban board in figure 4.4 there is a column Backlog which functions as a pool for features to be implemented in the future product. Please note that this column does not have a number below the label which indicates the maximum number of items in this state. This means that the Backlog column can be filled with cards without restrictions. The Backlog column can be compared to the product backlog in scrum. Section 4.4.1 goes into more detail about creating the work items.

The columns Backlog and Selected in this example function as so called *queues* in the workflow. The state Selected has in contrast to the Backlog state a WIP limit of two. This means that at any given state in the process there must not be more than two items in this column. Like scrum, kanban is a pull system. This means that every time a specific state has not reached its WIP limit a member of the team can pull an item from the previous stage. The product owner can always change items in both the Backlog as well as the Selected column, but he is not empowered to make changes to other columns than Backlog and Selected. Furthermore he is responsible for keeping those lists up to date. The possibility to make changes to those columns makes ALP-Mobile very agile and flexible, because there are no restrictions like in scrum where the product owner has to wait for the next iteration to end to make adjustments.

The next stage in process is the Develop stage. This stage shows the ability to divide a specific column into further stages. In this case there is an Ongoing and a Done stage, indicating the current progress of the work item. This makes it easier to see which items are ready for the next stage in the workflow. With a WIP limit of three the Develop stage already has reached its limit, but work items B and C are still in the state of ongoing work. In case B and C are also finished

they can be pulled to the Done column of the Develop stage. Since this example does not have a limit in the Deploy stage, those items can all be pulled without restrictions. In situations where the next stage is already at its limit the flow in the process stops, because the next stage is blocking the previous one. Such situations are not ideal, but they occur. To solve this problem as fast as possible the people working in the previous stage will help people in the next stage to finish their work. This way the flow does not stop for a long time and the next items can be pulled.

The kanban board is the perfect tool to support the three main constituents of kanban. It visualizes the workflow, the WIP limit is also visible and directly attached to each stage and additionally the lead time can easily be measured by the time one item needs for the whole workflow until the Done stage. The average lead time is updated with each new item in the Done stage.

4.3.2 Work Item Card

In the previous section the author presented the kanban board as the main source of information in ALP-Mobile. One very important part of this board is the used card to denote a specific work item. In *Kanban: Successful Evolutionary Change for Your Technology Business* [82] Anderson presents the information such a card should provide:

- ID
  The ID is used to uniquely identify this task. Especially in a distributed environment the ID is essential, because with the help of this ID an electronic tracking system can be used to support the process. Usually this number is written in the top left corner.
- Title
  The title of the working item is written in the middle of the card.
- Date of entry
  The date when the working item was pulled into the process is noted in the bottom left corner of the card. In cases the work items get pulled in a first-in, first-out principle this date is helpful. Furthermore it serves for the calculation of the lead time.
- Assigned person
  Above the ticket the name of the currently assigned person is noted. Since the ticket is in a steady flow it would make no sense to write the name of this person on the ticket.

4.3.3 Meetings

This section of the thesis discusses the meetings which are prescribed in ALP-Mobile.

Kick-Off

The kick-off meeting takes place before every other meeting in the project. The reason for the kick-off meeting is to distribute the information about the upcoming project within the team. In a distributed environment all people working on this project should be present if possible. Concerning the roles in ALP-Mobile every role in the model should be present in order to get a common understanding of the project. The product owner leads this meeting since he in possession of all necessary information for this meeting. This information was collected through prior communication (e.g. another meeting) with the customer.

The following list gives an overview over the activities discussed in the kick-off:

- Project overview
  This activity is dedicated to the introduction of the team to the upcoming project. This contains for example information regarding the content, the importance as well as the customer.
- Project schedule
  The schedule and the time frame of the project is presented.
- Communication
  The communication channels used in this project are determined. Especially in a distributed environment this activity is essential to ensure successful working and communication over the complete project lifecycle.
- Determination of the technology stack
  The team decides which technologies get used in this project. This contains the questions about the development approach of the mobile application. This specific decision is highly affected by the customer.
- Questions and answers session
  Open questions from the team members are discussed.

After the kick-off meeting every team member should know the essential information about the project.

Daily Stand Ups

Daily stand up meetings are a common practice in agile processes, e.g. daily scrums. Typically they are performed in the morning before the actual work starts. As stated in the description of the scrum framework, the daily scrum consists out of three questions which are answered by all team members. Those questions are: 1) What did you do since the last team meeting? 2) What obstacles are you encountering? 3) What do you plan to accomplish by the next team meeting?

In ALP-Mobile daily stand ups are performed differently. Nevertheless they should not take longer than fifteen minutes, which may need discipline in the beginning. Through the use of the kanban board all necessary information is available and therefore there is no need to ask the three questions advised in daily scrums. Instead ALP-Mobile follows the way Anderson suggests to perform daily stand ups [82].

The focus is set directly on the flow of work. Attendees of the daily meetings will see the changes by looking at the kanban board. Additionally bottlenecks in the process become easily visible. Anderson states that a facilitator will *walk the board*. In ALP-Mobile this task is usually executed by the agile coach. In cases the agile coach is not available, a member of the development team will function as the facilitator instead. Walking the board means to go in reverse direction to the flow (in the kanban board in figure 4.4 this means from right to left) over the board and to investigate each working card. The facilitator then may request a status update regarding this ticket or simply ask for additional information which is not visible on the board and which should be shared among the team.

Special focus is set on items which are stuck or which have not been moved for a couple of days. The information for not being moved longer than a day could be displayed with little dots for each day without moving next to the card. As the team becomes more self-organized and experienced in their work the facilitator will not have to go through every single card on the board, but rather focus only on the cards or items which are stuck or which cause problems.

In a distributed environment with overlapping working hours the daily stand ups should be held together via video conference to ensure that every member on the team has the same information about the project and the ongoing progress. If this is not possible then at least the main information gathered in this meeting should be communicated. The attendance of the product owner is not prescribed by ALP-Mobile. Nevertheless it is possible that the product owner joins the daily stand up.

Planning Meeting

In ALP-Mobile the features to be implemented are collected in a queue on the kanban board. In the example kanban board in figure 4.4 the Backlog stage serves this functionality. The product owner keeps this list up to date and ranks those items according to their importance. This ranking is based on the effort estimation which is performed during the planning meeting. This meeting takes place in the beginning of the project and every time new features need to be estimated. The meeting requires the role of the development team as well as the role of the product owner to participate. The agile coach should attend this meeting in the beginning of the project in order to make sure that everything is executed as expected. It is the responsibility of the product owner to wait for an appropriate time in the process to establish a new planning meeting. ALP-Mobile does not prescribe a fixed iteration for this kind of meeting. Nevertheless the retrospective meeting, which the author describes in the next section of the thesis, facilitates a possible chance for another iteration, since the necessary roles are already available and therefore no extra meeting needs to be scheduled.

A prerequisite for this meeting is that the product owner has a list of user stories[1] which need to be estimated. Once this prerequisite is accomplished the actual planning can start. For that the use of story points is advised. In [83] Cohn defines user points as follows: “Story points are a relative measure of the size of a user story. A user story estimated as ten story points is twice as big, complex, or risky as a story estimated as five story points. A ten-point story is similarly half as big, complex, or risky as a twenty-point story. What matters are the relative values assigned to different stories.” [83, p. 40]

Since story points are a relative measure there is the need for a specific user story to refer to. In [83] Cohn presents two approaches for this situation:

- One approach is to select the smallest expected story in the list. This story is then estimated with one story point.
- The other approach is to select a medium-sized user story and then estimate this story with story points somewhere in the middle of the estimation scale.

This user story is used as a relative measure in order to estimate all other stories. Concerning the estimation scale, there are various possibilities. Cohn proposes the use of the Fibonacci sequence as an example for this scale. The advantage of this sequence is that the gaps between two estimations become larger as the sequence increases. This expresses the uncertainty of the estimation of large user stories. After estimating the first user story, the estimation process can continue. For this the author recommends the usage of a practice called *planning poker*. This process is executed in the following way [84]: the product owner explains each user story to the development team. After the explanation the development team discusses the user story to a point that every developer has enough information to estimate the effort related to this user story. Then each developer

[1] See section 4.4.1 for more information about user stories.

writes the expected user points for this user story on a piece of paper which then simultaneously is presented to the other members in the team. Note that this estimation is a relative measure of the user story estimated prior to the planning poker process. In the next step the developers with the highest respectively lowest estimation clarify their decision. Eventually using more estimation rounds a commonly accepted decision within the team is made and a specific estimation is set for this specific user story. This process is continued until every user story is estimated.

Concerning the kanban board and the related work item card the author suggests to use the ID of the card to refer to the user story. This way the card does not contain too much information and the user story can be stored electronically. Furthermore the estimation of the user card should be noted on the card to directly see the effort this card will take. This is important, because the estimation of user stories brings changes to the WIP limit used in the kanban board. Kniberg and Skarin proposed two different possibilities to handle such situations: [46]

- Estimations vary a lot
  In cases the estimations of the user stories vary a lot, the WIP limit should be updated to the use of story points. This means that for example the Deploy stage has a WIP limit of 15 user points, which makes it possible to have three user stories with each 5 user points on this stage. Every other combination is acceptable as well, as long as the total number of story points in this stage does not exceed 15.

- User stories have almost the same effort
  In such cases the product owner is responsible for making user stories with roughly the same size. Therefore the standard WIP limit of work items in a specific stage is still functional. Since the understanding of user stories may vary between the product owner and the development team, a planning meeting is still necessary in such cases, because the defined user stories should be discussed with the development team. It is important that the product owner integrates the development team into this process in order to avoid disagreements between those two roles.

Retrospectives

ALP-Mobile prescribes the use of retrospective meetings. Those meetings help to reflect the work done and give the possibility to bring change into the team and the process. Derby and Larsen state in [85, Preface, p. xii] about those meetings: “A special meeting where the team gathers after completing an increment of work to inspect and adapt their methods and teamwork. Retrospectives enable whole-team learning, act as catalysts for change, and generate action.”

Scrum prescribes two different meetings regarding retrospectives. On the one hand there are sprint reviews in which the product itself is inspected and on the other hand there are sprint retrospectives in which the process itself is reviewed. In ALP-Mobile those two meetings get combined to one single meeting. The iteration for this meeting is advised to be every week. All roles in ALP-Mobile should attend this meeting. In a distributed environment it is not easy to come together in such short iterations. It would be the optimal solution, but it is also possible to make the retrospective with the help of a video conference, or at least to make a protocol for members of the team who are not attending.

This meeting is basically split into two parts which the author examines in detail below:

- Review Part
  The review part of the retrospective is the reason for the short iterations of one week prescribed in ALP-Mobile. This part of the retrospective gives the whole team the chance to

inspect the developed product and furthermore to get feedback from external stakeholders, e.g. customer.

The feedback from the customer is either received directly in case the customer attends this meeting or indirectly in case the customer did not get invited to this meeting and therefore the product is made available to the customer through the use of dedicated tools or in case of Android development simple by sending the apk file to the customer. Reasons for not inviting external stakeholders to every retrospective meeting are simple, since such meetings can often present information or bring up issues which should not be discussed in front of those stakeholders.

In cases external stakeholders attend the meeting the product owner will start the meeting with some introductory words. After that, the agile coach leads the meeting. In essence this part of the meeting is dedicated to present the current state of the project. Note that on a weekly basis the focus is set on value and not on presentation. That means that there is no need for a complex presentation. It sufficient to present a live demonstration of the product. Through instant feedback of all attendees the team gathers valuable information for further steps in the process. In total this part of the meeting should not take longer than 60 minutes.

- Retrospective Part
  The second part of this meeting should be executed every two weeks. Nevertheless especially in the beginning of the usage of ALP-Mobile a more frequent iteration is maybe advisable, e.g. weekly after the review part of the meeting. ALP-Mobile follows the structure for the retrospective suggested in [85]:

  1. Set the Stage
     The first part in the retrospective is opened by the agile coach who helps the people attending this meeting to focus on the goals the group has for the retrospective meeting. It is important to clarify the time frame of the meeting so team members have insight in the schedule. After that, every person should express their hope for this retrospective in a few words. This helps to involve everybody in the meeting. Additionally the coach presents the approach for this meeting, so that people know what to expect during the meeting.
  2. Gathering Data
     In this activity data is gathered in order to create a shared picture about the past working cycle. This is important because different people have different perspectives of things that happened. Additionally some team members may have missed parts of the past working cycle. The kanban board helps to get those data together. Additionally to hard facts like metrics or events, feelings of the team members should be collected. They give important insights in the process and how the team works together.
  3. Generate Insights
     After collecting the data, the data is used to detect strengths and weaknesses respectively issues in the project. This phase is dedicated to showing *why* things happened in the working cycle. Through those insights it is possible to determine how to work more effectively. A possible example for such an insight could be the fact that WIP limit in the kanban board was not optimal and therefore the flow of the system was disturbed.
  4. Decide What to Do
     In the previous stages of the retrospective ideas about change in the process may came up. In this phase the team decides what to change in the process in order to be more productive. It is important to keep the changes to a minimum for a specific iteration. Otherwise the changes will not be as effective as they could be. By taking just one

Activity	Time in percent
Set the stage	5%
Gather data	30–50%
Generate insights	20-30%
Decide what to do	15-20%
Close the retrospective	10%
Shuffle time	10–15%

Table 4.1: Distribution of time per activity [85]

or two ideas to implement, the team members will focus on them for the upcoming iteration.

5. Close the Retrospective
   The coach closes the retrospective in the last phase of the meeting. People should know how to move on from this point on. Additionally it is important to document the necessary information the meeting brought up.

For planning a retrospective it is vital to schedule the meeting in order to keep the time frame. Table 4.1 shows a possible distribution of the time. Note that 10-15% of the time is dedicated to the so called shuffle time. This time is needed to move from one activity to the other and additionally it functions as a buffer. Again this part of the meeting should also not take longer than 60 minutes in total.

4.4 Phases in ALP-Mobile

This section of the thesis focuses on the phases in ALP-Mobile. Looking at the bigger picture the process is divided into three main phases. Those phases are based on the proposed model in [64] and are referred to as the 3 D's of ALP-Mobile, which are 1) Definition, 2) Development and 3) Distribution. The author presents each of the phases in detail below.

4.4.1 Definition Phase

The first phase of the ALP-Mobile process is dedicated to activities which are related to the definition of the final product and to the analysis of the upcoming software project. It is very similar to related phases in software development in general. After the completion of this phase, the development team should be able to start with the actual implementation of the software.

In some cases this first phase of process will reveal issues regarding the success of the complete project. In such cases a decision has to be made if this project is worth continuing or not. It is not a shame to realize failure and it is definitely the better solution to end a project early in the life cycle than to a later moment when the costs already increased heavily.

This phase of the ALP-Mobile process can further be divided into activities which get examined in detail below:

Development Approach

A very fundamental decision is based on the development approach for the mobile application. As described in section 3.5 of this thesis there are three different approaches to develop a mobile

application. The decision whether a native app, hybrid app or web app is developed has a huge impact on the development phase of the product.

Of course this decision must be made according to the wishes of the customer. Nevertheless the development team can support the decision making with their technical expertise. Usually this decision is made during the kick-off meeting which takes place in the beginning of the project.

Requirements Engineering

Requirements engineering is essential in every software project. Especially in mobile application development this activity is very important, because mobile users expect to get the desired information as quickly as possible. Many mobile applications seem overloaded and therefore these applications do not get a positive feedback from users. The key to success is to concentrate on necessary functionalities. A good starting point is to focus on user requirements regarding the software. It is important to address the functions a user expects when using the application. As a result of this activity a list of user stories should be generated which define the expected functionalities of the application. In [86] *User Stories Applied: For Agile Software Development* Cohn states that a user story "describes functionality that will be valuable to either a user or purchaser of a system or software." [86, p. 4] User stories should not be too complicated, in fact according to Cohn a user story should be 1) independent, 2) negotiable, 3) valuable to users and customers, 4) estimable, 5) small and 6) testable. Furthermore a user story is composed out of three aspects: [86]

- a description which is typically written on a story card and which is used for planning and as a reminder
- conversations about the story which help to substantiate it
- tests about the story to decide when a story is finished

The creation of the user stories is performed by the product owner and requires close collaboration with the customer. After this task is finished the planning meeting takes place in which the user stories get estimated. As presented in section 4.3.3 especially in the beginning of the usage of ALP-Mobile all three presented roles should attend this meeting. It is vital that the development team is able to review the created user stories. Since the product owner does not have the technical background, he needs the support of the development team in order to specify the user stories correctly. The agile coach supports both the development team and the product owner to maximize the value of this meeting. As time goes by and the collaboration of the product owner and the development team solidifies the agile coach does not have to be present at each planning meeting.

Additionally to the estimation of the user stories, the product owner needs to prioritize the estimated user stories. This task again is based on proper communication with the customer. The product owner should focus on the user stories which bring the highest business value. Nevertheless this decision is again supported by the development team, because it is not always possible to implement the user stories with the highest business value at first. Sometimes user stories depend on each other and therefore the workflow must also be consider from a technical point of view.

The outcome of this activity is a ranked Backlog column on the kanban board which contains the generated item cards. Each item card refers to a specific user story. In preparation for the next phase in ALP-Mobile the product owner should already select user stories to be implemented at first and therefore those stories can get pulled into the Selected column of the kanban board.

Early Prototyping

Through the use of early prototypes the information gathered during the *requirements engineering* activity can be illustrated to the customer. Such prototypes should display already made decision regarding the design as well as the functionality. These prototypes can vary from simple drawings on paper to clickable UI mockups. Especially the usage of clickable UI mockups brings great value to the project team, because with such prototypes the future application can be modeled in detail. This way, the customer gets a great vision of the future application and therefore he is able to provide necessary feedback to the development team. This feedback may address simple changes concerning the design, but also bigger adjustments in the application, e.g. missing functionalities.

This activity should not consume too much time, nevertheless it is important to provide a proper user experience which is related to the functionalities of the final product.

4.4.2 Development Phase

The second phase of ALP-mobile is focused on the implementation and the testing of the software. The outcome of this phase are tested features which are ready for distribution.

Again, this phase is divided into activities which are presented in the following part of the thesis:

Setup

In this activity the technical infrastructure is prepared for the project. As the infrastructure is closely related to the project itself and the techniques used in the project, no detailed description about this activity is given. Nevertheless it is important to consider all necessary parts to setup in order to be able to start with the actual implementation of the software solution in the next activity. The technical infrastructure includes various tools and components, such as version control system, team and customer collaboration tools, project monitoring, among others. This activity is executed by the development team.

Implementation

The implementation phase of ALP-Mobile highly makes use of a kanban board, similar to the board illustrated in figure 4.4. Note that the author is not able to make suggestions concerning possible WIP limits for specific stages in the process. The reason for that is that those limits highly depend on the preferences of the project team and therefore this decision must be made in consideration with all three roles in ALP-Mobile. Nevertheless the author will point out whether a column in the process needs a limit or not.

In case the product owner and the development team decided that the user stories are equally sized, then the columns feature a WIP limit which is based on the number of items in this column. Otherwise a WIP limit, which correlates to the estimation unit defined in the planning meeting, needs to be determined.

Figure 4.5 illustrates the kanban board proposed for ALP-Mobile. Note that the columns which are supposed to have a WIP limit are marked with a placeholder, e.g. [X]. The author examines each stage of the workflow below:

- Backlog
  The first column in the kanban board is the Backlog column. This column is maintained by the product owner and it serves as a queue in the workflow. The product owner is responsible

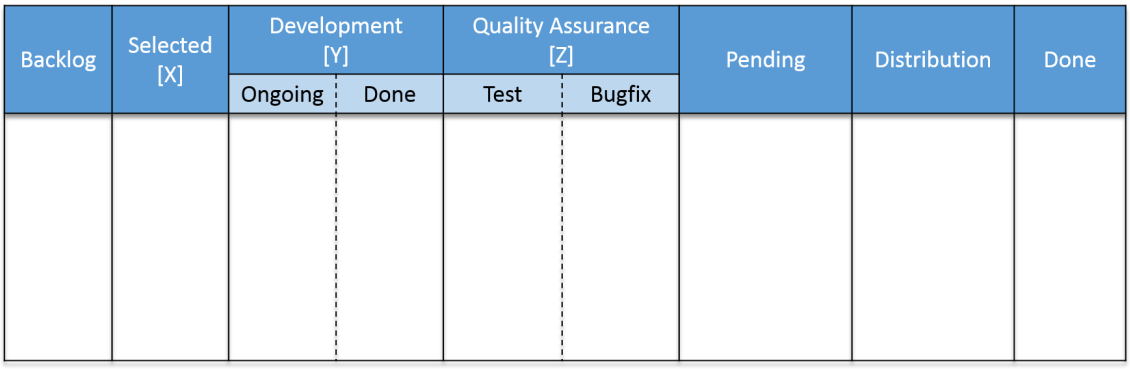

Figure 4.5: The kanban board in ALP-Mobile

for keeping the item cards up to date. In the beginning of the development phase this column should be filled with cards which refer to user stories.

- Selected
  According to the priority of the user stories, which is based on the business value and the technical feasibility of the stories, the product owner selects items from the Backlog column and puts it in the Selected column. Additionally a WIP limits needs to be defined for this column.

- Development
  This phase is divided into two sub phases which are Ongoing and Done. As illustrated in figure 4.5 this stage also features a specific WIP limit. Work items at first get pulled into the Ongoing column. After finishing the implementation task of this item it can be pulled into the Done stage. After that it is possible to pull it into the Quality Assurance stage.

- Quality Assurance
  ALP-Mobile utilizes a dedicated Quality Assurance column which has a WIP limit. This limit forces the team to test the features soon in the workflow, because otherwise the flow of the progress would struggle. The next activity in the development phase gives more insight into the testing routines in ALP-Mobile. As indicated in figure 4.5 this phase is split up into two subphases, which are Test and Bugfix. The reason for that is to avoid a possible deadlock in the workflow in case a feature is pulled in the Test stage and errors respectively bugs are found. Therefore this column features an extra stage in which the encountered problems are fixed. Note that it is possible to select an item from the Bugfix column and to put it back in the Test column. This is a necessary exception to the pull system of kanban in order to assure completely tested features when they are pulled in the Pending stage.

- Pending
  Since the distribution in mobile applications highly depends on the policies of the different distributions platforms like Google's Play Store or Apples App Store the distribution phase in ALP-Mobile cannot be as agile as the other two phases. Therefore ALP-Mobile introduces the Pending column on the kanban board which serves as a queue for distribution. In order to avoid the flow to stop, this column does not have a WIP limit, which means that as soon as an item card is tested and fixed if necessary it can be pulled into the Pending stage.

- Distribution
  Because of the mentioned restrictions of the different app stores the distribution column does not have a WIP limit either. This stage is used for item cards which are put into distribution and which need to go through the release process of the app stores.

- Done
  The last column on the kanban board collects all finished working items. As soon as this stage is reached the lead time can be measured.

According to the agile principle of inspect-and-adapt the chosen limits need to be adjusted to the preferences of each team. It is important to keep the right balance between a limit which is too low and a limit which is too high. A low limit could bring the situation that people on the team are finished with their work, but they are not able to pull a new item because of the WIP limit. A high limit on the other hand could bring up the situation that items which cause problems do not get finished, because new items just get pulled into the stage. As a result the team reacts late to problems, which increases the lead time. Such adjustments should be discussed in the proposed retrospective meetings.

Additionally to the presented concept the author advises to make use of *pair programming*, a technique used in XP. This technique is based on immediate feedback of the written code. The scenario is as follows: the code is written on one machine, with one keyboard, with one mouse and two people watching it. One person is using the keyboard and the mouse to do the actual coding by thinking about the best way to implement the feature. The other person is thinking in a more strategic way, e.g. concerning the approach of the implementation [81]. The benefit of this technique is an increase of code quality as well as an increase of the team spirit. Pairs should constantly be switched to make the most out of these pairing sessions. Of course this technique should not be performed all the time - additionally not every programmer likes the idea of this technique. So it depends on the team how much it uses this technique.

