The Red Queen's Race

From Delivery Stories to Effort Estimation

an empirical study

Master Thesis
Computer Science, Track: Information Systems Engineering

University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science

September 2012

R.M. Vermolen
s0167894

Master Thesis

From Delivery Stories to Effort Estimation - an empirical study

September 2012

Author

R. M. Vermolen
Student number: s0167894
Program: Computer Science
Track: Information Systems Engineering
School: Faculty of Electrical Engineering, Mathematics and Computer Science
E-mail: renskevermolen@gmail.com

Graduation Committee

Dr. N. Sikkel
Department: Information Systems
Faculty of Electrical Engineering, Mathematics and Computer Science
E-mail: k.sikkel@utwente.nl

Dr. M. Daneva
Department: Information Systems
Faculty of Electrical Engineering, Mathematics and Computer Science
E-mail: m.daneva@utwente.nl

UNIVERSITY OF TWENTE.

During my first few weeks in India I did a lot of reading. Scientific literature during the day and in the evening hours I enjoyed reading fiction. At some point I ended up reading the books of Lewis Carroll, known from her stories about Alice in Wonderland. In a sequel, I encountered something that immediately reminded me of my research.

> 'Well, in *our* country,' said Alice, still panting a little, 'you'd generally get to somewhere else—if you ran very fast for a long time, as we've been doing.'
>
> 'A slow sort of country!' said the Queen. 'Now, *here*, you see, it takes all the running *you* can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!

The Red Queen's Race is an event that appears in the book " Through the Looking-Glass". It involves the Red Queen and Alice, who are running very fast but remaining in the same spot.

With some quick research I found out that "The Red Queen's Race" is often used to illustrate similar situations[1] and this concept also fits the field of Software Development:

You have to move fast to keep up with others.

You will have to move even faster if you want to be the first.

[1] As an illustration of the concept of predestination paradox as described by Isaac Asimov, to illustrate the relativistic effect that nothing can ever reach the speed of light and in evolutionary biology to illustrate that sexual reproduction and the genetic recombination that results from it may be just enough to allow individuals of a species to adapt to their environment (called "Red Queen's Hypothesis").

Management summary

Accurate software effort estimation is a key factor for software development project success. Both the need for and the complexity of correct estimates have increased due to the growing trend for complex software and distributed projects. The introduction of agile software development has added another challenge to accurate effort estimation, this is partly due to the way requirements are engineered in agile projects. This master thesis is concerned with the state-of-the-practice of effort estimation in a large-scale, outsourced and agile project, in order to understand current effort estimation practices and identify possible causes for inaccuracies in those estimates.

The first part of the thesis presents a literature review on large, distributed and agile software development projects. It reports on the specific challenges related to: (i) agile projects, scaling those projects and dealing with distribution of such projects; and (ii) agile RE in large-scale environments and effort estimation in distributed and agile projects.

The state-of-the-practice in the case study project was analysed using the knowledge that was gathered during semi-structured interviews with software professionals. The vendor has organized this agile project based on a planning phase, which included a pilot scrum. The experiences from this planning phase helped to plan the different releases. A remarkable aspect of the application of agile in this project, is the use of the delivery story (DS) artefact. The DS is a design document which incorporates all information necessary for development of a particular piece of functionality. As a consequence, delivery stories also play a pivotal role in effort estimation. All participants in this study are