Testing

The importance of testing in software development cannot be emphasized enough. Nevertheless as described in the basic overview to ALP-mobile testing for mobile applications is not trivial at all. The very wide range of devices makes it very hard to test the applications in their natural environment. Kim states further: “Within Mobile applications software circles, the practices of Test-Driven Development (TDD) and Continuous Integration (CI) are either unknown or have been regarded as prohibitively difficult to use.” [87, p. 785]

The above citation represents exactly the problem mobile application development is currently focusing when it comes to software testing. A consequence of this situation is that apps are mainly tested manually by the developers[2]. This is no optimal solution, but due to the lack of proper alternatives manual testing is an important part of mobile software development nowadays. Additionally the fact that mobile applications often feature very detailed and customized solutions in the GUI makes it difficult to write automated tests for such scenarios.

A common situation in software development is that the first tester of the software is the actual developer. Especially in mobile application development this situation is met very often. This is the first stage in assuring a good quality of the code. Nevertheless it is essential that dedicated testers are involved in the quality assurance process of the mobile software. Because of the fact that those people interact differently with the software, bugs in the application can be discovered. Experienced testers additionally focus on possible weaknesses in the software which developers did not think of during the implementation.

Before going into more detail about testing in ALP-Mobile the author examines the characteristics of mobile software and the implications on testing. For that the following overview presents

[2] See figure 4.2.

specific characteristics of mobile applications and for each characteristic the impact on testing is discussed: [88]

- Connectivity
  Through the fact that mobile applications are constantly logged in to a network, mobile applications are always online. This network though does vary in speed, reliability and security. Especially slow and unreliable networks are a major issue in mobile application development. Concerning the impact on the testing strategy it is important that mobile applications are tested in different networks and connectivity scenarios.

- Convenience
  The design of mobile applications is heavily influenced by the operating system of the mobile device. Additionally screen resolution and size bring changes to the UI of the application. Issues related to changing UI components have been reported. Therefore it is important to test the same application on different mobile devices in order to make sure that the application runs properly on different screen resolutions and sizes.

- Supported Device
  Especially in Android development the high number of devices presents a real problem in mobile application testing. The different versions of the operating system installed on those devices increase the complexity of testing in this field of software engineering even further. Because of that situation, mobile applications must be tested throughout different devices featuring different versions of the operating system.

- Touch Screens
  The main source of input on mobile devices nowadays is a touchscreen. The response time of touchscreens varies and strongly depends on the resources of the device. The implication on testing is that mobile applications must be tested under different circumstances, e.g. high process load.

- New Programming Languages for Mobiles
  Because of the introduction of new programming languages for mobile applications (e.g. Objective-C for iOS) traditional structural testing techniques must be adapted to the programming languages in mobile application development.

- Resource Constraints
  Mobile applications become more and more complex, but the resources of mobile devices are still not comparable to desktop computers or laptops. Therefore the resource usage must be constantly monitored in mobile applications.

- Context awareness
  Mobiles feature a high number of sensors and connectivity devices. Apps make highly usage of those features and therefore those apps have to be tested very carefully when making use of such features. According to [88] bugs associated to contextual inputs have been reported quite frequently.

Based on the problems which are present in mobile application testing the author introduces a testing concept specified for ALP-Mobile. The idea behind this concept follows the lean principles of *learn constantly* and *keep getting better*. In detail this means that a three step concept is advised, whereas the first step is prescribed in ALP-Mobile. The steps are as follows: 1) Manual Testing, 2) Automated Testing and 3) Continuous Integration. The stages are built on each other, which means that even for a team implementing continuous integration in their project ALP-Mobile requires to

Figure 4.6: Testing concept in ALP-Mobile

perform basic manual testing. Figure 4.6 illustrates the concept. Each of these steps is presented below:

- Manual Testing
  As stated, mobile software is mainly tested manually nowadays. This is not ideal, because the costs of this testing solution are very high. Nevertheless especially in mobile application development the task of manual testing is very essential, since mobile software highly makes use of UI design and gestures to control the software. Such functionalities are hard to test automatically.

  The author already briefly discussed the importance of user acceptance tests in mobile application development. Such tests are executed in order to test the software in real world scenarios. Especially focusing the already mentioned problems in mobile application testing user acceptance tests provide essential information to the development team. The author recommends user acceptance tests executed by the customer during the development phase of ALP-Mobile. This implies a high customer collaboration which is necessary in mobile application development. Especially at the end of the development phase user acceptance tests are important to fully test the functionality of the software.

  Furthermore the task of exploratory testing is important in mobile application development. In exploratory testing the testers intentionally try to find bugs in the software. Besides the fact that new bugs have been discovered, the testing scenarios executed to find the problems in the software serve as a basis for the next step in the testing concept of ALP-Mobile.

- Automated Testing
  The next stage in the testing concept is automated testing which is built on top of manual testing. The experiences gathered through manual testing function as a foundation for the creation of automated test cases.

  In *Mobile Application Testing – Challenges and Solution Approach through Automation* [88] Kirubakaran and Karthikeyani concentrate their research on the automation of mobile application testing. Because of the inability of simulators to simulate real-world scenarios, they state that new capture-and-replay techniques can be used in automated testing. As an example they name the Android testing framework *Robotium* [89].

  Concerning GUI testing they state further: “An idea can be to automatically execute scripts that are captured during the user interaction and replayed and modified even on different devices. It is important to make this task automatic, so to avoid to manually interact with the GUI that is a time consuming task.” [88, p. 82]

- Continuous Integration
  The final and most desired step in the testing concept is continuous integration. This is still a topic for research in mobile application development, nevertheless is has been proved that this technique is very valuable in software development in general and therefore the

author of this thesis sees a great future for continuous integration in mobile application development.

Kirubakaran and Karthikeyani point in [88] out that the first step towards more reliable and correct mobile applications is the automation of the testing approaches in mobile application development. Since mobile apps have the common perception to be cheap, developers must focus on the reduction of costs in the development. As pointed out, in software development in general the techniques of automated testing and continuous integration have decreased the costs of testing.

Concerning the testing of mobile applications it is important to implement a proper test automation before setting up a continuous integration system which executes the automated tests on every commit of a developer. Nevertheless there are also situations in which it is advisable to set up a continuous integration system even before automated tests are created. A possible scenario could be the usage of a continuous integration system to ensure the right compilation of the code base at any given time in the project lifecycle. Although the application is not completely tested in an automated way, developers must make sure that they only commit working code, which increases the motivation of the developers to test the application manually respectively to work more accurate, because broken builds can be traced back to the developer.

Despite the fact that continuous integration is not very common in mobile application development, there are build servers available which focus on mobile application testing. The author recommends the use of *Jenkins* [90], which is the de facto standard in mobile application testing.

4.4.3 Distribution Phase

The last phase of ALP-Mobile is dedicated to the process of bringing the application to its desired market. Additionally product maintenance is a key to success in mobile application development, since for example discovered bugs in an application must be patched as soon as possible in order to keep up a good reputation about the software. This phase is divided into two activities which are presented below:

Deployment

According to which type of application is developed the deployment differentiates. When developing web apps the deployment is similar to web applications in general. When developing hybrid or native apps, which are distributed via dedicated app stores the deployment is different. In such cases the software needs to get prepared, e.g. certified by the app store, to be available for distribution respectively download. Furthermore licenses may need to be purchased for specific app stores and the policies of those stores need to get considered. At the end of this activity the software should be available to end users.

Maintenance

This activity covers various tasks of maintenance, e.g. support, bug fixing, etc. Especially for mobile applications a good user feedback is essential to be successful. Due to the fact that everybody is able to rate downloaded apps it makes it very important to receive positive ratings. Due to the extreme variety of mobile devices and the different scenarios in usage it is almost impossible to develop a bug free software. Such bugs then get reported through users. In order to keep up a

good reputation fast bug fixing is highly advised. Of course such maintenance contracts need to be negotiated with the customer.

4.5 The ALP-Mobile Workflow

This section of the thesis is dedicated to display the process of ALP-Mobile in detail and furthermore to show the differences to other process models and approaches. In order to achieve this, the author presents a typical workflow executed in ALP-Mobile. By displaying the workflow the author offers guidance to people implementing the ideas of ALP-Mobile. Figure 4.7 illustrates the simplified workflow in ALP-Mobile.

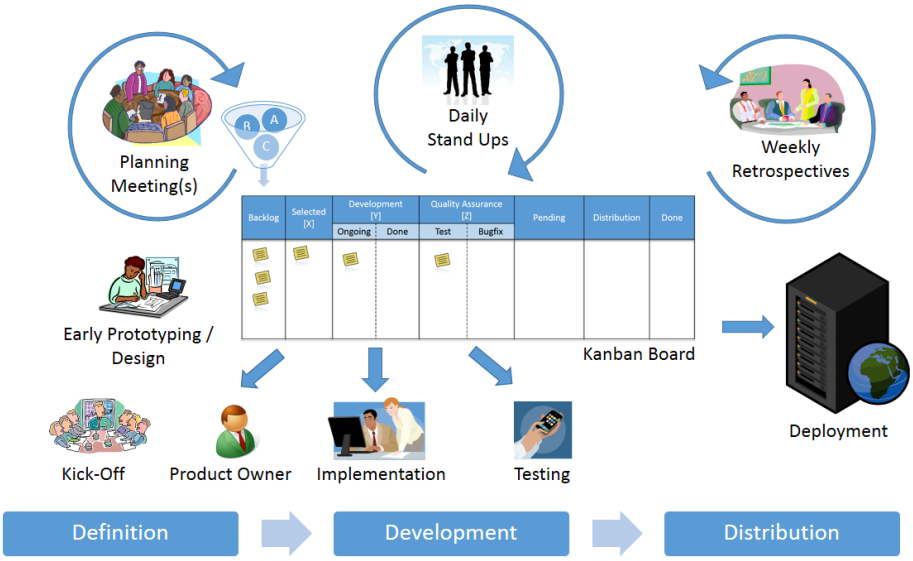

Figure 4.7: The ALP-Mobile workflow

Basically ALP-Mobile focuses on mobile software projects which are developed for a specific customer. Therefore details concerning the project are discussed with the customer in a preliminary meeting. After this preliminary meeting the prescribed kick-off meeting is held. The content respectively the significance of the kick-off is presented in section 4.3.3. The kick-off meeting already includes some tasks dedicated to the definition phase of ALP-Mobile, e.g. determination of the technology stack which is associated with the activity of defining the development approach.

After the kick-off meeting further activities from the first phase of ALP-Mobile are executed, e.g. requirements engineering. This activity is very important, because it functions as a basis for the development phase of ALP-Mobile. In a nutshell, the product owner defines user stories according to the wishes of the customer and furthermore schedules the planning meeting in which the user stories are estimated. As advised, early prototyping is strongly encouraged in ALP-Mobile, since this task leads to valuable feedback from the customer early in the project. Note that not all features respectively requirements must be defined in the beginning of the project. This flexible approach is necessary in mobile application development and also supports agile and lean thinking.

Communication is essential in all activities included in the first phase of ALP-mobile. The high collaboration with the customer fits the lean practice of customer participation. As an outcome of the activities included in this phase the development team should be ready to start with the implementation of the software.

As a next step in the process of ALP-Mobile the development phase with its activities is executed. The kanban board is essential throughout the whole process. The Backlog column is already filled

with the (estimated) user stories and the development team is ready to implement the first work items from the Selected column on the kanban board. The Selected column is maintained and filled by the product owner according to the priority of the user stories, which is based on the business value and the technical feasibility of the stories. The process is now controlled via the WIP limits on the kanban board. Features which are implemented and tested can get pulled into the Pending column of the board. Those features present the current version of the software prior to the first distribution. New features, which are added to the product after the initial definition phase, need to be written down in terms of user stories. As stated in the description of the planning meeting performed in ALP-Mobile, the product owner is responsible for keeping the user stories up to date and to request a new planning meeting if new user stories need to be estimated. The iteration of the planning meeting and the ongoing changes in the backlog represent a clear distinction to traditional software process models, e.g. the waterfall model, in which completed phases are not executed again.

In agreement with the customer a specific version of the software is then distributed. Therefore the activities in the last phase of ALP-Mobile are executed. After the distribution the work items can get pulled into the Done column. As stated, the process is not finished with the first distribution of the software. The enhanced flexibility which is established through the execution of a process without fixed iterations makes ALP-Mobile very efficient and productive in the very short-lived mobile market.

As illustrated in figure 4.7, ALP-Mobile requires ongoing retrospective meetings in order to review the development of the product and the product itself. This meeting brings great value to the whole project team, since valuable feedback is gathered from the customer. The retrospective part of the meeting helps to make changes to the process. This meeting is scheduled weekly. Additionally figure 4.7 displays the prescribed daily stand up meetings.

The defined three phases introduced into ALP-Mobile must not be seen as strictly linear. To the contrary, ALP-Mobile follows an iterative approach in which new features can be added to the product at any given time. Those phases should rather be seen as the main steps in mobile application development. The activities in each phase provide a guideline of tasks to be executed in order to finish a mobile project successfully. Concerning the process and the changes over the phases the author sees, especially because of the restrictions in distribution in the mobile market, the agility in the distribution phase decreased compared to the first two phases in which the process is very agile and lean.

4.6 Metrics in ALP-Mobile

In ALP-Mobile the time of a working item to complete the process in measured via the lead time. ALP-Mobile does not prescribe any type of chart like the burndown chart in scrum. Nevertheless the author advises the use of the cumulative flow diagram which displays the flow of the process and furthermore shows how the WIP affects the lead time. Figure 4.8 presents a possible example.

In this figure the x-axis denotes the days in the project and the y-axis denotes the total number of working items currently on the board. The vertically drawn arrow shows that on day four in total nine items were on the board. One in production, one in test, two in development and five in the backlog. Plotting those numbers every day a diagram like the example in figure 4.8 can be created. The horizontal arrow donates the time a working item added on day four needed to get to production, e.g. six days. This is exactly the lead time of this working item. The diagram shows that almost half of the lead time was used by the Test phase. By reducing the WIP limit in this Test stage the lead time could be decreased.

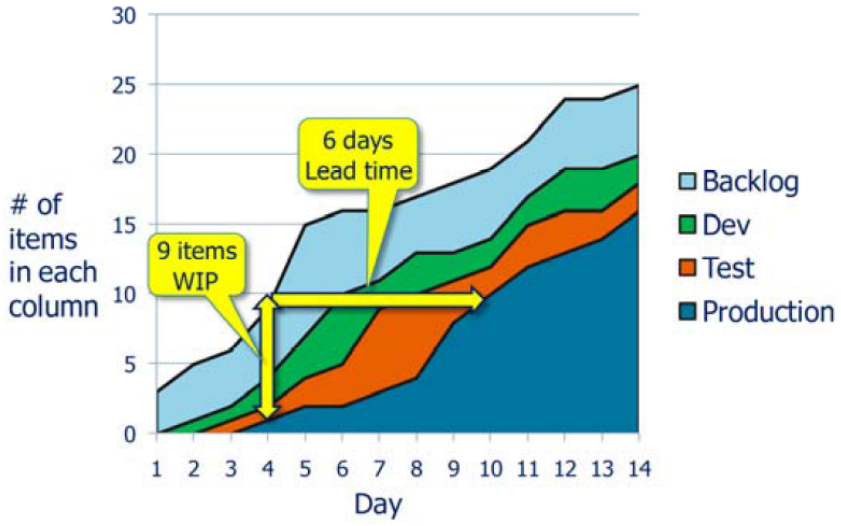

Figure 4.8: The cumulative flow diagram [46]

4.7 Distributed Development in ALP-Mobile

The author of the thesis already expressed the possibility to use ALP-Mobile in a distributed environment. As stated in section 2.4, teams in a distributed environment face problems which increase with the level of distribution. This section supports teams using ALP-Mobile in a distributed environment.

In an optimal situation the whole team is able to attend all meetings in person. In a distributed environment this situation is often not possible. Therefore the situation should be reconstructed as well as possible. The author already mentioned the help of video conferences in such cases. With this help people still can talk face-to-face which is very important. In cases where a video conference is not possible, protocols of the meetings should be created to inform the people working on another site about the progress in the project. Without having the chance to talk face-to-face the whole team cannot be part of the daily scrum. Therefore it is the responsibility of people working on other sites to give a daily status update about the progress. This status update then can be integrated into the daily stand ups.

Since the kanban board is the main source of information in ALP-Mobile it is important that every member in the team has the ability to view it whenever he needs to. For people working in the office this is no problem. To make this information available to people working from different sites the author advises to constantly film the kanban board and make it available over the Internet. This does not bring high costs, since a normal webcam is sufficient for this task. Additionally another webcam with a view on the working site can be installed, which helps to simulate the social component of working together in a team.

Nevertheless the face-to-face contact is essential and therefore the whole team should at least meet from time to time in order to keep up the relationship within the team.

4.8 Summary

This part of the thesis introduced the reader to ALP-Mobile. ALP-Mobile is an agile and lean process model for mobile application development which helps to overcome the challenges in this field of software development. It features three main phases which are referred to as the 3D's of

ALP-Mobile which are 1) Definition, 2) Development and 3) Distribution. Each of these phases features specific activities which are recommended to be executed within this phase. Through the use of kanban without fixed iterations the team is able to adapt to the high velocity of the mobile market and to the inputs of the customer. The kanban board functions as the main source of information within ALP-Mobile. Figure 4.9 gives an overview over ALP-Mobile.

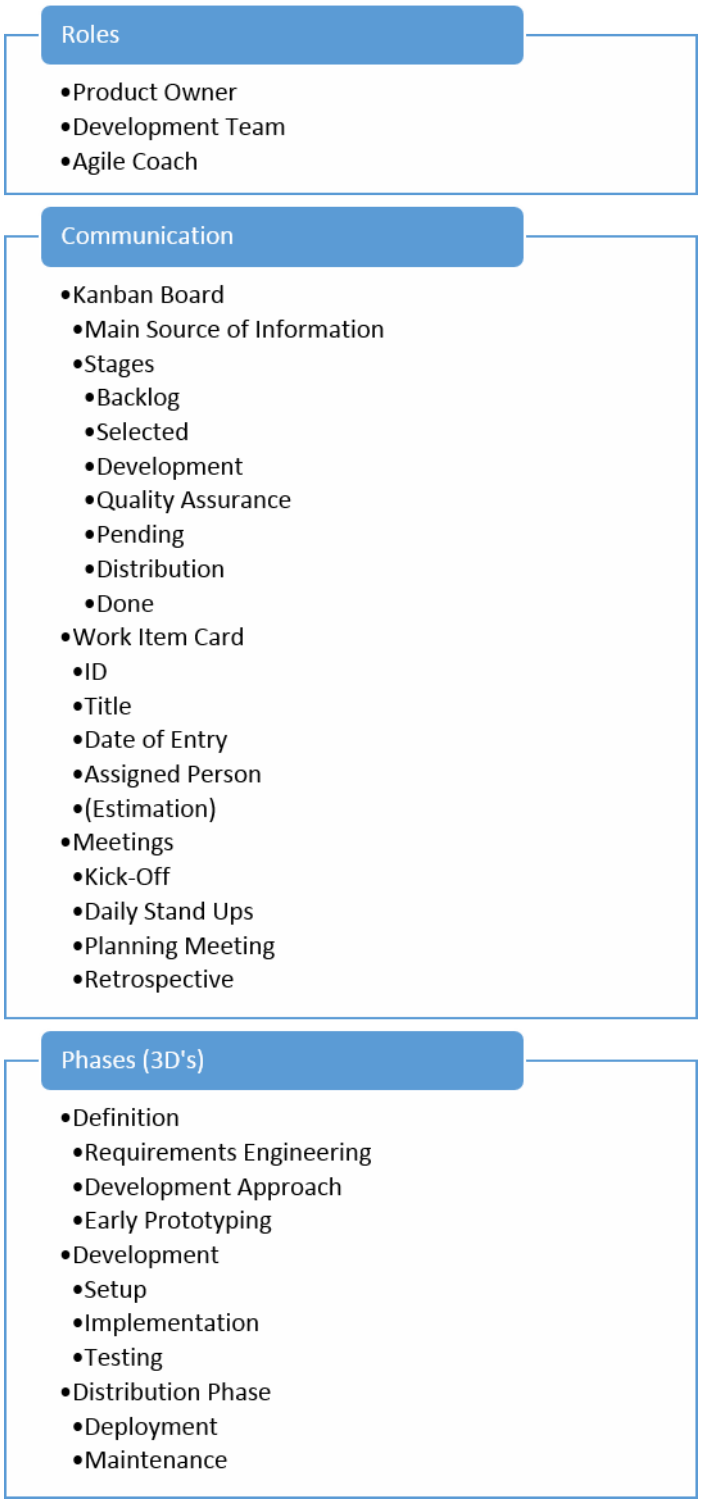

Figure 4.9: The ALP-Mobile process model

5 Case Study

The previous chapters introduced the reader to the topics of this master thesis. The result of the literature research was then illustrated in chapter 4 where ALP-Mobile, an agile and lean process model for mobile application development, was presented in detail. In this chapter a case study is conducted which investigates the applicability of agile and lean methodologies in the area of mobile app development. Interviews with different roles in this field of software development are the main source of information for the case study. Through the input of experts in the field of mobile app development, the author is able to derive strengths and weaknesses of ALP-Mobile in the next chapter of this thesis.

The case study performed in the course of this research is based on the guideline given in [91]. Verner et al. recommend to divide the case study into four different phases. The first phase is dedicated to the determination of the basic characteristics of the case study. Furthermore the selected cases get introduced. In the second phase all necessary steps to execute the case study are planned in detail. After that, the actual data for the case study is collected. Conclusively the last phase analyzes the collected data.

5.1 The Basics of Case Study Research

Before getting into more detail about the case study the author provides a deeper insight into case study research in general. In her article *Toward Better Case Study Research* [92] MacNealy discusses problems in case study research and furthermore presents strategies for better case study research. She points out that the term *case study* is often misused, which is a common issue. As an example MacNealy states that often personal histories of practitioners are referred to as case studies. Besides the value of such information, it must be referred to as *case history*. Another example of the misuse of the term case study is the classroom. Teachers often use cases to illustrate a concept by using a description of a particular situation. This case helps the students to understand specific problems. Nevertheless a case is used for this learning process, the term case study is not appropriate, because nothing new is discovered. The case was rather used to illustrate a specific issue. In such cases the right label would be *case material*.

In order to clear up the confusion MacNealy defines a case study as follows: “Case study research is a qualitative tool; as such, it aims to provide a rich description of an event or of a small group of people or objects. Because the scope of a case study is so narrow, the findings can rarely be generalized; but a case study can provide insights into events and behaviors, and it can provide hypotheses for testing .” [92, p. 183]

Another definition is provided in [91, p. 313]: “Rather than using a large number of samples and following a rigid protocol to examine a limited number of variables, case study methods involve an in-depth examination of a single case or a small number of cases (comparative case study). They provide a systematic way of looking at events, collecting data, analysing information, and reporting the results.”

Through the use of case studies the researcher gains deeper understanding of a specific research topic. Additionally case studies can unveil topics of future research. Due to the fact that case study research is performed in real-life settings, a greater insight into the investigated phenomena

is provided. In [92] advantages and disadvantages of case studies are presented. The following overview illustrates some positive aspects of case study research:

- Case studies present a holistic view of a specific event or situation. Furthermore they show the context of the problem as well as details of the event.
- Through the rich information gathered within the case study, a deeper understanding of certain aspects is provided.
- Case studies enable researchers to collect affective information which can not be collected otherwise. A possible example are feelings of a person when performing a specific task.
- Research questions can get determined more precisely, which helps to further explore problems which are not well defined.

After displaying the advantages the author presents the disadvantages listed in [92]:

- As stated in the beginning of this section a common issue in case study research is that the case study methodology is often misunderstood and that the term case study is wrongly used. The lack of knowledge in case study research frequently leads to poorly executed case studies.
- The results of case studies are often tied very closely to specific situations, which makes it hard to generalize the results.
- Since case study research does not make use of rigorous methods it is often referred to as *soft science*. This situation brings the criticism that case study research is often biased by the researcher.
- Compared to other research methods case studies are sometimes expensive to execute. As an example MacNealy provides the expensive task of transcribing conducted interviews.

After displaying the pros and cons of case study research, the author presents the two different types of data which can be collected during a case study. On the one hand there is quantitative data and on the other hand there is qualitative data. Both types are discussed in [93]:

- Quantitative Data
  This form of data is often used if time and resources are limited. It is reasoned with the fact that quantitative research usually involves the use of instruments like surveys and tests which produce useful data in a short time frame. This data is conducted from large groups of people and the investment in personnel and material is reasonable. Typically the quantitative method is chosen when only a few variables are the source of investigation.
- Qualitative Data
  In contrast to quantitative data collection, the qualitative approach focuses on the richness of the information gathered. Examples for the qualitative data collection are among others interviews, focus groups or direct observations. These methods consume more time and resources to represent the area of research. The qualitative approach is typically used when little is known about an issue, because with this approach various factors, which influence a specific situation, are examined. Another factor which influences the choice between quantitative and qualitative data is the amount of people participating in the research. Especially in qualitative research it is sometimes difficult to find people who are willing to contribute to the research since it consumes considerable more time than participating in quantitative research.

METHOD	(1) Form of Research Question	(2) Requires Control of Behavioral Events?	(3) Focuses on Contemporary Events?
Experiment	how, why?	yes	yes
Survey	who, what, where, how many, how much?	no	yes
Archival Analysis	who, what, where, how many, how much?	no	yes/no
History	how, why?	no	no
Case Study	how, why?	no	yes

Figure 5.1: Types of research questions [94]

In terms of which research type is the best fit for a specific research, Yin investigates in [94] different research questions and further associates those questions to different research methods. Figure 5.1 illustrates an overview concerning the investigation.

The investigation is based on three factors which are displayed in figure 5.1. These factors are 1) the form of the research question asked, 2) the control of the researcher over behavioral events and 3) the degree of focus on contemporary events. The research questions are associated to five main research methods which are 1) experiments, 2) surveys, 3) archival analyses, 4) histories and 5) case studies. In case the research question starts with *what*, the preferred method is either a survey or an archival analysis. *How* and *why* questions are on the other hand more exploratory and therefore those questions likely lead to the use of case studies, histories or experiments. In some cases *what* questions can also be exploratory, which leads to the possibility to use any of the proposed methods. [94]

Before the decision is made whether a case study is the appropriate research method or not, Verner et al. [91] recommend to define the study objectives of the research. The aim of this task is to set broad and overarching objectives for the identified area of interest. Following the author states the objective of this research:

The objective of this research is to investigate the impact of agile and lean methodologies in the field of mobile application development. Given the fact that these methodologies are well known in software development in general, this study aims to explore the usage of agile and lean in an area where only little research about the applicability of those approaches exists.

The next step before making a decision regarding the research method is to undertake a comprehensive literature research. The literature review serves as a solid foundation for the research. Additionally it should consider all previous significant work in the area of the research in order to find specific research problems for further investigation. Concerning this research, the author presented the findings of the literature review in the first chapters of this thesis. A detailed overview over the state of the art was given in section 3.6.

Following to the definition of the research objectives and the execution of the comprehensive literature research the investigator is able to decide whether a case study is appropriate or not. Based on the presented information about case study research in this section the author chose the case study approach as the adequate research method for this master thesis. This study can be

classified as an exploratory case study as it aims to explore a subject area where only little research exists.

By presenting the basics of case study research the author introduced the reader to the concepts of this research method. The remainder of the chapter focuses on the case study conducted within this research.

5.2 Case Study Focus

This section of the thesis describes the steps which comprise the phase that determines the basic characteristics of this case study.

5.2.1 Generation of the Research Question

The first step in the case study is the generation of a research question out of research objectives which should be the content of the study. In case study research this question usually begins with *how* or *why*. The author already defined the research question in the first chapter of this thesis:

How can agile and lean processes be adapted to mobile app software development?

Definition of the Research Propositions

Based on the research question research propositions are defined. Yin states the following about research propositions: "Each proposition directs attention to something that should be examined within the scope of study." According to this definition the author proposes the following six propositions which are related to the research question. [94, p. 60]

- PR1: Organizations use agile processes because it is a good fit for small projects
  The first proposition is concerned with the reason why organizations use agile processes in their projects. As pointed out by Abrahamsson in [8], ideal agile characteristics are among others small development teams and small systems. Although mobile software has become more and more complex over the years, app projects are still small compared to big, traditional software projects. Therefore the author states this proposition in order to find out if organizations, which focus their services on mobile application development, use agile respectively lean processes at all and furthermore to investigate the reasons for the execution of those processes.
- PR2: Mobile application development requires short iterations
  Short iterations are one practice of lean software development defined in [38]. The author is interested if organizations in the field of mobile application development agree with this practice and therefore the proposition regarding short iterations is proposed.
- PR3: Automated testing and continuous integration are not very common in mobile testing
  The third proposition targets the area of testing in mobile app development. Based on the literature research performed in the beginning of this master thesis the topics of automated testing and continuous integration are not very popular in this field of software development. In [75], Wasserman points out that testing is an important area for mobile software engineering research. Kim states in [87] that practices like test-driven development and continuous integration are either unknown or difficult to use. As a consequence the majority of apps is

tested manually. By defining this proposition the author wants to examine the opinion of the experts regarding testing of mobile applications.

- PR4: Small team sizes, which are common in mobile app development, support distributed development
  The introduced activities in ALP-Mobile, which are dedicated to the possibility to use ALP-Mobile in a distributed environment, help teams to overcome the challenges in distributed development. Abrahamsson defines in [8] small development teams as one ideal agile characteristic. The author proposes this proposition in order to investigate the possibilities of distributed working in the area of mobile application development.

- PR5: Early prototyping is very important in mobile app development
  Ben Morris et al. define in [64] early prototyping as an important activity in the development of a mobile application. The purpose of this proposition is to examine the opinion of the experts regarding this topic.

- PR6: Mobile application development requires high collaboration with the customer
  The last proposition concerns the collaboration with the customer in mobile app development. Customer collaboration is an important concept in both agile and lean software development. One of the four values proposed in the agile manifesto [6] is *Customer Collaboration over Contract Negotiation*. In [38] Hibbs et al. define customer participation as one practice of lean software development. By defining this proposition the author wants to investigate the importance of this concept in the field of mobile software development.

The defined propositions are examined in detail in chapter 6 where the author discusses the results of this master thesis. The discussion includes a detailed investigation of the research propositions and the answering of the research question.

5.2.2 Identifying the Unit of Analysis

As a next step in the case study focus the author defines the unit of analysis. The unit of analysis must be decided with care and it must fit to the research question. Concerning this research the unit of analysis are organizations which focus their services on the development of mobile applications. By choosing such organizations a great insight into the processes associated with the development of mobile applications is provided. Furthermore through the opinion of experts the applicability of ALP-Mobile in this field of software development can be determined.

5.2.3 Determination of the Boundaries of the Case Study

The next important step is to define the boundary of the case study. According to Verner et al. [91] this step describes the scope of the study as well as the criteria that is used to determine the scope.

In case of this research only organizations working in the area of mobile application development are considered to be a part of the research. The data collection performed within this case study is based on the execution of qualitative interviews. First-hand experience with the development of mobile software is a prerequisite for the selection of the participating experts of the interviews.

5.2.4 Finding Feasible Cases

In order to find experts for the interview the author performed an Internet research about organizations located in the city of Vienna which focus their services on the development of mobile

software. As pointed out in section 5.1, a problem in the collection of qualitative data is the fact that people often are not willing to contribute to the research since it consumes considerable more time than participating in quantitative research. As a benefit for the organizations, which are part of this case study, the final thesis and therefore the results of this research will be provided. Furthermore a review meeting after the completion of the thesis is offered to the experts in which the results can be discussed in person.

5.2.5 Presentation of the Selected Cases

The selection of the cases was performed according to the defined boundaries of the case study in section 5.2.3. Following the author presents the chosen cases and introduces the role of the experts, which participated in the interviews, to the reader. Additionally the attitude of the organizations concerning process models is discussed briefly. Due to data privacy reasons the organizations as well as the experts are anonymised in the remainder of this thesis.

- Alpha
  The organization Alpha is specialized on the development of native mobile applications for Android and iOS. On the one hand they work on projects for specific customers and on the other hand they implement their own products. The interview was held with the CEO of Alpha. Besides his activity as CEO, the expert is responsible for a clean and structured project management within the organization. Alpha has 10 employees, nine of which work in the development department. Concerning processes in the organization there are differences between projects for customers and the development of own products. This is reasoned with the fact that the budget is clearly defined within customer projects, which changes the development process of Alpha, because new features must be communicated with the customer, which decreases the agility in the development.

- Beta
  The organization Beta works in the health sector and they develop apps for diabetics. They solely concentrate on their own products, which means that they do not realize projects for customers. At the moment they have two apps. Both feature iOS and Android is coming soon. The representative of Beta, who attended the interview, was the CTO of Beta. The CTO of this organization is responsible for the successful development of the products. The organization has seven developers and in total there are approximately 25 people involved in Beta. Some of those people work only part times. Due to the fact that Beta is ISO certificated, which is needed to develop products for the health sector, they must document their processes to prove the quality standards that are needed for this specific sector. Therefore software processes are very important for Beta.

- Gamma
  The organization Gamma focuses on the development of apps and on the realization of customer projects. Furthermore they help organizations with the marketing of apps by using channels like Facebook and Google. Additionally they offer consulting services. At the moment they have 26 employees, 14 of which are developers. The other people work in the departments of Quality Assurance, Design and Project Management. Gamma develops apps for iOS, Android and Windows Phone. Furthermore they offer the following platforms: HTML5, Blackberry and Unity. The interview was conducted with the CEO of Gamma. The expert has 10 years of experience in the IT sector, six years of which in the area of mobile development. Since 2011 he is CEO of Gamma and responsible for the departments Mobile Health, Mobile Banking and Telecommunications. Concerning the processes established in the organization, they focused on agile software development from the beginning on, but

not with the "overhead of full blown implementations of scrum" as the CEO states. This is reasoned with the fact that scrum with all artifacts reduces the flexibility as well as the speed in the development. Therefore they make use of a lightweight version of scrum.

- Delta
  Delta is an IT provider in the area of Banking and Finance. The interviewed expert has eight years experience in software development. After his studies he started to work for the organization approximately three years ago. He is solution manager in the mobile department of the organization. The solution manager has a high focus on requirements engineering and is further responsible for the coordination and the lead in the mobile projects. Concerning the development of the software, the department is divided into interface development and app development. The interface development is responsible for the connection to the core backend system. This part of the development is performed directly in the organization and the classic app development is executed through nearshoring with other organizations in Vienna. Currently they use scrum as a process model. Regarding the team size, the mobile department has at the moment three solution managers, five developers and one tester. Due to the growth of the mobile market and the increased focus on those apps the number of employees will, according to the expert, increase further in the future.

- Epsilon
  The organization Epsilon focuses its services on digital communication and mobile app development. As a full-service agency they offer services in consulting, design, implementation and marketing of digital solutions. Their main competences are digital publishing, mobile sales support and the development of enterprise apps. The interview was conducted with the CTO of Epsilon. As technical operator the expert is responsible for everything which can be associated to the development of software in the organization. At the moment the organization has approximately 15 employees. Some of those employees are freelancers and roughly two thirds are developers. The rest of the personnel is focused on sales and project management. Concerning the processes in the organization, they do not follow any strict process models, but focus on the independence of their developers. This is reasoned with the fact that the organization realizes rather small projects from which 50% of those projects are implemented by only one developer. Therefore the implementation of a complete process model is seen as unnecessary overhead.

- Zeta
  The organization Zeta develops software for middle and large businesses. The interview was held with a software developer in the mobile department of the organization. The mobile department focuses its services on the development of native applications for iOS, Android and Windows Phone. The interviewee is tech lead for Android development. The mobile department has five full-time employees and if necessary additionally up to three students. Regarding the processes in the team respectively the organization, they focus on agile software development using scrum. They try to implement scrum as well as possible, but they also adapt the process to their own needs. Especially the iteration length of two to four weeks recommended in scrum is too long for the organization. Therefore they have, especially in the mobile department, shorter iterations to increase their flexibility in the development.

- Eta
  Eta is a full-service IT agency. Their product portfolio covers the area of project development, design and visualization, conception and the implementation of app-, web-application- and Internet-projects where they start from developing ideas and concepts over

to the design phase through the implementation of the project to the post-project phase. Regarding the size of the organization, at the moment they have two permanent employees and furthermore they work with six freelancers. The interview was conducted with the CEO of the organization. Before becoming CEO, the expert was employed as project manager and consultant at one of the biggest media agencies in Vienna. Due to the fact that this agency has a subsidiary in Serbia, the interviewee had to coordinate the work with developers both in Vienna in Serbia. Eta was founded in January 2014. Concerning the process established in the organization, they do not follow any strict agile process model, but they try to go an agile way during the implementation. Nevertheless they focus on traditional phases in software development, like conception, implementation, test and so forth.

Table 5.1 gives an overview over the presented cases and outlines the main characteristics of the organizations as well as the different roles of the experts which participated in the interviews. Note that the column *size* refers to the number of employees (+ freelancers) directly associated to the mobile department of the organization respectively to the total number of employees (+ freelancers) of the organization in case the organization solely focuses on the development of mobile applications.

Organization	Size	Role of the expert
Alpha	10	CEO
Beta	25	CTO
Gamma	26	CEO
Delta	9	Solution Manager
Epsilon	15	CTO
Zeta	5	Android Tech Lead
Eta	8	CEO

Table 5.1: Characteristics of the selected cases

5.2.6 Selection of Pilot Cases

Verner et al. recommend in [91] to undertake a pilot study before running the *real* cases. Concerning the selection of the pilot case Yin states in [94] that convenience, access and geographic proximity are the main criteria for selecting the pilot case. Another criteria for the selection of the pilot case could be the fact that the selected case is the most complicated case in the pool of selected cases. This way all data collection issues will be encountered in this case. As a result of the execution of a pilot case the researcher is able to refine the data collection plan regarding the content as well as the procedures. In general the scope of the inquiry in a pilot case can be much broader than the scope of the inquiry in the actual cases. This can be reasoned with the fact that the pilot case gives insight into the basic issues to be studied and therefore the inquiry can be less detailed compared to the ultimate data collection plan.

Since the presented cases do not vary a lot in their complexity, the author selected the organizations Alpha and Beta as the pilot cases, because the author already knew the CEO of Alpha upfront. The organization Beta was also selected, because due to the fact that they have an ISO certified quality management, the proper execution of process models is very important for the organization and therefore the author expected to receive valuable input for ALP-Mobile.

5.2.7 Needed Level of Confidence

As a last step in the case study focus Verner et al. [91] state that the researcher needs to decide if an appropriate level of confidence is provided by the number and type of the selected cases. The decision is of course based on the fact that especially in case study research the pool of possible cases is not very large, because of already mentioned factors like location, willingness of the organizations, resources and time just to name a few. Critics of case study research often claim that strong conclusions based on such a limited sample are not valid. Nevertheless in many cases it is sufficient to report the findings made in the course of the case study. Of course the researcher should not generalize the findings over a wider population, but rather interpret the results using rich interpretation methods. The outcome of the case study may later on be further validated via empirical research.

In terms of this research the number of cases is appropriate for the scope of this master thesis. Nevertheless the author focuses on this issue in chapter 6 in which the results as well as the limitations of the research are discussed.

5.3 Design of a Detailed Plan

The next part of this thesis is the design of a detailed plan for the case study conducted within this research. Yin states the following about the design of a case study: “In the most elementary sense, the design is the logical sequence that connects the empirical data to a study’s initial research questions and, ultimately, to its conclusions. Colloquially, a research design is a logical plan for getting from here to there, where here may be defined as the initial set of questions to be answered, and there is some set of conclusions (answers) about these questions.” [94, p. 57]

5.3.1 Definition of the Data Collection Strategy

A part of the design is the determination which data collection strategies are used in course of the case study. Verner et al. [91] state that it is important to collect data in as many ways as possible. In case of multiple sources of evidence the researcher must also define the order in which the data is collected. In case of this research the author already proposed interviews as the main source of evidence within this case study. Although it is recommended to use multiple sources of evidence the conduct of qualitative interviews is sufficient for the scope of this master thesis.

Following the author lists the six different sources of evidence described in [94]:

- Documentation
- Archival Records
- Interviews
- Direct Observation
- Participant-Observation
- Physical Artifacts

Since this research uses interviews as the main source of evidence, this type is investigated in detail in the next part of this thesis.

The interview is a very important source of evidence in case study research. MacNealy states about interviews: "An interview can help a researcher gather facts, opinions, goals, plans, and insights that may not be available from any other source." [92, p. 189]

In contrast to quantitative data collection methods like surveys, interviews are based on guided conversations rather than structured queries. This means that in interviews the interviewer must adapt to the answers of the interviewee in order to keep up a good conversation. An implication of this fact is that an interviewer has two jobs during an interview: 1) the interviewer should follow the line of inquiry as defined prior to the interview and 2) the interviewer should ask unbiased questions which serve the needs of the line of inquiry.

Yin describes in [94] different kinds of interviews. One type of case study interviews are *in-depth* interviews. In such interviews questions about facts of a matter as well as opinions on events can be raised. In some cases questions regarding the opinion to a specific topic can be asked. The insights given by the interviewee may later on be used as a basis for further investigation. In most cases such in-depth interviews take place not only once, but rather more often in order to get the best possible result.

Another type described in [94] are *focused* interviews. Focused interviews in contrast to in-depth interviews are conducted over a short period of time, e.g. an hour. The interviews are still open-ended in this type, but usually they follow a more rigid protocol like a set of certain questions to ask. This type of interview is used in situations where researchers corroborate facts which are proposed by the investigator. This means that the interview is targeted on specific topics chosen by the researcher and not about topics in an open-ended and broader nature. The purpose to corroborate facts during the interview implies that the questions asked during the interview must be carefully worded in order to allow the interviewee to answer without any influences by the researcher. Only this way the purpose can be served.

Formal *surveys* are a third type of interviews mentioned by Yin [94]. This type of interview is based on more structured questions and is targeted on the creation of quantitative data. It is very similar to regular surveys but the outcome is not directly taken as a valid measure but rather as just one component in the overall assessment of the topic under investigation.

Besides the fact that interviews are an essential source of evidence in case study research, researchers must not forget that all kinds of interviews remain verbal reports, which means that the information gathered during the interview is solely based on the feedback of the interviewee which may be biased and inaccurate. A reasonable countermeasure against this situation is again the use of other sources of evidence in order to corroborate the findings.

Another important topic concerning interviews is the question how to collect the evidence and whether to record the interview or not. Yin [94] states that it is a matter of personal preference, but nevertheless the recording of an interview provides the most accurate form of data collection compared to any other method. Still the interview should not be recorded in the following scenarios:

- the interviewee refuses the permission or seams uncomfortable with the recording
- there is no plan for transcribing or systematically listening to the interview afterwards
- the researcher is not experienced in the use of the audio recorder and therefore it would distract the interview itself
- the investigator does not follow the interview accurately because of the fact that the interview is recorded

5.3.2 Execution of Pilot Case Study

The next and also last step in this phase of the case study is the execution of the pilot case study. In general the pilot case study helps the researcher to check if the defined research propositions are still meaningful and that the questions asked during the interviews still focus on these propositions. Additionally other problems during this pilot case study can be addressed and solved for the actual case study. [91]

Based on the idea of a pilot case study an interview prior to all other interviews was conducted. This interview cannot be compared to a complete pilot case study because the scope of this interview is per definition not enough to count as an actual case study. Nevertheless this approach was taken based on the concept of a pilot case study advised in [94].

As stated in section 5.2.6, the organizations Alpha and Beta were selected as pilot cases. The insights gathered through the interview helped the author to integrate real life experiences into ALP-Mobile. Because of the fact that the interview was used as additional input to ALP-Mobile, the process model was not completely finished by the time of the interview and therefore no references to ALP-Mobile were made in course of this interview. Additionally the author wanted to receive feedback without having a bias regarding concepts of ALP-Mobile. Especially in this case, a review meeting after the completion of the case study was offered to the organizations Alpha and Beta in order to present the final results of this research.

In the following the author presents the main results of this interview. Therefore quotes[1] regarding the concepts of ALP-Mobile are displayed and investigated. Since the scope of the inquiry in a pilot case study can be much broader than in the actual case study, the questions asked during the interview were not as specific as the questions in the remainder of the interviews. After the analysis of the main results, the author presents the implications of the interview on ALP-Mobile.

Agile Processes in Mobile App Development

The first topic concerned the use of agile processes within mobile application development. The author asked the interviewees if process models in general are important in their projects. The CEO of Alpha differentiated between the realization of customer projects and the development of own products in their organization. According to the expert (Alpha) it is a lot easier to implement an agile software process in the development of own products, because there are no interferences with the customer.

> "*For us there is a difference between customer projects and our own products. The fact that there is a given budget as well as a requirements analysis in customer projects changes the development process for us. This fact is really exciting, because with our own products an agile software process is much easier and simpler to implement than with customer projects. In customer projects, the iteration highly depends on the customer and the okay of the customer. (...) With fixed price projects I can not begin to improve a product, before I do not have the okay from the customer.*"
>
> Alpha:#1

[1] All quotes are listed in the appendix of this thesis. Due to the fact that the interviews were held in German, the author translated the quotes presented in course of the case study.

The CTO of Beta replied that well-defined processes are necessary at Beta, because the organization offers medical products and therefore they need a quality management according to ISO 13485.

> "*For us process models are very important. Perhaps not process models directly, but at least the well-defined processes. The whole issue of the process of the development is very important for us, since Beta as an organization is ISO certified. We have a quality management according to ISO 13485 which is needed in order to offer medical products. (...) From this perspective, the whole subject of software development processes is very important for us.*"
>
> Beta:#1

The interviewees were further asked if they think that agile respectively lean processes are helpful in mobile projects. The CEO of Alpha pointed out that agile processes are especially helpful after the first release of the app in order to resolve a high number of reported issues.

> "*Taken a retail app which is located and sold in the app store. After the first release I get a lot of support requests that features are missing or that errors occur. Then I resolve those issues in sprints or iterations as in classical agile software projects. This makes total sense.*"
>
> Alpha:#2

The CTO of Beta was also the opinion that agile processes make sense, but he also mentioned the fact that it is possible to lose track in the development when using agile processes without having a clear vision regarding the final product.

> "*Makes perfect sense. However, I have had the experience that one could get lost using the agile approach when developing a product where the result is not entirely clear. Where I do not make a product that simply copies another product or where I do not go a specific path that is already clearly foreseeable, but where I'm more in an experimental stage and I don't exactly know how the final product will look like. You just start to develop, implement some stuff and then you discover that it should have been different. Especially in the mobile area, in particular with iOS apps. With web services I can make an update at any time in principle and thus deploy minimal changes. Several times a day if I want to. This means that I can be agile, not only in development but also in the deployment. This is different with mobile apps. With Apple for example I have a weeklong review process. Therefore I have to plan accordingly. The idea of small improvements in agile development is a lot harder to implement in such an environment, compared to web applications for example.*"
>
> Beta:#2

Prototyping

The next topic in the interview focused on the design of mobile applications. The author was interested if the organizations make use of UI prototypes within in their projects. The CEO of Alpha stated that they use low-fidelity and high-fidelity prototypes in the development.

> "*We use two types - low-fidelity and high-fidelity. Low-fidelity are classic Balsamiq mock-up prototypes at the beginning and high-fidelity is what we get from our designer, namely click-through prototypes. (...) Whenever we do projects, we always*

> *get totally cool click-through prototypes that almost look like an app. That makes it difficult with the customer, because then he thinks that the app is already finished, although we still have to implement the functionality. This makes it a bit difficult. Those are the two types we have at the moment.*"
>
> Alpha:#3

The CTO of Beta pointed out that unfortunately they do not use click-through prototypes a lot. He also said that because of the fact that they solely focus on their own products they have built up a relatively large community which is asked for advice when it comes to design decisions.

> "*Unfortunately we do not work a lot with clickable prototypes. We do use paper prototypes, usually they are digital. So we have both, the quasi hand-drawn as well as the ready-designed prototypes. We also have the following: Since we specifically work on our own products and not on changing customer projects, we have built up a relatively large community of users over time which we systematically keep asking for advice. We have set up a forum where we post screen designs or other designs. By doing that we try to find out, which features are needed most.*"
>
> Beta:#3

Distributed Development

By asking the experts on their opinion regarding distributed development the author wanted to investigate the impact of this concept in the area of mobile application development. The CEO of Alpha explained that they have two developers working in Tyrol, which works without problems. Still the face-to-face contact must not be underestimated.

> "*We have two developers who work in Tyrol. This works smoothly from a software development technical perspective. It is different from a business point of view. We have three Android developers and they are always happy when the Android developer from Tyrol comes, because then they are finally able to ask him directly. (...) Just based on the software development currently it works very well.*"
>
> Alpha:#4

The CTO of Beta also stated that distributed development works very well from a technical perspective. He sees this concept as very common in the area of start-ups, but he also points out that there is a lack of social interaction in this concept, which is not good.

> "*Especially in the start-up area you've always people working remotely, because often for different reasons there are no other ways. My observation is that the software development works from a technical point of view, but the social aspect is lacking. (...) I personally think it's always good if the people who work do not only participate technically, but also emotionally.*"
>
> Beta:#4

Testing of Mobile Applications

Based on the literature research performed in the beginning of this research the author observed that concepts like automated testing and continuous integration are not very common in the development of mobile software. Therefore the interviewees were asked about their opinion regarding

testing in mobile app development. The CEO of Alpha replied that their test processes are not very structured. According to the CEO the limited budget is a reason for this situation.

> "*We actually do not test very structured. Everyone in the organization has a device for testing and tries to find as many errors as possible. Then these bugs get fixed. This is our current approach, due to budget reasons*"
>
> Alpha:#5

Due to the fact that Beta develops software in the medical sector, testing is very important for the organization. With the wide range of mobile devices especially in the Android development the CTO of Beta explained that they invest time in the development of automated tests.

> "*We have a lot of structured tests. We have a test team that currently is still too small. Our target is that we have one tester for every two developers. Testing is very important for us. There are detailed test protocols. We organize test cases where each version has to go through a specific test protocol before it is allowed to go live. And lately we increasingly try to go towards automated testing, especially in the Android development, driven primarily by the problems with different devices and different screen sizes. The devices behave sometimes quite different. We try to overcome those issues by investing time in the development of automated tests. We then test automated in simulators or on physical devices by going through the app simulating the user.*"
>
> Beta:#5

The CTO of Beta stated further that a subset of these tests is executed with every commit to the repository which in fact is similar to the concept of continuous integration.

> "*The builds that come in already start tests, but only a basic set. So we do not run the build server with 40 Android devices on each commit. We do this with every release though. It would take too long and it would be too much effort. So in a small form we have continuous integration, but not full-scale.*"
>
> Beta:#6

Implications on ALP-Mobile

The first topic in the analysis of the pilot interview concerned agile processes in mobile app development in general. The CEO of Alpha pointed out that there is a difference between customer projects and the development of own products. According to the CEO it is a lot harder to implement agile processes in customer projects, because the development highly depends on the customer. Because of this fact ALP-Mobile is created for the use in customer projects. Since this approach is more difficult, ALP-Mobile can also be used for the development of own products without a lot of adaptations.

The experts stated further that agile processes in general make total sense in the development of mobile applications, but because of the parameters of the environment it is sometimes hard to follow agile methodologies in this field of software development, e.g. the necessary review process of app stores in the deployment. As an answer to the decrease of agility in the deployment the author implemented the columns Pending and Distribution on the kanban board. Those queues help to adapt to the specific conditions of the distribution phase in mobile projects.

Regarding the topic of prototyping the interviewees described that there are low-fidelity and high-fidelity prototypes. Especially high-fidelity prototypes, e.g. click-through mock-ups, provide a

very rich user experience, which is great to demonstrate functionalities of the future application. Concerning ALP-Mobile the author therefore prescribed the activity of early prototyping in the definition phase of the process model.

According to the experts the concept of distributed working works very well from a technical perspective. Nevertheless the lack of social interactions is a present problem of distributed working. Therefore the author provided guidance for the use of ALP-Mobile in a distributed environment. By doing that people who want to use ALP-Mobile in a distributed environment receive valuable support.

The last topic in the analysis of the pilot interview concerned the testing approaches executed in the organizations. The approaches perfectly matched to the current situation in mobile app development. Often apps are tested solely manually, but concepts like test automation and continuous integration are desirable. The implication on ALP-Mobile is the introduced testing concept in which manual testing is prescribed and the concepts of automated testing and continuous integration are preferable but not mandatory.

5.4 Data Collection

The next step in the case study is the actual collection of the data. The author already briefly discussed the data collection strategy used in the course of this research. As stated, qualitative interviews with experts in the field of mobile application development were executed in order to determine the strengths and weaknesses of ALP-Mobile. As presented in the previous section, there are different types of interviews. For this research the author chose the approach of focused interviews. By choosing this approach the author was able to ask open-ended questions about topics concerning ALP-Mobile, but through the use of a prepared question guideline the interview followed a more rigid protocol.

The appendix lists the complete set of questions prepared for the interviews. The questions are intentionally open-ended in order to get the most information about the requested topic. As stated, it is often not easy to find possible cases in case study research. Especially the time of the participants is a critical factor. Therefore the author chose the approach of focused interviews, because with this approach the interview only lasted around one hour. The time consumed in an in-depth interview approach is much higher.

Yin states in [94] that one principle of data collection is to collect multiple sources of evidence. A point of criticism regarding this master thesis is the use of only focused interviews in the case study part. This can be reasoned with the fact that the resources for the master thesis are limited and that it is sufficient for the scope of this research to use just one source of evidence.

5.4.1 Interview Topics

The following list presents an overview about the main topics discussed within the interviews. Additionally the author provides a short description to each of the topics.

- Process models in general
  The first topic in course of the interviews concerned process models in general. The interviewees were asked if process models are used in their organizations and what the reasons for the use of this models are.
- Mobile application development
  The next topic discussed within the interviews was mobile application development. The

author was especially interested in the differences between mobile app development and software development in general.

- Roles
  After the initial questions regarding the organizations, software process models and mobile application development in general the author presented ALP-Mobile to the interviewees. By introducing ALP-Mobile early in the interview the author had the chance to base the remaining questions of the guideline on ALP-Mobile and make direct associations. The next topic in the guideline concerned the various roles implemented in the organizations.

- Meetings
  After getting into more detail about the various roles implemented in the organizations the author asked the interviewees about the meetings established in their processes.

- Requirements Engineering
  The next topic in course of the interview was the activity of requirements engineering. Since this activity is very important in the definition phase of ALP-Mobile, the author asked the experts about their opinion regarding the task of requirements engineering.

- Development
  The development phase of ALP-Mobile was the next point in the guideline. The questions asked concerned the development approach applied in the organizations. The author was especially interested in the opinion of the interviewees regarding the concept proposed in ALP-Mobile where fixed iterations in the development phase are waived in contrast to a defined sprint length prescribed in scrum.

- Distributed Working
  The author also addressed the topic of distributed working within the interviews. The interviewees were asked about their experiences with this technique.

- Testing
  Based on the fact that testing is an important activity in ALP-Mobile, this topic was also an important part of the conducted interviews. The experts were asked how they test their applications and where they see problems concerning testing in this field of software development.

- Distribution
  The last topic of the focused interviews concerned the distribution and the maintenance of the software. The interviewees were asked where they see problems in this phase of the project lifecycle and also how they handle new versions of operating systems which require an update of the software.

5.4.2 The Interview Process

Regarding the interview process itself, the author contacted the participants via telephone after an initial Internet research. The author already introduced the interviewees in course of the presentation of the selected cases. The interviews were conducted at the offices of the interviewees. As stated, it is good practice to record the interview in case the participants agree to the recording. The recording is used for transcribing the interview afterwards. Additionally to the possibility to transcribe the interview, the researcher does not have to take notes during the interview, which helps to fully concentrate on the conversation with the interviewee. In case of this research all participants agreed to the recording of the interview.

5.5 Data Analysis

The last phase of the case study is concerned with the analysis of the data. In order to analyze the collected data the author uses the topics presented in section 5.4.1. For each topic the author presents quotes from the focused interviews which display the opinion of the interviewees regarding the specific subject.

To further analyze the findings a cross-case analysis about the opinions of the experts regarding each of the topics is performed in the next chapter of the thesis.

5.5.1 Gamma

The following sections present the results of the interview conducted with the CEO of Gamma.

Process Models in General

Concerning process models in general the expert stated that the organization Gamma follows an agile software development approach using scrum but without all artifacts which come with an implementation of scrum.

> "*From the beginning on we used agile software development, but without the overhead a full-blown-implementation of scrum brings. (...) Basically we are using a lightweight version of scrum with weekly cycles.*"
>
> Gamma:#1

Mobile Application Development

The CEO of Gamma sees the technological environment and the market as the major distinctions to software development in general. He pointed out that the market, the technology and the environment are changing rapidly and that these factors cannot be influenced by the organizations developing mobile software.

> "*The biggest differences are the technological environment and the market. On the one hand there are traditional IT systems like control systems for aircrafts or ERP systems with a lifecycle of decades and on the other hand there are apps with a lifecycle of a few months. This means: the market, the technology and the environment are changing constantly and* [those factors] *are also outside the influence.*"
>
> Gamma:#2

Roles

Compared to the roles prescribed in ALP-Mobile, Gamma follows a quite similar approach. They have a sales role which is similar to the product owner presented in ALP-Mobile. Additionally they have the role of a project manager which can be compared to the agile coach. Besides these two roles they also implement the role of the development team. A real difference to the roles in ALP-Mobile is the fact that Gamma has an extra quality assurance department which is responsible for testing.

> "*We use Sales / PO who is responsible for the content and for the budget. We use PM / scrum master. We call the scrum master project manager - perhaps less fancy, but he does just that: he ensures that the project is running and that it gets delivered. Yes and then there is the designer and the developers on the target platforms. The QA team is involved, as that the QA team leader will be notified when a project is there. If user stories need to be created, then he* [QA team leader] *sets that in motion. He also plans with his team when the test team is available.*"
>
> Gamma:#3

Meetings

According to the CEO the meetings at Gamma are also quite similar to the meetings in ALP-Mobile. In essence they have a kick-off meeting, daily stand ups and a weekly meeting which combines estimation, review and retrospective. The reason for the combination of estimation, review and retrospective into one meeting is that the organization wants to reduce the meeting overhead which comes with a full implementation of scrum. Furthermore the main idea behind the weekly meeting is to get instant feedback from the customer which helps to successfully develop the expected application.

> "*We do not have as much meeting overhead as it is intended in principle. This means that we do not do a retro in every sprint or a long planning 1 respectively planning 2, as scheduled in scrum. Due to the shorter cycles this is not necessary. (...) The greatest effect in a positive sense a weekly sprint meeting has, is that we really try - it does not have to be every week - to invite the customer as often as possible to this meeting in order to get immediate feedback. This way the expectations of the customer can be managed very well.*"
>
> Gamma:#4

Requirements Engineering

Regarding the activity of requirements engineering the CEO of Gamma pointed out that this task highly depends on the experiences of the customer with mobile applications. Nevertheless they have always a specific workflow before a feature gets implemented. This workflow contains the creation of wireframes of the whole app, including all screens. Additionally they have a confirmed design of each screen prior to the actual development.

> "*This always highly depends on the customer we have to deal with. There is a huge difference whether the customer has experience with mobile projects or not. Basically the customer generally has more or less concrete ideas about the outcome of the project. We then formulate those ideas. We have one to X meetings with the customer, where we ask for details. What is the objective? What should be achieved with the app? (...) Before we start with the development of a feature in the app, there is a workflow: We have wireframes of the entire application, each screen. There is also a confirmed design of each screen before it is implemented. This goes that far that we use various tools - we always try new ones* [tools] *- to design it. We already made useful experiences with online click dummies. Designs which are linked to each other and where the customer is able to click through the designs. If that's okay, it gets developed.*"
>
> Gamma:#5

Development

Concerning the development process the CEO of Gamma described that the process in their company is very similar to the process presented in ALP-Mobile. Besides the fact that they have weekly iterations in the development, which already increases the flexibility, the goals for the weekly iterations are not fixed either, which means that work items can be added to the sprint backlog on-demand. The author asked the CEO about his opinion regarding the concept in ALP-Mobile in which fixed iterations are waived in the development. He replied the following:

> "*I think that's hard to say. If the iterations take three or four weeks, then for me it would be way too inflexible. With weekly iterations I don't really see the need for non-fixed intervals. (...) The thing is that the scope is also flexible in our sprints. That means that it can be more or less depending on the circumstances.*"
>
> Gamma:#6

Distributed Working

The CEO sees the overhead in the coordination as a main problem in distributed working. He pointed out that you have to invest a lot more in order to get same outcome.

> "*If your team is distributed, then your work must be much better in order to have the same results and the same project success as compared to a non distributed environment. The communication has to be top notch. (...) It all has to be better documented. It starts at the beginning: the specification must be much clearer and misunderstandings are discovered much later. It can work, but you have be very, very good in the management in order to work nearly as well. And people have less fun.*"
>
> Gamma:#7

Testing

Concerning the testing of mobile applications, the expert pointed out that it is important that people test the application who did not develop it. At Gamma the developers still test their own application, but they also have a dedicated testing team which is responsible for testing.

> "*Basically each developer is constantly testing. The developer who implemented something then has to see if it works in principle. That means, he must constantly test anyway. It is very important that not only the programmers themselves test, but also someone else who did not develop the part of the application because it is natural that you get routine-blinded and therefore you don't notice a thing. We have a test team. There are currently three people who test, but also not full-time all the time, because that is not possible. I wonder if there are people who endure such a thing. I think if you want to test in a good quality, then that needs very, very much attention. And to do that for a whole day, I think this is practically impossible.*"
>
> Gamma:#8

Regarding automated testing the CEO of Gamma said that due to the fact that they implement a lot of customized solutions the overhead for the creation of automated tests would be unreasonable.

> "*We have had a look at it, but for us it does not make sense, because the overhead is too big. We make very customized solutions in the smallest things. In order to automate such things we would need to invest a lot of work in the beginning which is only useful for a small area. (...) We didn't invest a lot of time though.*"
>
> Gamma:#9

Distribution

According to the CEO the necessary review process at Apple is an unknown in the distribution of the software. He pointed out that it is important to communicate this fact with the customer in order to avoid any problems with the schedule of the project.

> "*You have to plan that you need two weeks for the review at Apple. Then it can still happen that Apple says no, because an employee is just in a bad mood that day. We simply have no control over that. (...) It is possible to miss an important deadline because of that. The reasons why it is being rejected are clearly not traceable. (...) And because of that you have for example lost three weeks of time to market just because someone has thought ... well, not today. So that's an unknown. Therefore it is very important that this fact is communicated to the customer.*"
>
> Gamma:#10

5.5.2 Delta

The following sections present the results of the interview conducted with the expert of Delta.

Process Models in General

The solution manager explained that the mobile department of the organization used to be very small when he joined the organization. In the beginning they had in reality just one solution manager, five developers and one tester. To control their processes they used the plan-based waterfall process model. The expert stated further that they try to improve things concerning this topic. They moved towards agile processes, in particular currently they use scrum as a process model.

Mobile Application Development

Concerning the differences to software development in general the expert stated that the increased need of flexibility is one of the biggest differences to mobile application development.

> "*The big difference is that we experienced that we should react much more flexible. We have never driven this rigid system. At the beginning we followed the waterfall model where we first made the specifications. Then we implemented it and then it got reviewed. Our experience showed that as soon as there was a trial version which we showed the customer, change requests were instructed, where we had to be more flexible.*"
>
> Delta:#1

Roles

The expert explained that they have the role of the solution manager which is mainly responsible for the activity of requirements engineering. In most projects they also have the role of the project manager which is responsible for the budget of the project. The expert noted that the bigger the project the more people they use for each of the roles. Besides the presented roles they have developers and testers. Due to the fact that they use nearshoring for the classic app development the roles implemented in the organization cannot be compared adequately to the roles prescribed in ALP-Mobile.

Meetings

Delta tries to implement the meetings as prescribed in scrum. Due to the fact that they outsource the GUI development of the app, the meetings are more concerned with the backend and interface development. A difference is that the organization does not perform daily meetings. Instead those meetings are held on an on-demand basis.

> "*We currently try to use scrum, but more in the direction of the backend development. Since we have outsourced the client GUI development, there are not the traditional dailys, but rather on demand. We certainly have daily contact, but not the classic scrum questions, but rather in the direction of stories or current assignments or tasks.*"
>
> Delta:#2

Requirements Engineering

At Delta the requirements engineering is based on an early prototype. Similar to ALP-Mobile the outcome of the requirements engineering is a set of user stories which describe the software.

> "*In case a project is commissioned, then there is already a prototype and then this prototype is the basis for the requirements engineering. Then we try to break it down into use cases respectively user stories and then we define the user stories.*"
>
> Delta:#3

Development

The author asked the interviewee regarding his opinion about the development concept introduced in ALP-Mobile which highly makes use of the kanban board as a tool for the process execution. Regarding the elimination of fixed iterations the expert sees advantages and disadvantages in this approach. He noted that if changes to sprint backlogs with iterations of one week are necessary then there are probably problems in the sprint planning respectively the requirements engineering.

> "*There are advantages and disadvantages. What I really liked was the point that you have a column* [in the Kanban Board] *for bug fixing provided. We see that this is necessary for us in each sprint. Even if there are little things that we need to take into the next sprint. I like that. If the development has already started and you do weekly retros, then in practice I believe you should watch out earlier in case that user stories change in such a short period. Those user stories should not be available for*

development in such cases."

Delta:#4

Distributed Working

Due to the fact that Delta uses nearshoring for the GUI development of their mobile applications, they have experiences in the area of distributed working. The solution manager stated that they prefer nearshoring in Vienna, because in that case the people working in the outsourced organizations also work part times directly at Delta. They already used a big sourcing organization in India. Although the development is sometimes faster in such big sourcing organizations, the flexibility you have with a local partner is much higher. Concerning misunderstandings in the communication the expert pointed out that sometimes the language barrier can pose a real problem in a distributed environment. The effort to clear misunderstandings must not be underestimated.

"*Although language shouldn't be a barrier nowadays, I personally have encountered the situation in which misunderstanding occurred if the specification was written in English. Such misunderstandings bring the need of teleconferences in order to define what is desired. This takes a lot of time.*"

Delta:#5

Testing

The expert said that they have to distinguish between interface and frontend testing. The interface testing is closely bound to scrum and the frontend testing is performed with the help of test catalogs. At the moment the testing is performed manually.

"*Again we must distinguish between interface functionality and pure mobile functionality. When testing the interfaces we are bound to scrum. That means, before a user story is finished this story will be tested by us and then there is the okay from the product owner. If the application to test is a pure client-side development, then there are different iterations of test versions and these test versions are tested based on a test catalog. This means that we have our tester who is responsible for the creation of a test catalog based on the user stories. The app developer then tells us exactly the features implemented. These features are then tested with each version.*"

Delta:#6

Regarding the topic of automation, the solution manager noted that they see great potential in the use of automated testing processes in order to increase their quality, but the lack of proper solutions on the market makes it hard to implement such automated processes.

"*This is an interesting topic. We will focus on this topic soon, since we see a lot of potential to improve our quality. To test faster and more accurate, but there are not so many products respectively providers on the market that offer this functionality. We are currently investigating how to proceed with these concepts. It is an issue, but for now testing is performed manually... I think there are a lot of proprietary, customized solutions. We are looking for something that covers everything. In other words we don't want a framework for iOS and a framework for Android. We want something more sustainable which covers different or all platforms of our apps.*"

Delta:#7

Distribution

When it comes to the distribution respectively the maintenance of the software the expert noted that beta versions of the operating system offered by the operators are very helpful in order to release an update of the application soon after the release of the new operating system. Regarding the updates of the operating systems he stated the following:

> "*This is a classic topic for us. It is not too complicated, because the developers or Apple provide us with beta versions of the operating systems, and therefore we know the changes soon enough. Thus we can react accordingly. Recently we did the update from iOS 6 to iOS 7. It should always be timed the way that on the day where Apple publishes the update of the operating system, we release our update of the application.*"
>
> Delta:#8

5.5.3 Epsilon

The following sections present the results of the interview conducted with the CTO of Epsilon.

Process Models in General

The organization Epsilon does not make use of any particular process models. This is reasoned with the fact that the projects realized within the organization are small sized and therefore the execution of such processes would be an unnecessary overhead according to the CTO.

> "*No, we do not really use any* [process models]. *Of course there are guidelines on how to do something. Ultimately, the projects we realize are in the magnitude of 20 man-days. 50 percent of the projects are developed by one developer, the rest by a maximum of two* [programmers] *simultaneously. Therefore it would be unreasonable to define any processes.*"
>
> Epsilon:#1

Mobile Application Development

The next topic concerns mobile application development. The author asked the interviewee where he sees the differences to software development in general concerning the use of process models. The CTO pointed out that Epsilon focuses on rather small projects and therefore they have a lot of specific tasks for which it is hard to define standardized processes.

> "*In contrast to larger projects, our projects can be classified as rather small. There are no real processes and the developer has to work independently. It would not make sense to define any developer processes, because the overhead of this task would be too high. We have so many special cases - to account for this, we probably would need a process manual of 500 pages which then no one would have a look at. That wouldn't bring anything. Ultimately, for us it is essential that we finish our smaller projects that of course make less contribution margin very quickly. There should be very little troubles and therefore the whole thing is done as quickly as possible.*"
>
> Epsilon:#2

Roles

After the presentation of the roles prescribed in ALP-Mobile, the CTO stated that because of the fact that their projects are rather small and that the developers work very independently there is no need for a specific role of an agile coach in their team. Nevertheless they have the role of the project leader which is comparable to the product owner introduced in ALP-Mobile. The project leader is responsible for the functionalities of the applications and furthermore communicates with the customer. Besides this role they have developers who develop the mobile applications.

Meetings

The organization Epsilon follows an on-demand approach concerning the meetings which take place in the organization. Besides the kick-off meeting in the beginning of the project they do not really have any scheduled meetings. Still, they have a weekly progress evaluation in which the developers report their current status regarding the implementation. Based on the question if the organization implements any meetings besides the kick-off meeting the expert replied:

> "*There are no other meetings. It would not help if the developer who has a problem in the afternoon has to wait until the next day in the morning at the meeting to discuss the encountered problem. Does he have to wait for the rest of the day? Therefore, it is better for me that he comes when he has the problem and not waits until the next meeting.*"
>
> Epsilon:#3

Requirements Engineering

According to the CTO of Epsilon this task highly depends on the customer. Some customers already know exactly what they want and other customers only know that they want to have an app. The latter option implies more work for the organization. Nevertheless, either way at the end there is a requirements document which defines the future application.

> "*This highly depends on the customer. Some customers know exactly what they want. Usually they send us a document with the design. Especially larger organizations already have finished screens from a designer. This goes pretty quickly then and it is also possible to determine the costs quite accurately. Other customers in turn just have the idea that they want to have an app. They show us their website and expect from us a description of what they will get. These customers actually do not know what they want and they leave it to us to define it. We then present how we imagine the app and what features the app will have. The customer then only says yes or no to each of these features and this results in the requirements document.*"
>
> Epsilon:#4

Development

Concerning the development the author asked the expert about his opinion regarding the different approaches in the development of mobile applications. The CTO explained that they have a framework for hybrid apps which is used for the visualization of static HTML content within the app.

> "*We have a framework for the development of hybrid apps which loads static HTML code from a server. The content is zipped, unpacked in the app and then the content is updated. In principle I do not like hybrid apps. That's why we only use it very sparingly. We use it when we need to render HTML content. In such cases it is about the minimal effort for the creation of the app.*"
>
> Epsilon:#5

Distributed Working

The organization Epsilon works partly with freelancers. According to the CTO self-organization is a big requirement for these freelancers. In most cases the freelancers work completely independent and they only ask for help if they do not know how to implement some specific tasks.

> "*Our freelancers must work independently - even more than our own developers. Therefore we do not really hear a lot from them. They only come when they face any ambiguities and therefore they just do not know how to implement something. But besides that, you actually never hear from them.*"
>
> Epsilon:#6

Testing

According to the CTO the testing in the organization Epsilon is mainly performed by developers who are assigned to test a specific application. On the question how much they test the expert answered the following:

> "*Actually, very little in house - someone will be determined who is responsible for testing the app. This is not very complicated in principle. He tests the application in order to see if there are any obvious bugs. Then the app will be made available to the customer who usually has different and multiple devices to test the application. These are pretty much the tests we have.*"
>
> Epsilon:#7

Concerning principles like test automation and continuous integration he stated that because of the variety of the applications it would not make sense for the organization to set up automated testing. Furthermore he noted that nobody wants to pay for testing and therefore it is not easy to invest parts of the budget on this activity.

> "*For us I think it would not make sense, because after all every app is different. To determine for every input field which values need to be entered in order that on another screen something happens is too much effort. We rather try this by hand, before setting up a dedicated test machine. (...) The problem is that nobody pays for testing. The customer just wants to have an app, and assumes that it is error-free programmed and if now 50 percent of the price of the app would be used for testing, then he* [the customer] *would probably hire someone else.*"
>
> Epsilon:#8

Distribution

Concerning the distribution of the mobile applications the CTO of Epsilon does not see a lot of problems. Regarding the review process at Apple prior to the distribution in the app store the expert pointed out that it is a bit tedious, but in the end you just have to plan the project accordingly.

> "*The only tedious thing is that Apple needs the alleged two weeks for the review - we always plan two weeks for the app to be accepted. One must also not forget that at Christmas the app store is closed. That is a bit tedious in the planning, but otherwise this is not really a problem. (...) In most cases the customer is aware that a review process is required at Apple, and therefore an understanding is often already available.* "
>
> Epsilon:#9

5.5.4 Zeta

The following sections present the results of the interview conducted with the Android tech lead of Zeta.

Process Models in General

Concerning process models in general the organization uses scrum, but as the expert declared, they do not manage to execute scrum in the complete form.

> "*Mostly we try to use scrum, but we have difficulties to implement it completely. We live in an adapted scrum-world that fits best for our team.*"
>
> Zeta:#1

Mobile Application Development

Regarding the differences of mobile application development to software development in general the interviewee sees the fast pace of the mobile market as a major distinction between those two fields of software development. Especially the desire of the customer to make adjustments very late in the development phase of the product was often experienced in his organization.

> "*Fast pace I think is the biggest point. Also the agility that you need to react quickly. It is often the case that features get traded late* [in the development]. *That means that the customer has relatively long time to decide on the appearance of his app. Features are often removed, because the customer wants to invest more time respectively money on a different feature.*"
>
> Zeta:#2

Roles

The expert pointed out that they have a very similar approach regarding the roles as prescribed in ALP-Mobile. The main difference is the fact that the scrum master respectively the agile coach is no separate person, but rather a member of the development team.

> "*We also have the product owner, the development team and as a coach the scrum master who is responsible for very similar tasks. He moderates the daily, the sprint plannings and the retrospectives, but in our case he is also always part of the development team. In principle, our roles are very similar to the roles that you have presented.*"
>
> Zeta:#3

Meetings

The meetings established in the organization Zeta are pretty much the same as the meetings presented in ALP-Mobile. They also have a kick-off, daily stand ups, retrospectives and planning meetings. The difference is that the planning meeting is executed even before the kick-off meeting takes place.

> "*Before the kick-off meeting we have a proposal phase in which we have a meeting for the estimation of the project. The estimation is performed by our developers, ideally by the developers who also implement the project. Usually the product owner and one or two developers from the development team sit together and estimate the features. This is also a small difference to your presented model. We have the estimation respectively the story point distribution and the T-shirt sizing prior to the project, because we need it for the initial proposal.*"
>
> Zeta:#4

Requirements Engineering

Concerning the activity of requirements engineering the interviewee stated that the product owner clarifies the requirements of the future application with the customer in a workshop or in one or two meetings. After this clarification the estimation phase begins in which user stories are created and estimated. As pointed out in the previous section, the requirements engineering already happens before the actual kick-off meeting of the project.

> "*A customer wants to have a software. That means that he already roughly knows what he wants. Usually the PO then sits together with customer in one or two meetings or perhaps in a workshop. The goal is to clarify what the software is about respectively what the customer wants from us. If this requirements engineering is completed, then it goes into the estimation phase with the developers internally without the customer, only with the PO. In this phase, we are creating the user stories.*"
>
> Zeta:#5

Development

Regarding the concept of no fixed iterations in the development the interviewee stated that due to the fact that they have weekly iterations in their process, which is an adaption to the usual sprint iterations advised in scrum, they are already more flexible in the development. Nevertheless he pointed out that with the complete waiver of fixed iterations the flexibility can be increased even further.

> "*I think the idea is actually not bad, but since we have to sit down by fixed points anyway and since we do retrospectives and plannings, it is not necessary for us. In*

> *principle, the planning is the meeting in which the PO can decide which features get implemented. The advantage I see in your model is that he* [PO] *can even better and faster decide which work items he really wants to have.*"
>
> Zeta:#6

Distributed Working

The organization has experiences in the field of distributed working. The expert explained that, especially in agile processes where daily contact is very important, this concept is not very easy to execute. The coordination must be very strict in order to be successful.

> "*Especially with very agile processes, or processes with the kanban board which depend on the personal contact on a daily basis it is very hard. To work with another organization respectively department is like working with a customer. You need to be more strict. You have to divide a lot more precisely. One can not easily map dependencies. For example someone from the other team gets sick and you depend on this person - then perhaps the whole team can't continue to work. It is much more difficult and requires much more communication overhead.*"
>
> Zeta:#7

Testing

Besides testing performed by the developers and the customer the PO tests the application in course of the weekly meetings at Zeta. The Android tech lead confirmed that concepts like automated testing are very interesting especially with applications which have a longer lifetime than small mobile apps. Unfortunately the organization does not have a lot experiences with this approach.

> "*In our mobile development the developers test directly. Additionally the PO, who tests the applications during the weekly demo meetings, takes a decisive part in the testing role. He really tests the application from a customer perspective. Furthermore the customer tests the application with test versions. Unit testing is also an interesting story - build server that automatically run unit tests. But we haven't done so much with that. Currently we have a lot of developer testing, a lot of manual testing, writing test cases, etc. (...) Of course it would be a desirable point in long-term software to have an automated testing process.*"
>
> Zeta:#8

Distribution

Regarding the distribution of the mobile application the expert pointed out that Apple is much more complex in the distribution than Android. He states further that Apple requires various certificates even for demo versions of the app.

> "*Apple is much more complex. You can see it even in the demo apps that we ship to the customers. With Android devices you just use the result of a daily programming located in the the bin folder and then you send the APK to the customer. Apple is much more restrictive. You have to create certificates. In cases you use push, then you need your own push certificates... server certificates. That is much more complicated with Apple compared to Android releases in my opinion, but it makes the app also*

> *safer. Because of all the Apple approving processes safety increases and therefore the number of frauding apps is significant less compared to Android.*"
>
> Zeta:#9

5.5.5 Eta

The following sections present the results of the interview conducted with the CEO of Eta.

Process Models in General

The CEO explained that they do not implement any particular process model. They try to go through the phases in a traditional way, but within the phases they follow agile methodologies.

> "*In principle, we do not plan to implement one* [process model] *concretely. So we will not do scrum, lean or six sigma from start to finish. Due to my experiences in my old organizations we try to do something in between. Usually and especially in mobile development we follow the classic process, so we go through the phases roughly how they are planned, but in the phases we try to work relatively agile.*"
>
> Eta:#1

Mobile Application Development

Regarding the actual development of the software the expert stated that the decreased cycle time of mobile applications can be seen as a big difference to software development in general. He states further that although mobile applications are smaller in size, the effort after the release is bigger compared to traditional software. The motto "Make it work and then make it better!" stands for his thinking concerning mobile application development, because apps need to be developed further after the release to fit to new environments like new screen sizes or new versions of the operating system.

> "*What I see in mobile applications is that the cycles are usually much shorter, because usually the development is much more delimited. You have smaller applications which need less work to develop, but they also must be implemented faster. Especially after delivery I see a lot more work than in traditional software, e.g. web applications.*"
>
> Eta:#2

Roles

Compared to the roles prescribed in ALP-Mobile the CEO of Eta pointed out that they also have the role of the development team, but they combine the role of the product owner and the role of the agile coach in one person. This is reasoned with the fact that the newly founded organization is still small sized and therefore the resources for an extra role are not available.

> "*We have the development team, that is clear. The agile coach and the product owner are combined in one person. I know the idea in scrum, where you have an agile coach who has nothing to do with the development and only deals with the process. That I think has to do with the size* [of the team]."
>
> Eta:#3

Meetings

Concerning the meetings established at Eta, the CEO stated that for them daily stand ups would be an unnecessary overhead since the size of the organization is not appropriate. They have a weekly meeting which is the only scheduled meeting at the moment.

> "*No, there are none* [meetings]. *The only meeting we have is a weekly meeting where you virtually have the planning, the retrospective and the stand up all in one. We cannot afford a 15 minutes long meeting every day when there is nothing to talk about anyway. This again I think strongly depends on the size and grows with time.*"
>
> Eta:#4

Requirements Engineering

According to the CEO the task of requirements engineering is really important, because only with a profound requirements engineering the customer satisfaction at the end of the project can be ensured. He stated further that especially in the IT often specific terms mean different things to different people and therefore it is essential to really determine the desired functionalities in order to be successful.

Development

The author asked the expert about his opinion regarding the concept of no fixed iterations in the development phase of ALP-Mobile. The interviewee pointed out that this concept is inevitable in mobile development. For him the iteration length proposed in scrum is not applicable in this field of software development, because usually the projects are small and therefore the flexibility is lost with long iterations.

> "*I think this is essential in mobile. Especially for mobile development scrum cycles with two to four weeks are too long. (...) For projects that are smaller in the scope I think four weeks are too long.*"
>
> Eta:#5

Distributed Working

Concerning the concept of distributed working the CEO thinks that this technique is essential in mobile application development. To the question if he sees distributed working as a valid opportunity in this field of software development he replied the following:

> "*Yes definitely. I think distributed working is inevitable. We do it not only with the resources which are located far away, but also with other* [resources]. *For example our designer is no one with whom we are always working on site and I believe it is now no longer necessary, because the technology is so good that you save a lot of resources with screen sharing, TeamViewer, Skype etc.*"
>
> Eta:#6

Testing

The testing process executed at the organization Eta starts with unit tests on the bottom and user tests at the top of the application. Concerning the size of the project, the CEO also mentioned that in small mobile projects the concepts of automated testing and continuous integration are probably an unnecessary overhead.

> "*We only do user tests, meaning we have no regression or integration testing. We have unit tests at the bottom and we have user tests at the top. I think automated testing and continuous integration is interesting in mobile development, but I see the very short cycles and the small packages as a problem. For example, if you want to execute test driven development in mobile development, then that's usually not worth the effort because you don't have so much functionality that it would be worth it. (...) We cover the usability features with user testing, since you'll never be able to test all those features with automated tests. Regarding the effort I therefore consider automated testing and continuous integration as very questionable for smaller projects.*"
>
> Eta:#7

Distribution

Concerning the distribution of the mobile application the expert pointed out that at Apple you have to plan one up to two weeks where you cannot do anything, because you have to wait for the deployment of the application. Therefore you need to be very careful in the development, since updates take very long. He also stated that this is a very important issue in the project management, because this situation must be communicated with the customer.

6 Discussion

This chapter deals with the discussion of the results of this master thesis. At first the author performs a cross-case analysis about the opinions of the interviewed experts. After that, the research propositions stated in section 5.2 will be examined in detail. Subsequently the author investigates related work of this thesis in order to show the differences of ALP-Mobile compared to other methodologies. Conclusively the author answers the research question of this thesis. In order to achieve that the author focuses on the feedback regarding ALP-Mobile gathered in course of the focused interviews. By investigating this feedback strengths and weaknesses of ALP-Mobile can be derived. The end of this chapter discusses the limitations of this research.

6.1 Cross-Case Analysis

In this part of the thesis a cross-case analysis about the opinions of the interviewed experts regarding the topics presented in section 5.4.1 is performed. Table 6.1 provides an overview over the information gathered in course of the focused interviews.

6.1.1 Process Models in General

Most of the experts stated that they use agile processes within their organization. According to the CEO of Gamma they make use of a lightweight version of scrum. This approach is similar to the approach executed at Zeta. The expert (Zeta) pointed out that they have difficulties to follow scrum in the complete form. The solution manager of Delta stated that they made use of the waterfall process model in the beginning but now they use scrum as a process model. The CEO of Eta explained that they do not make use of any particular process model. They follow the classic process, but in the phases they try to work relatively agile. A complete different approach was proposed by the CTO of Epsilon who said that they do not have any real processes in the organization. This is reasoned with the small size of the projects for which an implementation of a specific process model would be an unnecessary overhead.

Concerning the use of process models in general the author concludes that there is no perfect solution which satisfies every organization. The interviews showed that agile respectively lean processes are a good fit for mobile application development, which is the first indicator that ALP-Mobile can bring real value to this field of software development. Nevertheless as executed by the organization Epsilon, organizations are also successful without using any real processes at all. This fact is probably true for small projects in which the coordination overhead is not as big as in large projects. In the end it always depends on the organization respectively on the projects which processes are the best solution. Another interesting fact was that the used process models were not implemented as prescribed, but rather adapted to the specific needs of the project teams. This is an interesting fact which is also true for ALP-Mobile.

	Gamma	Delta	Epsilon	Zeta	Eta
Process	- Agile (Scrum)	- Agile (Scrum)	- No defined process - Projects too small	- Agile (Scrum)	- Traditional - Agile in phases
Roles	- Sales - Project Manager - Designer - Developer - Tester	- Solution Manager - Project Manager - Developer - Tester	- Project Leader - Developer	- Product Owner - Scrum Master - Development Team	- PO / Coach - Dev. Team
Meetings	- Kick-off - Dailys - Weeklys	- As in scrum - No Dailys - On-demand	- Kick-off - On-demand	- As in ALP-Mobile - First planning before kick-off	- Weeklys
Distributed Working	- Too much overhead	- Nearshoring	- Freelancers	- Not easy	- Inevitable
Testing	- Manually	- Manually	- Manually	- Manually	- Manually

Table 6.1: Summarized information about the selected organizations in the case study

6.1.2 Mobile Application Development

The quotes concerning the differences of mobile application development to software development in general show that the interviewees see especially the increased need of flexibility in the development as a major distinction. This can be reasoned with the fact that the mobile market is very fast paced and therefore mobile software must constantly be developed further. The ongoing development regarding mobile technology brings also changes to the implementation of mobile apps, since the products must be updated to new environment conditions like changing screen sizes or new versions of the operating system.

6.1.3 Roles

The Android tech lead of Zeta stated that the roles in ALP-Mobile are very similar to the roles they have in their organization. Besides the fact that they use other names for the roles, the approach regarding the roles in the organization Gamma is also comparable to ALP-Mobile. According to the CEO of Eta they combine the roles of the product owner and the agile coach into one role. He reasoned this situation with the fact that by the time of the interview the organization was just newly founded and therefore the resources are limited. The CTO of Epsilon described that they have the role of the project leader which is comparable to the product owner prescribed in ALP-Mobile. Besides that role they have developers who implement the mobile applications. The situation in the organization Delta is a bit different. Due to the fact that they use nearshoring for the classic app development the roles implemented in the organization cannot be compared adequately to the roles prescribed in ALP-Mobile. Nevertheless they have the role of the solution manager which is mainly responsible for the activity of requirements engineering. In most projects they also have the role of the project manager which is responsible for the budget of the project. Besides those roles they have developers and testers.

As stated in the examination of process models in general, the usage of such models with all artifacts highly depends on the organizations and on the projects they realize. The investigation of the roles implemented in the organizations show exactly that situation. Limited resources respectively unnecessary overhead are reasons why organizations do not follow process models as they are prescribed.

6.1.4 Meetings

Similar to the roles in the organizations the established meetings also depend on the available resources. According to the CEO of Gamma their approach of the meetings is very similar to the approach in ALP-Mobile. They have a kick-off meeting, daily stand ups and a weekly meeting which combines estimation, review and retrospective. The interviewee of Zeta pointed out that the concept regarding the meetings prescribed in ALP-Mobile is almost the same as in the mobile department of Zeta. The only difference is that the first planning meeting is executed before the actual kick-off meeting of the project. The organization Delta tries to implement the meetings as advised in scrum. A difference to the meetings in scrum is the fact that they do not perform daily meetings. Instead those meetings are held on an on-demand basis. The on-demand approach is also the approach executed in the organization Epsilon. The CTO of Epsilon prefers that developers directly ask if they have problems instead of waiting for the next meeting to discuss this issue. The CEO of Eta stated that for them daily stand ups would be an unnecessary overhead since the size of the organization is not appropriate. They have a weekly meeting which is the only scheduled meeting at the moment.

The conclusion regarding the roles in the organizations can be mapped to the the meetings without further adjustments. It depends on the organizations and on the projects which meetings they establish respectively which meetings are reasonable.

6.1.5 Requirements Engineering

The Android tech lead of Zeta said that the requirements engineering already happens before the actual kick-off meeting of the project. The outcome of the first estimation is a set of user stories. According to the interviewed expert of Delta the requirements engineering is based on an early prototype of the application. Similar to ALP-Mobile the outcome of the requirements engineering is a set of user stories which describe the software. Concerning user stories the organization Gamma takes another approach. The CEO of Gamma stated that they do not write user stories in their projects, because he sees the creation as unreasonable since the developers already know the target group as well as the functionalities of the app. Therefore their focus is set directly on the features. The CTO of Epsilon pointed out that the knowledge of the customer regarding mobile applications defines this activity. The result of this activity is a requirements document. According to the CEO of Eta the task of requirements engineering is really important, because only with a profound requirements engineering the customer satisfaction at the end of the project can be ensured.

Concerning the activity of requirements engineering all experts were almost the same opinion. Each of the interviewees pointed out that this activity is an important part of their project lifecycle. Another similarity which was encountered was that this activity highly depends on the knowledge of the customer.

6.1.6 Development

Regarding the concept of no fixed iterations in the development the CEO of Eta pointed out that this concept is inevitable in mobile development. For him the iteration length proposed in scrum is not applicable in this field of software development, because usually the projects are small and therefore the flexibility is lost with long iterations. The Android tech lead of Zeta stated that due to the fact that they have weekly iterations in their process, which is an adaption to the usual sprint iterations advised in scrum, they are already more flexible in the development. A similar view regarding this subject was expressed by the CEO of Gamma. He described that the process in the organization Gamma is very similar to the process executed at Zeta. They also have weekly iterations, which highly increases the flexibility. In essence they have almost the same approach as proposed in ALP-Mobile since even the goals for the weekly iterations are not fixed, which means that work items can be added to the sprint backlog on-demand. The solution manager of Delta expressed a bit different opinion. He noted that if changes to sprint backlogs with iterations of one week are necessary then there are probably problems in the sprint planning respectively the requirements engineering. Because of the fact that the organization Epsilon does not make use of any particular process models, the author did not ask the CTO of Epsilon about his opinion regarding the concept of no fixed iterations in the development.

The experts were almost the same opinion regarding this issue. Iterations of two to four weeks as prescribed in scrum are too long in mobile application development, since the flexibility in the development is lost. Therefore processes must be adapted to the fast pace of the mobile market.

6.1.7 Distributed Working

The opinions of the experts were mixed concerning distributed working. The CEO of Eta thinks that this technique is essential in mobile application development, especially since modern technology supports this activity efficiently. Delta usually outsources the GUI development. The solution manager of Delta stated that distributed working is a good solution for the organization, especially if the partner offers high quality. The organization Epsilon works partly with freelancers. According to the CTO of Epsilon self-organization is a big requirement for these freelancers. The Android tech lead of Zeta explained that, especially in agile processes where daily contact is very important, this concept is not very easy to execute. The coordination must be very strict in order to be successful. The CEO of Gamma sees the overhead in the coordination as a main problem in distributed working. He pointed out that you have to invest a lot more in order to get same outcome.

In essence all experts were the opinion that distributed working is a possible alternative respectively opportunity, but the coordination and communication must be excellent. The social component must also not be underestimated. The interviewees pointed out that a distributed environment cannot be compared to a situation where all team members work together on site concerning social interactions.

6.1.8 Testing

Concerning the testing of mobile applications the organization Gamma has a dedicated testing team which is responsible for testing. Regarding automated testing the CEO of Gamma said that due to the fact that they implement a lot of customized solutions the overhead for the creation of automated tests would be unreasonable. The testing process executed at the organization Eta starts with unit tests on the bottom and user tests at the top of the application. Concerning the size of the project the CEO of Eta also mentioned that in small mobile projects the concepts of automated testing and continuous integration are probably an unnecessary overhead. The solution manager of Delta said that they have to distinguish between interface and frontend testing. The interface testing is closely bound to scrum and the frontend testing is performed with the help of test catalogs. At the moment the testing is performed manually. According to the CTO of Epsilon the testing in the organization Epsilon is mainly performed by developers who are assigned to test a specific application. Besides testing performed by the developers and the customer, the PO tests the application in course of the weekly meetings at Zeta. The Android tech lead of Zeta confirmed that concepts like automated testing and continuous integration are very interesting especially with applications which have a longer lifetime than small mobile apps.

The opinions of the experts perfectly displayed the current situation in mobile application testing. Concepts like automated testing and continuous integration are not very common. Nevertheless with the increase of complexity of mobile software those concepts will become more interesting in the future.

6.1.9 Distribution

The last topic in the cross-case analysis of the focused interviews concerns the distribution and the maintenance of the software. Regarding the distribution of the mobile application the Android tech lead of Zeta pointed out that Apple is much more complex in the distribution than Android. According to the CEO of Gamma the necessary review process at Apple is an unknown in the distribution of the software. He pointed out that it is important to communicate this fact with the customer in order to avoid any problems with the schedule of the project. The CEO of Eta also

mentioned that at Apple you have to plan one up to two weeks where you cannot do anything, because you have to wait for the deployment of the application. Therefore you need to be very careful in the development, since updates take very long. Regarding the review process at Apple prior to the distribution in the app store the CTO of Epsilon pointed out that it is a bit tedious, but in the end you just have to plan the project accordingly. Concerning new versions of the operating systems the solution manager of Delta noted that beta versions of the operating system offered by the operators are very helpful in order to release an update of the application soon after the release of the new operating system.

Based on the operating system, all experts where the same opinion. Apple is much more complex compared to Android when it comes to the distribution of the application. The necessary review process at Apple must also be considered in the schedule of a project. Nevertheless the impact is not that high, because the organizations are used to this situation. The experts also stated that it is important to communicate these dependences with the customer.

6.2 Examination of Research Propositions

In the previous chapter the author proposed six research propositions which are related to the initial research question of this thesis. As stated, these propositions should direct attention to something that should be examined within the scope of this study. The author defined the propositions on basis of the initial literature research. Following each of the propositions will be investigated in more detail:

- PR1: Organizations use agile processes because it is a good fit for small projects
  As a result of the focused interviews the author sees especially the increased need for flexibility in mobile application development as the main argument for the use of agile processes in this field of software development. As experienced in the case study sometimes the mobile projects are even too small for the use of such processes and therefore the implementation of a proper process model is seen as unreasonable. Nevertheless, as noted in the literature, smaller projects are in theory a good fit for agile processes, but the main reason why organizations use agile methodologies in mobile app development is the increased need for flexibility in the development.

- PR2: Mobile application development requires short iterations
  This proposition was confirmed by the majority of the interviewees. As pointed out in the data analysis of the case study, the experts noted that short iterations help to adapt to the wishes of the customer. This fact is also associated with the before mentioned need of increase in flexibility in mobile app development.

- PR3: Automated testing and continuous integration are not very common in mobile testing
  In course of the focused interviews the perception, that automation in mobile software testing is not very common, was corroborated. As the main reason for this situation the interviewees stated that the overhead for the creation of an automated test system is unreasonable in the most cases. This is reasoned with the fact that a lot of apps have a strong focus on the user interface which is hard to test with automated test cases. Additionally the experts pointed out that features are often very customized, which increases the effort of automated testing since the tests cannot be reused within other projects. Therefore mobile software is still mainly tested manually.

  Nevertheless some of the interviewees said that it would be great to have an automated testing process within their projects. Especially for changes late in the project lifecycle such

tests are very useful in order to check all functionalities of the software automatically. In course of the pilot interview the CTO of Beta, stated that they already invest time in the development of automated tests. Additionally he pointed out that a subset of these tests is executed with every commit to the repository which in fact is similar to the concept of continuous integration.

Based on the interviews conducted within the case study the author concludes that concepts concerning automated testing of mobile applications will become more important as the complexity of mobile software increases. For simple apps, which mainly focus on the visualization of information, the overhead of such concepts may be unreasonable, but, similar to software development in general, more complex applications depend on activities like automated testing and continuous integration in order to ensure the high quality of the software which is necessary to be successful in the highly competitive mobile market.

- PR4: Small team sizes, which are common in mobile app development, support distributed development
  As presented in the analysis of the data gathered through the interviews within the case study the experts had mixed opinions concerning the topic of distributed working within mobile app development. On the one hand there was the opinion that distributed working can bring great value to an organization since modern technology offers a huge variety of tools which help to communicate in a distributed environment. On the other hand there was the opinion that the lack of social interactions decreases the motivation of the employees and most importantly that the overhead of communication and coordination exceeds the benefits which come with this approach.

  The introduced activities in ALP-Mobile, which are dedicated to the possibility to use ALP-Mobile in a distributed environment, help teams to overcome the challenges in distributed development. It therefore depends on the organizations to decide whether a distributed environment is the right choice for the organization or not. The current state of the technology makes productive development in a distributed setting possible, but it is still not comparable to a collocated environment.

- PR5: Early prototyping is very important in mobile app development
  The interviewees stated that they highly make use of prototypes early in the development. The solution manager of Delta explained that a prototype of the application is the basis for every project proposal. That means that the prototyping of the app already happens before the actual kick-off in this organization. Other experts pointed out that they made great experiences with clickable mockups of the application. Such prototypes help to visualize the vision of an application and therefore feedback can be gathered early in the project. Especially the design and the functionality of the software can be addressed very soon.

  A disadvantage of this activity is the fact that clickable mockups already provide such a great user experience so that customers sometimes do not understand why it takes quite a long time to actually develop the software. Regarding this issue the solution manager (Delta) pointed out that they face exactly this problem.

  > "*We set strong focus on high-fi mockup prototype development. That means that the final prototypes are very close to the final product. The problem then is that many believe that the product is already finished after they have seen the prototype. We have to fight with that. For example if we present a nice-looking prototype and we still have another half a year until we can go live.*"
  >
  > Delta:#9

Besides this disadvantage the experts did not mention any problems with that concept. In contrary, the interviewees confirmed that early prototyping is very important in mobile app development.

- PR6: Mobile application development requires high collaboration with the customer
  The opinion of the interviewees regarding this topic was very clear. Customer collaboration is very important in mobile application development. The author sees the high focus on the user interface in mobile software as a main reason for the importance of customer collaboration. The experts also pointed out that especially in mobile app development the customer is often closely integrated in the development of the software which means that he can define the functionalities respectively the design of the app relatively long in the project lifecycle. This fact implies high communication with the customer, because only that way the high agility in the development can be ensured.

6.3 Comparison to Related Work

In section 1.4 the author presented related work of this thesis. In detail this section focused on different process models specific for mobile application development. In [23], Corral et al. concentrate on those models and investigate their impact in real environments. Mobile-D [18], one of those models, was introduced in chapter 4. Since the introduction of Mobile-D in 2004 the characteristics of mobile software have changed. Nevertheless according to Corral et al. this approach is the only model which has received considerable support in real world settings. Table 6.2 presents a summary of documented implementations of each methodology. The presented numbers show a need for more empirical evidence concerning those methodologies in order to answer the question whether they are applicable in practice or not.

Methodology	Year	Case Studies	Cited by
Mobile-D	2004	16	17
MASAM	2008	0	3
Hybrid	2008	0	9
Scrum	2010	1	4
SLeSS	2011	1	1

Table 6.2: Agile-based mobile software development processes and their implementations [23]

In the following the author compares each of the methodologies to ALP-Mobile. The first model in table 6.2 is Mobile-D. As already stated in this section, Mobile-D was first introduced nearly a decade ago. Mobile software has changed since then and therefore Mobile-D does not focus on those changed characteristics. Nevertheless this model has already been used in large industry settings [23] and therefore Mobile-D is probably the most prominent agile representative in mobile application development. In contrast to ALP-Mobile, Mobile-D does not make use of specific lean concepts, e.g. kanban.

The next methodology presented in the table is MASAM [19] which has a strong tie to Mobile-D. There is no case study with evidence of the execution of this model in a real environment. Similar to Mobile-D, MASAM also does not introduce any lean principles into the process.

Rahimian and Ramsin take a different attempt compared to ALP-Mobile by combining plan-based and agile methodologies into a framework for the development of mobile applications. Again this hybrid approach [20] does not make use of any lean concepts and there is also no case study evidence of the implementation of this approach in a real world setting.

In [21] Scharff and Verma make use of scrum for the development of mobile applications in a scholar setting. Since ALP-Mobile makes use of concepts of scrum, this approach shows similarities to the concepts introduced into ALP-Mobile. Nevertheless it also lacks the input of lean principles.

SLeSS [22] is the last process model in table 6.2 and it is also a combination of agile and plan-based methodologies. In case of SLeSS it is a combination of scrum and lean six sigma. Besides the case study, which involves a real world project, this model is cited just once according to [23].

The above comparison shows that ALP-Mobile takes a unique approach in combining agile and lean methodologies in order to create a new process model for the development of mobile applications.

6.4 Answering of the Research Question

After concentrating on the cross-case analysis, the defined research propositions and the related work the author focuses in this section on the answering of the initial research question:

How can agile and lean processes be adapted to mobile app software development?

In order to answer this question the author proposed an agile and lean process model for mobile app development called ALP-Mobile. This model is on the one hand based on an extensive literature research in relevant areas and on the other hand on practical experiences gathered in course of the pilot interview conducted in the case study. Additionally to this pilot interview the author conducted focused interviews with experts in the field of mobile application development. By presenting ALP-Mobile to the interviewees the author was able to derive strengths and weaknesses of this process model. As a part of the interviews the interviewer asked the interviewees about their opinion regarding ALP-Mobile. In the following the author presents quotes concerning ALP-Mobile.

The CEO of Eta thinks that the concept of ALP-Mobile is really good. Especially the kanban board can bring benefit since it is easily scalable. Nevertheless he pointed out that because of the current size of the organization the idea of an extra coach is not reasonable at the moment. Currently they have the coach and the product owner in one person.

> "*I think the model is quite good. We probably will not afford a coach besides the product owner for a long time, because it makes no sense. But I think especially the board is a pretty good thing, because it is very flexible. It probably will grow easily. There is no difference if you have limits of one and two on the board, because you only have two developers and for example only five stories in the backlog or if there are limits of seven or eight and you have 25 stories in the backlog. The concept remains the same.*"
>
> Eta:#8

The solution manager of Delta also had a positive opinion of ALP-Mobile. He stated that he really likes the approach of the dailys in which the facilitator walks the board and focuses on the actual user stories instead of focusing on the people as advised in scrum.

> "*It gives the impression that you were trying to combine scrum and kanban - I think it's good. Concerning the role of the coach I do not see the big difference to the scrum*

> *master. What I really liked is your approach of the dailys. That you concentrate on the perspective of the stories and not on the perspective of the people who implement it.*"
>
> Delta:#10

As already discussed in section 5.5, the size of the organization respectively of the projects is very important concerning the decision which process model to use in the project. The CTO of Epsilon pointed out that the organization Epsilon mainly develops marketing apps which have a strong focus on data visualization. Therefore he sees the execution of a proper process model as an unnecessary overhead for their projects, but he states further that for more complex projects ALP-Mobile would be a good fit.

> "*The purpose of our apps is to draw attention to the customer. I would rather see your process in another type of app such as an app for employees of insurance companies who have their product portfolio displayed within the app and where you have cost calculators to various insurance products. An app where actually calculations in the background happen which normally would be done on a laptop or desktop. With the app, I then have the advantage that I can use this app directly with my customers. These are more complex things, where such a process model I think would be optimal.*"
>
> Epsilon:#10

The Android tech lead of Zeta sees already a lot of similarities in ALP-Mobile to the process model executed within their organization. The only point in which he did not agree with the concept of ALP-Mobile is the fact that the first estimation of the user stories happens after the initial kick-off meeting. He explained that within their process they need the first estimations for the initial proposal of the project. Besides that he did not mention any problems he would expect when using ALP-Mobile in practice. Based on the statement that ALP-Mobile is very similar to the processes in their organization the interviewed expert (Zeta) responded the following:

> "*We use a lot of scrum. The kanban approach we follow implicitly. For example, the Selected column in the kanban board: in our process the PO is also present at the meetings and then he has a say in what is important. Concerning the roles we are also much the same. There's nothing unfamiliar. Parts were really very similar and therefore good. The only point that would not work for us is that the story points are estimated after the kick-off. For us the kick-off is only after the project has been confirmed and for that we already need the estimation of the effort for the proposal. But this is probably the only point that does not work in the real world - at least for us. It's different when the project is internally and simply uses its own manpower. Since I do not need to estimate in advance, that works better.*"
>
> Zeta:#10

The CEO of Gamma also liked the concept of ALP-Mobile in general. A good point for him is that there are no fixed iterations, which increases the flexibility in the development of the software. He pointed out that, although they have weekly iterations, their process model is very similar compared to ALP-Mobile regarding this issue.

To conclude this section the author presents ALP-Mobile as the answer to the initial research question on how agile and lean processes can be adapted to mobile app software development. ALP-Mobile combines concepts from the agile respectively lean world and furthermore focuses

on the fast pace of the mobile market. The focused interviews in course of the case study provided positive feedback, which can be seen as a first indicator of the fact that ALP-Mobile is capable of bringing real benefits to organizations working in the field of mobile application development. Nevertheless only practical experiences with ALP-Mobile will show if this process model can be executed successfully in the real world.

6.4.1 Limitations

The case study conducted in course of this master thesis provided valuable insights into real projects in the field of mobile application development. A valid criticism regarding the case study is the fact that only focused interviews were used as source of evidence. As noted in chapter 5, it is advised to use multiple sources of evidence in case study research in order to corroborate the findings. Due to the limited resources of the author within this research this approach was not possible. Another factor which was influenced by the limited resources of this master thesis is the fact that the boundaries for the selection of the cases in the case study were not very strict. Therefore not only organizations using agile and lean principles were selected. A positive aspect of this situation is that it was possible to investigate different perspectives regarding mobile application development.

Besides those limitations the gathered information throughout the focused interviews helped the author to derive strengths and weaknesses of ALP-Mobile which are presented in the next section of this thesis.

6.4.2 Strengths and Weaknesses of ALP-Mobile

Based on the displayed quotes in this section the first impression of the experts regarding ALP-Mobile was very positive. The interviewees in particular liked the increased flexibility which is a result of the use of kanban as a tool for the process execution in which fixed iterations in the development are waived in order to create a steady flow of work. The kanban board as the main source of information was another positive concept introduced in ALP-Mobile. Especially the ability to use the board for small as well as for large projects makes it very interesting for a variety of teams. Due to the fact that ALP-Mobile is a combination of popular process models the learning curve should not be too high, which helps to decrease the overhead of the initial implementation of ALP-Mobile.

Concerning the weaknesses of ALP-Mobile, the experts did not mention any big disadvantages of the model. Nevertheless they pointed out that some concepts would not work at their organization, e.g. the story point estimation after the initial kick-off meeting as stated by the Android tech lead of Zeta. A reason for that is the fact that ALP-Mobile is mainly based on an extensive literature research which was conducted in course of the theoretical part of this thesis and therefore limitations in the real world are possible. Another reason for this situation is that process models always need to be adapted to the settings of the organizations which implement a particular model. This is also true for the execution of ALP-Mobile. In order to really benefit from ALP-Mobile organizations must adapt this process model according to their preferences.

7 Conclusion

The highly competitive mobile market requires standardized processes in order to be successful in this fast paced environment. Therefore the author proposed a new process model called ALP-Mobile which is a combination of agile and lean methodologies to overcome the challenges in mobile application development.

ALP-Mobile features three main phases which are referred to as the 3D's of ALP-Mobile which are 1) Definition, 2) Development and 3) Distribution. Those phases cover a complete project lifecycle in mobile application development. The first phase focuses on the definition of the final product and on the analysis of the upcoming software project. The second phase of the model is dedicated to the implementation and to the testing of the application. The goal of the last phase is to bring the app to its desired market and to provide proper maintenance of the software in order to ensure the high quality of the software.

In order to overcome the challenges in mobile application development the author combined concepts from scrum, XP and kanban. To increase the flexibility in the development kanban was used as a tool for the process execution. Compared to the more prescriptive scrum, kanban only has three main constituents which are listed below:

- Visualize the workflow
- Limit Work In Progress (WIP)
- Measure the lead time

The kanban board introduced as the main source of information in ALP-Mobile is the perfect tool to support the three main constituents of kanban. It visualizes the workflow, the WIP limit is visible and directly attached to each stage in the process and additionally the lead time can easily be measured by the time one item needs for the whole workflow until the last stage in the process.

Besides the presented phases ALP-Mobile prescribes three roles with different responsibilities. The development team introduced in ALP-Mobile is responsible for the development of the application. It should be interdisciplinary in order to avoid dependences to other departments or organizations. The second role in the process model is the product owner who is responsible for the close collaboration with both the development team and the customer in order to make sure that the visions of the customer regarding the product are met. As the third role in ALP-Mobile the author proposed the agile coach who is a coach for XP, scrum and kanban. He therefore helps everyone involved in the process to understand the values, principles and practices applied in ALP-Mobile.

ALP-Mobile furthermore prescribes four different types of meetings. First, there is the kick-off meeting in which all necessary information about the upcoming project is distributed within the project team. Second, the daily stand ups directly focus on the flow of work in ALP-Mobile. Therefore bottlenecks in the process can easily be discovered. In contrast to daily scrums the stand ups performed in ALP-Mobile are moderated by a facilitator who walks the board. This means that he uses the kanban board and focuses directly on the user stories rather than on the individuals who implemented the stories. The third meeting is dedicated to the effort estimation of the user stories. Those planning meetings take place in the beginning of the project and every

time new features need to be estimated. The last prescribed meetings are the retrospectives which are split into two different parts. In the review part the product is inspected and in the retrospective part the process is reviewed. The iteration for this meeting is advised to be every week.

In course of the case study, which was conducted within this research, ALP-Mobile was introduced to experts in the field of mobile application development. The overall feedback regarding the presented concepts was very positive. Especially the increased flexibility in the development was seen as a necessary concept in the field of mobile application development. Nevertheless the focused interviews showed that process models in general and also ALP-Mobile in particular need to be adapted to the preferences and to the settings of the respective organization which implements the specific methodologies of a model.

In order to give an outlook into the future regarding mobile application development the author uses on the one hand the information collected through the theoretical literature research and on the other hand the practical experiences gathered in course of the case study. By looking on the predictions concerning the market share of operating systems in the mobile phone market, there will be three main operating systems which are Android, iOS and Windows Phone in the future. This forecast reduces the uncertainty in the market and furthermore helps organizations in the decision for which operating system an app should be developed. Additionally with the further evolution of web 2.0 technologies like HTML5, CSS3 and JavaScript web apps will become even more interesting in the future.

Regarding future research the author sees especially in the area of mobile application testing a lot of work to do. With the predictions of having three main operating systems in the future there is a need for tools which support developers in testing their applications on various platforms. Concepts like automated testing and continuous integration have been very useful respectively essential in software development in general. Those concepts help to ensure the high quality of the software which is needed to survive in the very competitive markets. Especially with the increased complexity of apps these concepts will become necessary in the development of mobile software. Additionally to the ability to test multiple operating systems with a single tool, such tools must also be able to adapt to the specific characteristics of mobile devices, e.g. different screen sizes, sensors and other hardware specifications.

In summary it can be stated that based on the feedback collected within the case study the concepts introduced in ALP-Mobile can bring great benefits to organizations working in the field of mobile application development. In order to corroborate the findings and the results of this master thesis ALP-Mobile must be the subject of further research in which empirical evidence of the practical usage of this process model is collected and analyzed.

Bibliography

References

[1] Yu Gao and XiaoQiu Yao. “The Two Approaches to Sustainable Development of the Theory of Software Process Models”. In: *3rd International Conference on Information Management, Innovation Management and Industrial Engineering*. 2010.

[2] Alok Mishra and Deepti Mishra. “A Curriculum for Agile Software Development Methodologies”. In: *SIGSOFT Software Engineering Notes, Volume 36, Issue 3*. 2011.

[3] Xiaofeng Wang. “The Combination of Agile and Lean in Software Development: An Experience Report Analysis”. In: *Agile Conference*. 2011.

[4] Dean Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison Wesley, 2010.

[5] Mary Poppendieck and Michael A. Cusumano. “Lean Software Development”. In: *IEEE, Software, Volume 29, Issue 5*. 2012.

[7] Crescencio Rodrigues Lima Neto and Eduardo Santana de Almeida. “Five Years of Lessons Learned from the Software Engineering Course: Adapting Best Practices for Distributed Software Development”. In: *Collaborative Teaching of Globally Distributed Software Development Workshop (CTGDSD)*. 2012.

[8] Pekka Abrahamsson. “Mobile software development – the business opportunity of today”. In: *Proceedings of the International Conference on Software Development*. 2005.

[9] Eva del Nuevo, Piattini Mario, and Francisco J. Pino. “Scrum-based Methodology for Distributed Software Development”. In: *6th IEEE International Conference on Global Software Engineering*. 2011.

[15] Ashley Aitken and Vishnu Ilango. “A Comparative Analysis of Traditional Software Engineering and Agile Software Development”. In: *46th Hawaii International Conference on System Sciences*. 2013.

[16] Pavel Smutný. “Mobile development tools and cross-platform solutions”. In: *13th International Carpathian Control Conference*. 2012.

[17] Elizabeth Woodward, Steffan Surdek, and Matthew Ganis. *A Practical Guide to Distributed Scrum*. Ibm Press, 2010.

[18] Pekka Abrahamsson et al. “Mobile-D: an agile approach for mobile application development”. In: *OOPSLA, Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. 2004.

[19] Yang-Jae Jeong, Ji-Hyeon Lee, and Gyu-Sang Shin. “Development Process of Mobile Application SW Based on Agile Methodology”. In: *10th International Conference on Advanced Communication Technology*. 2008.

[20] Vahid Rahimian and Raman Ramsin. “Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach”. In: *Second International Conference on Research Challenges in Information Science*. 2008.

[21] Christelle Scharff and Ravi Verma. “Scrum to Support Mobile Application Development Projects in a Just-in-time Learning Context”. In: *CHASE, Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*. 2010.

[22] Thiago Ferraz V. da Cunha, Valéria L. L. Dantas, and Rossana M. C. Andrade. “A Scrum and Lean Six Sigma integration approach for the development of sofware customization for mobile phones”. In: *25th Brazilian Symposium on Software Engineering (SBES)*. 2011.

[23] Luis Corral, Alberto Sillitti, and Giancarlo Succi. “Agile Software Development Processes for Mobile Systems: Accomplishment, Evidence and Evolution”. In: *10th International Conference on Mobile Web Information Systems (MobiWIS)*. 2013.

[24] William Chaves de Souza Carvalho et al. “A comparative Analysis of the Agile and Traditional Software Development Processes Productivity”. In: *30th International Conference of the Chilean Computer Science Society*. 2011.

[25] Roger S. Pressman. *Software Engineering: A Practitioner’s Approach, Seventh Edition*. McGraw-Hill, 2010.

[26] Sebastián Tyrrell. “The Many Dimensions of the Software Process”. In: *ACM, Crossroads, Volume 6, Issue 4*. 2000.

[27] Niklaus Wirth. “A Brief History of Software Engineering”. In: *IEEE, Annals of the History of Computing, Volume 30, Issue 3*. 2008.

[28] Edsger W. Dijkstra. “The Humble Programmer”. In: *ACM, Communications of the ACM, Volume 15, Issue 10*. 1972.

[29] Li Jiang and Armin Eberlein. “An Analysis of the History of Classical Software Development and Agile Development”. In: *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*. 2009.

[30] Yu Beng Leau et al. “Software Development Life Cycle AGILE vs Traditional Approaches”. In: *2012 International Conference on Information and Network Technology (ICINT 2012)*. 2012.

[31] Winston Royce. “Managing the development of large software systems”. In: *IEEE, Proceedings of Westcon*. 1970.

[32] Richard E. Fairley. *Managing and Leading Software Projects*. Wiley-IEEE Press, 2009.

[33] Barry Boehm and Richard Turner. “Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods”. In: *Proceedings of the 26th International Conference on Software Engineering (ICSE’04)*. 2004.

[34] Tom DeMarco. “Software Engineering: An Idea Whose Time Has Come and Gone?” In: *IEEE, Software, Volume 26, Issue 4*. 2009.

[35] Tom DeMarco. *Controlling Software Projects: Management, Measurement & Estimation*. Prentice Hall/Yourdon Press, 1982.

[36] Barry Boehm. “Get Ready for Agile Methods, with Care”. In: *IEEE, Computer, Volume 35, Issue 1*. 2002.

[37] G.I.U.S. Perera and M.S.D. Fernando. “Enhanced Agile Software Development – Hybrid Paradigm with LEAN Practice”. In: *Second International Conference on Industrial and Information Systems, ICIIS 2007*. 2007.

[38] Curt Hibbs, Steve Jewett, and Mike Sullivan. *The Art of Lean Software Development*. O’Reilly Media, 2009.

[39] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional, 2003.

[40] Thomas Stober and Uwe Hansmann. *Agile Software Development: Best Practices for Large Software Development Projects*. Springer, 2009.

[41] Orit Hazzan and Yael Dubinsky. *Agile Software Engineering*. Springer, 2009.

[42] Hirotaka Takeuchi and Ikujiro Nonaka. "The New New Product Development Game". In: *Harvard Business Review*. 1986.

[43] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2001.

[44] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Longman, 2012.

[45] Marko Ikonen et al. "On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation". In: *16th IEEE International Conference on Engineering of Complex Computer Systems*. 2011.

[46] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum - making the most of both*. lulu.com, 2010.

[47] Thomas Epping. *Kanban für die Softwareentwicklung*. Springer, 2011.

[49] Raghvinder Sangwan, Neel Mullick, and Daniel J. Paulish. *Global Software Development Handbook*. Auerbach Publishers Inc., 2006.

[50] Thomas W. Malone and Kevin Crowston. "What is coordination theory and how can it help design cooperative work systems?" In: *CSCW '90 Proceedings of the 1990 ACM conference on Computer-supported cooperative work*. 1990.

[51] Thomas J. Duffy. *Programming with Mobile Applications: Android(TM), iOS, and Windows Phone 7*. Cengage Learning, 2012.

[52] Marcelo Nogueira Cortimiglia, Alejandro Germán Frank, and Liziane Seben. "Tablets: The Next Disruptive Computing Technology?" In: *IEEE, IT Professional, Volume 15, Issue 3*. 2013.

[59] Anar Gasimov et al. "Visiting Mobile Application Development: What, How and Where". In: *2010 Ninth International Conference on Mobile Business / 2010 Ninth Global Mobility Roundtable*. 2010.

[60] Ann Nosseir et al. "Mobile Development Process Spiral". In: *Seventh International Conference on Computer Engineering & Systems (ICCES)*. 2012.

[61] Manuel Palmieri, Inderjeet Singh, and Antonio Cicchetti. "Comparison of Cross-Platform Mobile Development Tools". In: *16th International Conference on Intelligence in Next Generation Networks (ICIN)*. 2012.

[63] Jeff McWherter and Scott Gowell. *Professional Mobile Application Development*. John Wiley & Sons, Inc., 2012.

[64] Ben Morris et al. *An Introduction to bada: A Developer's Guide*. Wiley, 2010.

[65] "Native, web or hybrid mobile-app development". In: *IBM Software: Thought Leadership White Paper*. 2012.

[67] David Sin, Erin Lawson, and Krishnan Kannoorpatti. "Mobile web apps – the non-programmer's alternative to native applications". In: *5th International Conference on Human System Interactions*. 2012.

[69] S Mohorovičić. "Implementing Responsive Web Design for Enhanced Web Presence". In: *36th International Convention on Information & Communication Technology Electronics & Microelectronics (MIPRO)*. 2013.

[70] Djoni Haryadi Setiabudi and Lady Joanne Tjahyana. “Mobile Learning Application Based On Hybrid Mobile Application Technology Running On Android Smartphone and Blackberry”. In: *International Conference on ICT for Smart Society (ICISS)*. 2013.

[72] Ridi Ferdiana. “Agile Software Engineering Framework for Evaluating Mobile Application Development”. In: *International Journal of Scientific & Engineering Research, Volume 3, Issue 12*. 2012.

[73] Jay Trimble and Christopher Webster. “From Traditional, to Lean, to Agile Development: Finding the Optimal Software Engineering Cycle”. In: *46th Hawaii International Conference on System Sciences*. 2013.

[74] Pekka Abrahamsson. “Agile Software Development of Mobile Information Systems”. In: *19th International Conference on Advanced Information Systems Engineering*. 2007.

[75] Anthony I. Wasserman. “Software Engineering Issues for Mobile Application Development”. In: *FoSER, Proceedings of the FSE/SDP workshop on Future of software engineering research*. 2010.

[76] Shane Conder and Lauren Darcey. *Android Wireless Application Development, Second Edition*. Addison-Wesley Longman, 2010.

[77] Ichiro Satoh. “A testing framework for mobile computing software”. In: *IEEE, Transactions on Software Engineering, Volume 29, Issue 12*. 2003.

[78] Sean Stolberg. “Enabling Agile Testing through Continuous Integration”. In: *Agile Conference*. 2009.

[79] Mona Erfani Joorabchi, Ali Mesbah, and Philippe Kruchten. “Real Challenges in Mobile App Development”. In: *ACM / IEEE, International Symposium on Empirical Software Engineering and Measurement, 2013*. 2013.

[80] Lyssa Adkins. *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*. Addison Wesley, 2010.

[81] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000.

[82] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[83] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall International, 2005.

[84] Nils C. Haugen. “An empirical study of using planning poker for user story estimation”. In: *Agile Conference*. 2006.

[85] Esther Derby and Diana Larsen. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Programmers, 2006.

[86] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison-Wesley Longman, 2004.

[87] Haeng Kon Kim. “Test Driven Mobile Applications Development”. In: *Proceedings of the World Congress on Engineering and Computer Science*. 2013.

[88] B. Kirubakaran and V. Karthikeyani. “Mobile Application Testing – Challenges and Solution Approach through Automation”. In: *Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME)*. 2013.

[91] J.M. Verner et al. “Guidelines for Industrially-Based Multiple Case Studies in Software Engineering”. In: *Third International Conference on Research Challenges in Information Science*. 2009.

[92] Mary Sue MacNealy. “Toward Better Case Study Research”. In: *IEEE, Transactions on Professional Communication, Volume 40, Issue 3*. 1997.

[93] Dawson R. Hancock and Bob Algozzine. *Doing Case Study Research: A Practical Guide for Beginning Researchers*. Teachers College Press, 2006.

[94] Robert K. Yin. *Case Study Research: Design and Methods (Furth Edition)*. SAGE Publications, 2009.

Online References

All URLs were last checked for their availability on 04/14/2014.

[6] *Agile Manifesto*. 2001. URL: http://agilemanifesto.org/.

[10] Sara Dover. *Study: Number of smartphone users tops 1 billion*. 2012. URL: http://www.cbsnews.com/8301-205_162-57534583/study-number-of-smartphone-users-tops-1-billion/.

[11] James Johnson. *52.5 Million Tablets Sold In Q4 2012, Record Sales Included 22.9 Million iPads*. 2013. URL: http://www.inquisitr.com/503134/52-5-million-tablets-sold-in-q4-2012-record-sales-included-22-9-million-ipads/.

[12] *Top iOS and Android apps largely absent on Windows Phone and BlackBerry 10*. 2013. URL: http://www.canalys.com/newsroom/top-ios-and-android-apps-largely-absent-windows-phone-and-blackberry-10.

[13] *Worldwide Mobile Phone Market Forecast to Grow 7.3% in 2013 Driven by 1 Billion Smartphone Shipments*. 2013. URL: http://www.idc.com/getdoc.jsp?containerId=prUS24302813.

[14] *The Mobile App Market will be Worth $27 Billion in 2013 as Tablet Revenue Grows*. 2013. URL: https://www.abiresearch.com/press/the-mobile-app-market-will-be-worth-27-billion-in-.

[48] *Jack Nilles*. URL: http://www.jala.com/jnmbio.php.

[53] *More Smartphones Were Shipped in Q1 2013 Than Feature Phones*. 2013. URL: http://www.idc.com/getdoc.jsp?containerId=prUS24085413.

[54] *Over 1 billion Android-based smart phones to ship in 2017*. 2013. URL: http://www.canalys.com/newsroom/over-1-billion-android-based-smart-phones-ship-2017.

[55] *Gartner Says Worldwide PC, Tablet and Mobile Phone Shipments to Grow 4.5 Percent in 2013 as Lower-Priced Devices Drive Growth*. 2013. URL: http://www.gartner.com/newsroom/id/2610015.

[56] *IDC Forecasts Worldwide Tablet Shipments to Surpass Portable PC Shipments in 2013*. 2013. URL: http://www.idc.com/getdoc.jsp?containerId=prUS24129713.

[57] *Gartner Says Mobile App Stores Will See Annual Downloads Reach 102 Billion in 2013*. 2013. URL: http://www.gartner.com/newsroom/id/2592315.

[58] *Most popular Android market categories*. 2013. URL: http://www.appbrain.com/stats/android-market-app-categories.

[62] *Gartner Says Worldwide PC Shipments in the Second Quarter of 2013 Declined 10.9 Percent*. 2013. URL: http://www.gartner.com/newsroom/id/2544115.

[66] *Android and iOS Combine for 92.3% of All Smartphone Operating System Shipments in the First Quarter While Windows Phone Leapfrogs BlackBerry*. 2013. URL: http://www.idc.com/getdoc.jsp?containerId=prUS24108913.

[68] *Bootstrap*. URL: http://getbootstrap.com/.

[71] *Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid*. 2013. URL: http://www.gartner.com/newsroom/id/2324917.

[89] *Robotium*. URL: http://code.google.com/p/robotium/.

[90] *Jenkins*. URL: http://jenkins-ci.org/.

A Appendix

A.1 Interview Guideline

1. Vielen Dank, dass Sie sich Zeit für dieses Interview genommen haben. Können Sie sich kurz vorstellen und Ihr Aufgabengebiet in Ihrer Firma erläutern?
2. Je nach Aufgabengebiet der zu interviewenden Person:
   - Mit wie vielen Personen arbeiten Sie in Ihrem Team zusammen?
   - Wie viele Mitarbeiter sind in Ihrem Team tätig?
   - Wie viele Personen arbeiten in Ihrer Firma?
3. Welche Dienstleistungen bietet Ihre Firma an?
4. Im Rahmen meiner Diplomarbeit beschäftige ich mit agilen und lean Prozessmodellen. Können Sie beschreiben, inwiefern Prozessmodelle in Ihrer Firma zum Einsatz kommen?
   - Welches Bild haben Sie im Allgemeinen von agilen und lean Vorgehensmodellen?
   - Werden im Speziellen agile oder lean Modelle in Ihrer Firma eingesetzt?
5. Warum verwenden Sie diese Art von Prozessen?
   - Können Sie bitte speziell bezogen auf Ihrer Rolle im Unternehmen den Mehrwert beschreiben, den Sie durch den Einsatz von diesen Prozessen erhalten?
6. In der Literatur gibt es unzählige Beispiele für Prozessmodelle in der Softwareentwicklung. Inwiefern denken Sie, dass diese Modelle im Bereich der Entwicklung von mobilen Applikationen eingesetzt werden können?
   - Bezogen auf die Entwicklung von mobilen Applikationen, worin genau sehen Sie die Unterschiede zur allgemeinen Softwareentwicklung beziehungsweise was sind für Sie die Besonderheiten bei der Entwicklung von mobilen Applikationen?
7. *Präsentation von ALP-Mobile*
   - *Rollen*
   - *Meetings*
   - *Phasen*
   - *Workflow*
8. ALP-Mobile sieht 3 Rollen vor: den Product Owner, das Development Team und einen agilen Coach. Welche Rollen verwenden Sie in Ihren Projekten?
9. Die vorgestellten Meetings sind ein wichter Bestandteil von ALP-Mobile - welche Meetings werden in Ihrer Firma durchgeführt?

10. ALP-Mobile basiert grundlegend auf drei Phasen, welche den kompletten Projektzyklus abdecken. Die erste Phase ist die sogenannte Definitionsphase. Requirements Engineering ist eine sehr wichtige Aktivität in dieser Phase. Wie läuft bei Ihnen die Anforderungsanalyse ab?
    - Wie gehen Sie mit sich ändernden Requirements um?
11. Wie schätzen Sie den Aufwand von Ihren Projekten?
    - Haben Sie mit Story Points gearbeitet?
    - Wenn ja, wie benutzen Sie diese?
12. Ein wichtiger Punkt von ALP-Mobile ist der Kontakt mit dem Kunden. In wieweit wird der Kunde in den laufenden Entwicklungsprozess integriert?
13. Wie stehen Sie zu dem Einsatz von UI - Prototypen früh in der Entwicklung?
14. Welche weiteren Schritte sind für Sie in der Designphase des Projekts wichtig?
15. Das Kanban Board stellt den zentralen Informationsbeschaffungspunkt in ALP-Mobile dar. Besonders in der Developmentphase ist dieses Konzept sehr wichtig. Wie ist Ihr Vorgehen in Bezug auf die Entwicklung der Software?
    - ALP-Mobile verzichtet auf den Einsatz von fixen Iteration wie in Scrum vorgeschrieben. Dadurch wird mehr Flexibilität im Prozess angestrebt. Was halten Sie von diesem Konzept?
16. Im Rahmen meiner Arbeit habe ich mich unter anderem mit den verschiedenen Möglichkeiten der App - Entwicklung beschäftigt. Ich habe hierbei drei Arten beschrieben: 1) Native Apps, 2) Hybrid Apps, 3) Web - Apps. Welcher Typ wird in Ihrem Team / Ihrer Firma präferiert?
    - Bezogen auf diese drei Arten: Sehen Sie hierbei konkrete Unterschiede im Entwicklungsprozess?
17. Global software development beziehungsweise verteiltes Arbeiten stellt in der heutigen Zeit eine Möglichkeit dar, den Entwicklungsprozess auszulagern. Haben Sie Erfahrungen auf diesem Gebiet?
    - Bezogen auf den Prozess, worin sehen Sie die Schwierigkeiten in einem verteilten Umfeld?
    - Ergeben sich Ihrer Meinung nach spezielle Probleme bei mobilen Projekten?
18. Tests sind ein weiteres wichtiges Konzept von ALP-Mobile. Können Sie ihre Erfahrungen von Testing im Bereich von mobilen Applikationen wiedergeben?
    - Automated Testing und Continous Integration sind Kernpunkte der Lean - Softwareentwicklung. Was ist Ihr Standpunkt in Bezug auf diese zwei Techniken vor allem im Bereich der mobilen Applikationen.
    - Worin sehen Sie die grundsätzlichen Unterschiede zum Testing von konventionellen Softwareprojekten?
    - In welchen Phasen des Projekts testen Sie für gewöhnlich Ihre Software?
19. Die letzte Phase von ALP-Mobile, die Distributionsphase, beschäftigt sich mit dem Deployment und der Wartung der Software. Die Distribution der Software hängt sehr stark von der jeweiligen Entwicklungsart der App sowie der Wahl des Betriebssystems ab. Für welche Betriebssysteme entwickeln Sie Ihre Applikationen?

- Kommt es hierbei zu Veränderungen im Entwicklungsprozess, wenn für unterschiedliche Betriebssysteme entwickelt wird?
- Wenn ja, von welchen Unterschieden beziehungsweise Veränderungen sprechen wir hier?
- Was sind die Vor- und Nachteile der Distribution von Apps via Stores?

20. Als nächstes würde ich Sie gerne zu Ihrer Meinung zum Thema Wartung von Apps befragen. Empfinden Sie dieses Thema als einen wichtigen Punkt in Bezug auf Projekte im mobilen Umfeld?

    - Bezogen auf den Prozess an sich, inwiefern verändert sich dieser nach der Fertigstellung der Software?

21. Welchen Eindruck haben Sie von ALP-Mobile nach diesem kurzen Einblick?

    - Welche Punkte finden Sie in Ihrem aktuellen Prozess wieder?
    - Welche Punkte können Sie sich vorstellen umzusetzen?
    - Bei welchen Punkten sehen Sie Probleme?

A.2 Interview Quotes

A.2.1 Alpha

#1 „Bei uns ist das bei Kundenprojekten anders als bei eigenen Produkten. Bei Kundenprojekten ist die Tatsache, dass es ein vorgegebenes Budget gibt und eine Anforderungsanalyse dem Auftrag zu Grunde liegt ist der Entwicklungsprozess ein ganz ein Anderer als bei eigenen Produkten. Die Tatsache ist eben total spannend, weil bei eigenen Produkten ein agiler Softwareprozess viel leichter und einfacher zu implementieren ist als bei Kundenauftragsprojekten. Bei Kundenauftragsprojekten ist die Iteration maßgeblich vom Kunden und dem Okay vom Kunden abhängig. (...) Bei Fixpreisprojekten kann ich nicht anfangen ein Produkt zu verbessern, bevor ich nicht das Okay vom Kunden habe."

#2 „Wenn ich jetzt eine Retailapp nehme, die im App Store ist und dort verkauft wird. Nach dem ersten Release bekomme ich sehr viele Supportanfragen, dass Features fehlen oder dass irgendwo Fehler auftreten. Dann ist das natürlich wie in klassischen agilen Softwareprojekten, dass ich das Ganze in Sprints beziehungsweise Iterationen abarbeite. Dann macht das total Sinn."

#3 „Wir verwenden 2 Arten - low-fidelity and high-fidelity. Low-fidelity sind klassische Balsamiq Mock-Up Prototypen am Anfang und high-fidelity ist das, was uns der Designer gibt und zwar click-through Prototypen. (...) Immer wenn wir Projekte machen, dann bekommen wir immer total coole click-through Prototypen, die fast ausschauen wie eine App. Was es beim Kunden schwierig macht, weil der dann denkt, dass er die App sozusagen schon in der Hand hat. Man muss dann aber noch die Funktionalitäten implementieren. Das macht es ein bissi schwieriger. Die zwei Arten haben jetzt wir als Beispiel."

#4 „Wir haben zwei Entwickler, die in Tirol sitzen. Das funktioniert reibungslos aus softwareentwicklungstechnischer Sicht. Aus Betriebs- oder aus Organisationssicht schaut das anders aus. Weil wir haben hier drei Android Entwickler und die freuen sich jedenfalls sehr, wenn der Martin, der Android Entwickler aus Tirol, kommt, weil dann könnens ihn endlich direkt fragen. (...) Rein auf die Softwareentwicklung bezogen funktioniert das aktuell sehr gut."

#5 „Wir testen eigentlich nicht sehr viel strukturiert. Jeder im Unternehmen hat ein Device zum Testen und versucht so viele Fehler wie möglich zu finden. Diese Fehler werden dann gefixed. Das ist unser aktueller Ansatz, der auch einfach aus Budgetgründen so ist."

A.2.2 Beta

#1 „Für uns sind Prozessmodelle sehr wichtig. Vielleicht weniger Prozessmodelle, aber zumindest die wohl definierten Prozesse. Das ganze Thema, wie die Entwicklung abläuft, ist für uns sehr wichtig, da Beta als Unternehmen ISO zertifiziert ist. Wir haben ein Qualitätsmanagement nach ISO 13485, was wir brauchen um Medizinprodukte anbieten zu dürfen. (...) Aus dieser Perspektive heraus ist das ganze Thema Softwareentwicklungsprozesse sehr wichtig für uns."

#2 „Macht absolut Sinn. Jedoch habe ich die Erfahrung allerdings schon gemacht, dass man sich grade im Bereich wo das Ziel noch nicht ganz klar ist, also wo ich nicht ein Produkt mache, welches einfach nur ein anderes Produkt kopiert oder einen bestimmten Pfad gehe, der schon klar absehbar ist, sondern wo ich eher in einem experimentellen Stadium unterwegs bin und ich vielleicht selber noch nicht genau weiß, wie das Endprodukt aussieht, der agile Ansatz

schon das Problem hat, dass man sich gerne verläuft. Man fängt einfach mal an, setzt was um, um danach darauf zu kommen, dass das eigentlich anders hätte sein sollen. Grade im mobilen Bereich, speziell auch zum Beispiel im bei iOS Apps. Bei einem Webservice kann ich im Gegensatz im Prinzip jederzeit ein Update machen und somit minimale Änderungen deployen. Mehrmals täglich wenn ich will. Das heißt: Ich kann nicht nur in der Entwicklung, sondern auch im Deployment agil sein. Das ist im Bereich von mobilen Apps anders, wo ich zum Beispiel immer einplanen muss, dass Apple vorher den einwöchigen Reviewprozess durchgeht. Kleine Schritte zu machen, was in der agilen Entwicklung das um und auf ist, ist in dem Umfeld um ein großes Eck schwieriger zu realisieren, als beispielsweise im Umfeld von Web Applikationen."

#3 „Wir arbeiten muss ich leider sagen zu wenig mit klickbaren Prototypen. Die Papierprototypen, also meistens sind diese digital, verwenden wir schon. Also sowohl die quasi handgezeichneten als auch die fertig designten Prototypen. Was bei uns noch dazu kommt: Da wir gezielt an unseren Produkten arbeiten und nicht an wechselnden Kundenprojekten haben wir im Laufe der Zeit eine relativ große Community an Benutzern aufgebaut, die wir immer wieder systematisch um Rat fragen. Wir haben ein Forum aufgesetzt, wo wir dann zum Beispiel Screendesigns oder andere Entwürfe reinposten. Wir versuchen damit herauszufinden, welche Features am meisten benötigt werden."

#4 „Gerade im Start-Up Bereich hast du immer wieder Leute die Remote arbeiten, weil es sich aus verschiedensten Gründen nicht anders ausgeht. Meine Beobachtung ist, dass aus einem technischen Standpunkt aus gesehen die Softwareentwicklung funktioniert. Was einfach abgeht ist der soziale Aspekt. (...) Ich persönlich finde es immer gut, wenn die Leute, die mitarbeiten, nicht nur technisch mitarbeiten, sondern auch emotional dabei sind."

#5 „Wir haben sehr viele strukturierte Tests. Wir haben ein Testteam, dass momentan noch zu klein ist. Unsere Zielgröße ist, dass wir ungefähr auf zwei Entwickler einen Tester haben. Das heißt: Das Testen ist bei uns sehr wichtig. Es gibt detaillierte Testprotokolle. Es werden Testfälle organisiert, wo man sagt, dass jede Version ein bestimmtes Testprotokoll durchlaufen muss, bevor es rausgehen darf. Und in letzter Zeit versuchen wir noch vermehrt in Richtung automatisiertes Testen zu gehen, speziell auf der Android Schiene, vor allem getrieben durch die Problematik mit den unterschiedlichsten Devices und unterschiedlichen Screen Größen. Die Geräte verhalten sich teilweise ganz unterschiedlich. Wir versuchen dem zu entgegnen, dadurch dass wir sagen wir investieren Zeit in die Entwicklung von automatisierten Tests, wo wir dann in Simulatoren oder in physischen Geräten automatisiert die App durchlaufen lassen und den User simulieren."

#6 „Die Builds, die reinkommen, stoßen schon Tests an, aber nur ein grundlegendes Set. Also wir lassen nicht bei jedem Commit den Build Server mit 40 Android Devices hochlaufen. Das machen wir dann pro Release. Das würde immer zu lang dauern und wäre auch zu viel Aufwand. Also in einer kleinen Form haben wir Continuous Integration, aber nicht Full-Scale."

A.2.3 Gamma

#1 „Wir haben von Anfang an eine agile Softwareentwicklung gefahren, aber nicht mit dem ganzen Overhead den *Full Blown Implementierungen* von Scrum mit sich bringen. (...) Grundsätzlich machen wir eine lightweight Version von Scrum mit einwöchigen Zyklen."

#2 „Die größten Unterschiede sind die technologischen Rahmenbedingungen und der Markt. Auf der einen Seite habe ich bei der klassischeren IT vielleicht Steuerungssysteme für Flugzeuge

oder ERP Systeme mit einem Lebenszyklus von Jahrzehnten und auf der anderen Seite habe ich Apps mit einem Lebenszyklus von wenigen Monaten. Das heißt: Es ändern sich der Markt, die Technologie und die Rahmenbedingungen ständig und die [Faktoren] sind auch außerhalb des Einflusses."

#3 „Wir verwenden Sales / PO. Also der inhaltlich und fürs Budget Verantwortliche. Wir verwenden PM / Scrum Master. Der Scrum Master heißt bei uns Projektmanager - vielleicht ein weniger fancy der Name, aber er macht genau das: Er sorgt dafür, dass das Projekt läuft und geliefert wird. Ja und dann gibts den Designer im Team und die Developer auf den Zielplattformen. Das QA Team is bei uns insoweit involviert, als dass der QA Teamleiter verständigt wird, wenn ein Projekt da ist. Wenn User Stories erstellt werden sollen, dann leitet er [QA Teamleiter] das in die Wege. Außerdem plant er mit seinem Team, wann das Testteam zur Verfügung steht."

#4 „Wir machen nicht so viel Meeting-Overhead wie es grundsätzlich gedacht ist. Das heißt: Wir machen nicht bei jedem Sprint eine Retro oder bei jedem Sprint ein langes Planning 1 beziehungsweise Planning 2, wie es in Scrum vorgesehen ist. Durch die kürzeren Zyklen ist das nicht notwendig. (...) Der eigentlich größte Effekt im positiven Sinn, den ein weekly Sprint Meeting hat, ist, dass wir dann wirklich versuchen - es muss nicht jede Woche sein - so oft wie möglich den Kunden dabei sitzen zu haben, um sofortiges Feedback zu bekommen und dadurch die Erwartungshaltung vom Kunden total gut gemanaged werden kann."

#5 „Das ist immer total davon abhängig, mit welchen Kunden wir es zu tun haben. Es ist ein Riesenunterschied ob der Kunde Erfahrung mit mobilen Projekten hat. Grundsätzlich ist es so, dass der Kunde in der Regel schon mehr oder weniger konkrete Vorstellungen hat, was am Ende raus kommen soll. Wir formulieren diese Vorstellungen aus. Wir haben mit dem Kunden eins bis X Termine, wo wir Details erfragen. Was ist die Zielsetzung? Was soll mit der App erreicht werden? (...) Bevor wir in einem Feature in der App irgendetwas programmieren gibt es einen Workflow: Also Wireframes von der gesamten Applikation, jeder Screen. Es gibt auch ein abgenommenes Design von jedem Screen bevor dieser implementiert wird. Das geht soweit, dass das Design mit verschiedenen Tools - wir probieren auch immer neue [Tools] aus - nachgestellt wird. Wir haben da jetzt schon brauchbare Erfahrungen mit online Klick Dummys gemacht. Designs die miteinander verlinkt sind und wo sich der Kunde dann durchklicken kann. Wenn das passt, wird das dann entwickelt."

#6 „Ich denke, das ist schwer zu sagen. Wenn die Iterationen drei oder vier Wochen dauern, dann wäre es mir viel zu unflexibel. Bei einwöchigen Iterationen sehe ich den Bedarf nach nicht fixen Abständen eher weniger. (...) Bei uns ist nämlich der Scope im Sprint auch flexibel, der kann mehr oder weniger werden, je nachdem wie die Umstände sind."

#7 „Wenn man räumlich getrennt ist, dann müssen für das gleiche Ergebnis wie bei einer nicht räumlichen Trennung andere Dinge viel besser funktionieren, um den gleichen Projekterfolg zu haben. Die Kommunikation muss spitzenklasse sein. (...) Es muss alles besser dokumentiert sein. Es fängt schon vorne an: Die Spezifikation muss viel klarer sein und man kommt auf Missverständnisse auch viel später drauf. Also es geht schon, aber man muss das sehr, sehr eng und gut managen, damit das ansatzweise so gut funktioniert. Und es macht den Leuten weniger Spaß."

#8 „Grundsätzlich ist es so, dass jeder Entwickler ständig testet. Der Entwickler programmiert etwas und muss dann schauen, ob das grundsätzlich funktioniert. Das heißt: Er muss da eh selber ständig testen. Ganz wichtig ist, dass nicht nur der Programmierer selber testet, sondern jemand Anderer, der das nicht entwickelt hat, weil man mit der Zeit natürlich betriebsblind wird und daher einem Dinge nicht auffallen. Wir haben ein Testteam. Bei uns sind das derzeit

drei Personen, die auch nicht Fulltime die ganze Zeit testen, weil das nicht geht. Ich frage mich, ob es Menschen gibt, die so etwas aushalten. Ich denke, wenn man in einer guten Qualität testen möchte, dann braucht das sehr, sehr viel Aufmerksamkeit. Und das einen ganzen Tag lang zu machen halte ich für praktisch unmöglich."

#9 „Wir haben uns das angeschaut und für uns hat das keinen wirklichen Sinn gehabt, weil der Overhead zu groß ist. Wir machen sehr stark customized Lösungen in kleinsten Dingen. Um da Dinge zu automatisieren müsse ich am Anfang sehr viel Arbeit reinstecken, dir mir aber nur für einen kleinen Bereich etwas bringt. (...) Wir haben uns jetzt aber nicht extrem lange damit beschäftigt."

#10 „Man muss einplanen, dass man bei Apple zwei Wochen für das Review braucht. Dann kann es noch immer passieren, dass Apple nein sagt, weil ein Mitarbeiter gerade schlecht drauf ist an einem Tag. Man hat einfach überhaupt keine Handhabe. (...) Man verpasst dadurch eine wichtige Deadline. Die Gründe warum es abgelehnt wird sind ganz klar nicht nachvollziehbar. (...) Und dadurch hat man dann zum Beispiel drei Wochen an Time to Market verloren, nur weil jemand gerade gedacht hat... ja, heute nicht. Also das ist eine Unbekannte. Da ist es auch ganz wichtig, dass man das kundenseitig so kommuniziert."

A.2.4 Delta

#1 „Der große Unterschied ist, dass wir die Erfahrung gemacht haben, dass wir wesentlich flexibler reagieren sollten. Wir sind nie dieses starre System gefahren. Zu Beginn hatten wir das Wasserfallmodell gehabt, wo wir zuerst die Spezifikation gemacht haben. Dann wird das implementiert und dann abgenommen. Man kommt dann darauf, dass sobald es eine Testversion gibt und diese dem Kunden gezeigt wird, dass man hier sehr schnell Changes beauftragt bekommt, wo man flexibler reagieren muss."

#2 „Aktuell versuchen wir nach Scrum vorzugehen, was aber in die Richtung der Backendentwicklung geht. Da wir ja die Client GUI Entwicklung outgesourced haben, gibt es da nicht die klassischen Dailys, sondern eher On Demand. Wir haben sicher täglich Kontakt, aber nicht die klassischen Scrum Fragen, sondern eher in Richtung Stories oder aktuelle Aufgaben oder Tasks."

#3 „Wenn es wirklich dazu kommt, dass ein Projekt beauftragt wird, dann gibt es bereits einen Prototypen und dann ist dieser Prototyp die Basis für das Requirements Engineering. Dann versuchen wir das klassisch in Use Cases beziehungsweise in User Stories herunterzubrechen und dann die User Story auszudefinieren."

#4 „Es hat alles seine Vor- und Nachteile. Was mir sehr gut gefallen hat, war der Punkt, dass du eine Spalte [im Kanban Board] für Bugfixing vorgesehen hast. Wir sehen, dass das bei uns in jedem Sprint notwendig ist. Selbst wenn es Kleinigkeiten sind, die wir noch in den nächsten Sprint mitnehmen müssen. Das finde ich gut. Wenn jetzt schon die Entwicklung gestartet hat und du die Retros wöchentlich machst, dann glaube ich, dass man in der Praxis, wenn sich die User Stories schon in diesem Zeitraum derart ändern, vielleicht schon früher aufpassen sollte, dass man unsichere User Stories nicht an die Entwicklung gibt."

#5 „Sprache sollte zwar heutzutage keine Barriere mehr sein, aber ich persönlich kenne schon die Situation, in der es zu Missverständnissen kommen kann, wenn die Spezifikation in Englisch verfasst wurde. Was dann in weiterer Folge Telefonkonferenzen beinhaltet - also sehr viel Zeit, um festzuhalten was gewünscht ist."

#6 „Wir müssen wieder unterscheiden zwischen Schnittstellenfunktionalität und reiner mobiler Funktionalität. Zum einen sind wir beim Testen der Schnittstellen an Scrum gebunden. Das heißt, wenn eine User Story abgenommen wird, dann wird diese von uns vorher getestet und dann gibt es das OK vom Product Owner. Wenn es jetzt um rein clientseitige Entwicklungen geht, dann gibt es in unterschiedlichen Iterationen Testversionen und diese Testversionen werden anhand eines Testkatalogs durchgetestet. Das heißt: Wir haben unseren Tester, der dafür zuständig ist, dass er basierend auf den User Stories einen Testkatalog erstellt und der App Entwickler nennt uns dann genau die Punkte die umgesetzt wurden. Diese erfassen wir und dann werden diese Punkte mit jeder Version getestet."

#7 „Das ist ein interessantes Thema. Wir werden dieses Thema demnächst auch angehen, sprich wir sehen sehr viel Potential unsere Qualität zu steigern. Zum Teil auch schneller zu sein und genauer zu testen, aber es gibt am Markt noch nicht so viele Produkte beziehungsweise Anbieter, die das anbieten. Wir sind gerade in der Entscheidungsfindung, wie wir da weitergehen wollen. Es ist ein Thema, aber aktuell ist es manuell... Es gibt denke ich sehr viele proprietäre, individuelle Lösungen. Wir suchen aber eher etwas was alles abdeckt, sprich wir wollen nicht ein eigenes Framework für iOS und ein eigenes Framework für Android, sondern etwas Nachhaltigeres, was unterschiedliche oder alle Plattformen unserer Apps abdeckt."

#8 „Also das was du ansprichst ist ein klassisches Betriebsthema bei uns. Es ist jetzt nicht allzu kompliziert, weil die Entwickler oder Apple vorab Beta Betriebssysteme herausgeben und wir früh genug wissen was kommen wird und damit entsprechend darauf reagieren können. Wir haben erst vor Kurzem das Update von iOS 6 auf iOS 7 gemacht. Sollte bei uns immer so getaktet sein, dass mit dem Tag wo Apple das Update vom Betriebssystem herausgibt, wir unser Update releasen."

#9 „Wir gehen sehr stark in die high-fi Mockup Prototype Entwicklung. Sprich, die finalen Prototypen sollen schon sehr nahe dem Endprodukt sein. Das Problem ist dann, dass sehr viele glauben, dass das Produkt schon fertig ist, nachdem sie den Prototypen gesehen haben. Mit dem kämpfen wir. Wenn wir mal einen so schön aussehenden Prototyp präsentiert haben, dass es dann noch zum Beispiel ein halbes Jahr dauert bis wir live gehen können."

#10 „Es macht den Eindruck, dass du versucht hast Scrum und Kanban zu verbinden - ich finde es gut. Bezogen auf die Rolle des Coachs sehe ich hier nicht den großen Unterschied zum Scrum Master. Was mir sehr gut gefallen hat ist dein Ansatz des Dailys. Dass du das aus Sichtweise der Stories angehst und nicht aus Sichtweise der Personen, die das umsetzen."

A.2.5 Epsilon

#1 „Nein so etwas gibt es nicht wirklich bei uns [Prozessmodelle]. Es gibt natürlich schon Richtlinien wie etwas zu tun ist. Letztendlich sind die Projekte bei uns eher so im Größenordnungsbereich 20 Manntage. 50 Prozent der Projekte werden von einem Entwickler entwickelt, der Rest von maximal zwei [Programmierern] gleichzeitig. Von daher wäre es überzogen, irgendwelche Prozesse festzulegen."

#2 „Also im Gegensatz zu umfangreicheren Projekten sind unsere Projekte eher im kleineren Bereich angesetzt. Es gibt hald nicht wirklich Prozesse und der Entwickler ist auf sich selbst gestellt. Es würde auch keinen Sinn machen irgendwelche Entwicklerprozesse zu definieren, weil das Ganze dann überzogen wäre. Wir haben so viele Spezialfälle - um diese zu berücksichtigen hätten wir dann vermutlich ein 500 Seiten starkes Prozesshandbuch, welches dann eh keiner überblicken würde. Das würde nichts bringen. Letztendlich leben wir davon, dass wir die Projekte, die kleiner sind und natürlich auch weniger Deckungsbeitrag machen, zügig

durchziehen. Es sollten möglichst wenig Probleme auftauchen und das Ganze möglichst schnell erledigt ist.“

#3 „Meetings gibts sonst keine. Es würde nichts bringen, wenn der Entwickler, der am Nachmittag ein Problem hat, dann bis zum nächsten Tag in der Früh beim Meeting warten muss, wo dann das Problem besprochen wird. Was macht er dann den restlichen Tag? Von daher ist es mir lieber er kommt dann, wenn er das Problem hat und nicht dann, wenn das Meeting ist.“

#4 „Das kommt ziemlich stark auf den Kunden an. Manche Kunden wissen schon ganz genau, was sie haben wollen. Die schicken uns dann schon meistens ein Dokument mit dem Design. Vor allem größere Firmen haben bereits fertige Screens von einem Grafiker. Das geht dann ziemlich rasch und es lässt sich auch ziemlich genau bestimmen wie hoch der Aufwand ist. Andere Kunden wiederum haben hald die Vorstellung, dass sie eine App haben wollen. Sie zeigen uns ihre Webseite und erwarten sich von uns eine Beschreibung von dem, was sie dann bekommen werden. Diese Kunden wissen eigentlich überhaupt nicht, was sie haben wollen und sie überlassen es praktisch uns, das zu definieren. Wir präsentieren dann, wie wir uns die App vorstellen, welche Features die App haben wird und die Kunden sagen dann im Großen und Ganzen nur mehr ja oder nein zu den einzelnen Features und daraus ergibt sich dann das Requirements Dokument.“

#5 „Wir haben ein Framework für hybride Apps, die statischen HTML Code von einem Server nachladen. Das wird gezippt, in der App ausgepackt und dann wird der Content aktualisiert. Prinzipiell mag ich hybride Apps nicht. Darum verwenden wir diese nur sehr spärlich, wenn es darum geht, HTML Content praktisch darzustellen. Da geht es dann um minimalen Aufwand für die Erstellung der App.“

#6 „An die Freelancer wird die Anforderung gestellt, dass sie selbstständig arbeiten - noch mehr als bei den eigenen Entwicklern. Von daher hört man eigentlich nicht sehr viel von ihnen. Sie kommen dann selbst wenn es Unklarheiten gibt und sie eben nicht wissen, wie etwas gemacht gehört. Aber sonst hört man eigentlich nie was von ihnen.“

#7 „Eigentlich in Haus sehr wenig - es wird jemand bestimmt, der dann die App durchtestet. Das ist im Prinzip nicht sehr aufwändig. Der schaut dann, ob es irgendwelche offensichtlichen Bugs gibt. Dann wird die App dem Kunden zur Verfügung gestellt, der meistens andere und mehrere Geräte zum Testen hat. Das sind im Großen und Ganzen die Tests.“

#8 „Ich glaube nicht, dass es Sinn machen würde bei uns, weil doch jede App anders ist und sich dann dazu überlegen, bei welchen Eingabefeldern welche Werte eingegeben werden müssen, dass dann auf einem anderen Screen irgendetwas passiert ist zu viel Aufwand - das probieren wir dann lieber händisch aus, bevor wir dann extra einen Testautomaten aufsetzen. (...) Das Problem ist auch, dass Tests keiner zahlt. Der Kunde möchte eben eine App haben und geht davon aus, dass fehlerfrei programmiert wird und wenn jetzt 50 Prozent vom Preis der App fürs Testen drauf gehen würde, dann würde er [der Kunde] vermutlich jemand anderen beauftragen.“

#9 „Das einzig mühsame bei Apple sind die angeblichen 2 Wochen für das Review - hier planen wir immer 2 Wochen ein, damit die App dann freigeschaltet ist. Man darf auch nicht vergessen, dass zu Weihnachten der App Store geclosed ist. Das ist dann in der Planung immer ein wenig mühsam, aber sonst ist das eigentlich kein Problem. (...) Meistens ist es dem Kunden eh bewusst, dass bei Apple ein Reviewprozess erforderlich ist und daher ist oft schon ein Verständnis dafür vorhanden. “

#10 „Der Sinn unserer Apps besteht eher darin, auf den Kunden aufmerksam zu machen. Wo ich Ihren Prozess eher sehen würde, wäre bei einer anderen Art von Apps wie zum Beispiel eine

App für Mitarbeiter von Versicherungen, die ihr Produktportfolio abgebildet haben und wo es Kalkulationsrechner zu verschiedenen Versicherungsprodukten gibt. Wo wirklich Berechnungen im Hintergrund passieren, die normalerweise am Laptop oder Desktop erfolgen würden. Mit der App habe ich dann den Vorteil, dass ich direkt bei Kunden diese App verwenden kann. Das sind aufwändigere Dinge, da wäre glaube ich so ein Prozessmodell wirklich optimal."

A.2.6 Zeta

#1 „Überwiegend versuchen wir Scrum zu verwenden. Wir schaffen es jedoch nicht ganz es durchzuziehen. Wir leben sozusagen in einer adaptierten Scrumwelt, die am Besten für unser Team passt."

#2 „Schnelllebigkeit ist denke ich der größte Punkt. Auch die Agilität, dass man schneller reagieren kann. Was schon auch oft bei uns vorkommt ist, dass Features noch getraded werden [in der Entwicklung]. Sprich der Kunde hat relativ lang Zeit das hundertprozentige Aussehen seiner App zu beschließen. Es fallen dann oft Features raus, die er anfangs gerne gehabt hätte, weil er dann doch lieber in ein anderes Feature mehr Zeit beziehungsweise Geld investieren will."

#3 „Also wir haben auch den Product Owner, das Development Team und als Coach den Scrum Master, der sehr ähnliche Aufgaben bei uns erfüllt, sprich das Daily, die Sprint Plannings und die Retrospektiven moderiert, bei uns aber auch immer Teil des Teams ist. Im Prinzip sind unsere Rollen sehr ähnlich zu den Rollen, die du vorgestellt hast."

#4 „Also wir haben vor dem Kick-Off Meeting eine Angebotsphase und da gibt ein Meeting für die Schätzung des Projekts. Die Schätzung übernehmen bei uns die Entwickler. Im Idealfall die Entwickler, die das Projekt dann auch umsetzen. Meistens setzt sich da der Product Owner mit einem oder zwei Entwicklern aus dem Development Team zusammen und die schätzen die Features. Das ist auch ein kleiner Unterschied, der mir im Gegensatz zu deinem Modell aufgefallen ist. Dass wir im Prinzip das Schätzen beziehungsweise die Story Point Vergabe und das T-Shirt Sizing schon losgelöst haben, weil wir es schon für die Angebotslegung brauchen."

#5 „Ein Kunde will eine Software haben. Das heißt, der weiß eigentlich schon grob was er haben will. Bei uns setzt sich dann meist der PO alleine mit dem Kunden in ein, zwei Meetings, vielleicht auch in einem Workshop, zusammen. Hier wird versucht abzuklären, worum es im Wesentlichen bei der Software geht beziehungsweise was er von uns haben will. Wenn dieses Requirements Engineering abgeschlossen ist, dann geht es in die Schätzphase mit den Entwicklern intern ohne Kunde, nur mit dem PO. In dieser Phase entstehen bei uns auch die User Stories."

#6 „Ich finde die Idee eigentlich nicht schlecht, aber nachdem wir uns sowieso nach fixen Punkten zusammensetzen müssen und sowieso Retrospektiven und Plannings machen ist es bei uns nicht unbedingt notwendig. Bei uns ist im Prinzip das Planning das Meeting, bei dem der PO entscheiden kann, welche Features jetzt implementiert werden. Den Vorteil, den ich bei deinem Modell sehe ist, dass er [PO] noch besser und schneller entscheiden kann, welche Work Items er jetzt wirklich haben will."

#7 „Vor allem gerade bei sehr agilen Prozessen, beziehungsweise bei Prozessen mit dem Kanban Board, die davon leben, dass man sich täglich sieht ist es sehr schwer. Mit einer anderen Firma, mit einer anderen Abteilung zusammen zu arbeiten, das kann man sich dann wieder eher wie einen Kunden vorstellen. Man muss viel strikter sein. Man muss viel schärfer abtrennen. Man kann jetzt nicht leicht Abhängigkeiten abbilden. Sollte zum Beispiel jemand krank werden und du bist abhängig von dem - dann steht vielleicht das ganze Team. Es ist wesentlich schwieriger und erfordert wesentlich mehr Kommunikationsaufwand."

#8 „Im mobilen Bereich ist es so, dass direkt Entwickler testen. Andererseits übernimmt bei uns der PO durch die wöchentlichen Demomeetings einen entscheidenden Part in der Testrolle. Er testet das dann wirklich aus Kundensicht. Zuzüglich testet auch der Kunde mit Testversionen. Unit Testing ist auch noch eine interessante Geschichte - Build Server, die automatisch Unit Tests fahren. Haben wir aber noch nicht so viel gemacht damit. Zur Zeit haben wir viel Entwicklertesting, viel manuelles Testen, Testcases schreiben, etc. (...) Es wäre natürlich ein wünschenswerter Punkt bei längerfristiger Software hier einen automatisierten Testprozess zu haben."

#9 „Apple ist wesentlich komplexer. Man sieht es auch bei den Demoapps, die wir an den Kunden verschicken. Wo du bei Androidgeräten im Endeffekt einfach das Abfallprodukt eines täglichen Programmierens aus dem bin Folder ziehst und das APK einfach dem Kunden schickst. Hier ist Apple wesentlich restriktiver. Da muss man Zertifikate erstellen. Wenn es um Push geht, dann brauchst du eigene Pushzertifikate... Serverzertifikate. Also das ist bei Apple meiner Meinung nach ungemein komplizierter als beim Android Release, dadurch wird die App aber auch sicherer. Durch die ganzen Apple Approving Prozesse steigt schon die Sicherheit und das führt auch dazu, dass es wesentlicher weniger Frauding Apps als bei Android gibt."

#10 „Wir ziehen sehr viel aus Scrum. Den Kanban Ansatz verfolgen wir implizit. Zum Beispiel die Selected Column in dem Kanban Board: Bei uns ist der PO auch bei den Meetings anwesend und entscheidet dann aktiv mit was wichtig ist. Von den Rollen her sind wir auch ziemlich gleich. Also es hat sich jetzt nichts nicht vertraut angehört. Teilweise war das wirklich sehr ähnlich und auch gut daher. Der einzige Punkt, der bei uns nicht funktionieren würde ist, dass das Story Points Schätzen erst nach dem Kick-Off erfolgt. Bei uns ist das Kick-Off erst nachdem das Projekt bestellt wurde und dafür brauchen wir bereits die Schätzung für die Angebotslegung. Das ist aber vermutlich der einzige Punkt, der in der echten Welt nicht funktioniert - zumindest bei uns. Es ist anders, wenn das Projekt firmenintern ist und man einfach die eigene Manpower nutzt. Da brauche ich die Schätzung nicht vorab und da funktioniert das besser."

A.2.7 Eta

#1 „Prinzipiell planen wir keines von denen [Prozessmodelle] konkret umzusetzen. Also wir werden nicht Scrum, Lean oder Six Sigma von vorne bis hinten machen. Wir werden, auch aufgrund meiner Erfahrung aus meinen alten Firmen, ein Mittelding machen. Normalerweise und speziell im Mobile Development setzen wir den ganzen Prozess klassisch auf, also wir gehen die Phasen grob an wie sie geplant sind, aber in den Phasen versuchen wir relativ agil zu arbeiten."

#2 „Was ich bei den Mobile Applications sehe ist, dass die Zyklen meistens viel kürzer sind, weil das Development im Schnitt auch viel abgegrenzter ist. Du hast viel kleinere Applikationen, musst dafür weniger machen, musst diese aber schneller entwickeln. Vor allem wenn du die Applikation mal ausgeliefert hast, sehe ich danach viel mehr Arbeit als bei traditioneller Software wie Web Applications."

#3 „Das Development Team gibt es, das ist klar. Der agile Coach und der Product Owner sind bei uns in einer Person. Ich kenne das natürlich aus Scrum, wo du einen agilen Coach hast, der mit der Entwicklung nichts zu tun hat und sich nur mit dem Prozess beschäftigt. Das glaube ich hat mit der Größe [des Teams] zu tun."

#4 „Nein, es gibt keine [Meetings]. Das einzige was wir machen sind weekly Meetings, wo du quasi das Planning, die Retrospektive und das Stand Up in einem hast. Wir machen eine Art Daily Stand Up nach Bedarf. Wir können es uns nicht leisten jeden Tag 15 Minuten für etwas aufzuwenden, wenn es eh nichts zu besprechen gibt. Das denke ich hängt wiederum stark von der Größe ab und wächst mit der Zeit."

#5 „Also ich empfinde dies grade in Bezug auf Mobile unabdingbar. Vor allem bei Mobile Development sind die Scrum Zyklen mit zwei bis vier Wochen zu lang. (...) Bei Projekten die kleiner vom Aufwand beziehungsweise Umfang sind empfinde ich vier Wochen einfach zu lange."

#6 „Ja auf jeden Fall. Ich halte verteiltes Arbeiten für unumgänglich. Wir machen das nicht nur mit Ressourcen die ganz wo anders sind, sondern auch mit anderen (Resourcen). Auch unser Designer ist niemand mit dem wir immer vor Ort arbeiten und ich glaube es ist mittlerweile auch nicht mehr notwendig, weil die Technologie so gut ist, dass du dir mit Screensharing, Teamviewer, Skype etc. schon ganz vieles ersparst."

#7 „Wir machen nur User Tests, sprich wir haben keine Regression oder Integration Tests. Wir haben Unit Tests ganz unten und wir haben oben User Tests. Automated Testing und Continuous Integration finde ich relativ interessant beim Mobile Development, sehe ich aber insofern als problematisch an, als dass die Zyklen sehr kurz sind und die Pakete meistens relativ klein. Wenn du zum Beispiel Test Driven Development am Mobile machen willst, dann ist das meistens den Aufwand nicht wert, weil so viel Funktionalität hast du nicht, dass sich das auszahlen würde. (...) Beim User Testing kannst du dann die ganzen Usability Sachen abdecken, die du mit einem automatisiertem Test nie einfängst. Daher halte ich Automated Testing und Continuous Integration für sehr fragwürdig bei kleineren Projekten, ob es den Aufwand rechtfertigt."

#8 „Ich halte das Modell für ziemlich gut. Das was wir uns wahrscheinlich lange nicht leisten werden ist einen Coach neben dem Product Owner zu haben, weil es keinen Sinn macht. Ich halte aber vor allem zum Beispiel das Board für eine ziemlich gute Sache, weil es sehr flexibel ist. Das wird vermutlich gut wachsen können, weil ob du auf dem Board Limits von eins und zwei hast, weil du nur zwei Developer hast und zum Beispiel nur fünf Stories im Backlog hast oder ob da Limits von sieben oder acht sind und du im Backlog 25 Stories hast ist völlig egal. Das Konzept bleibt das Gleiche."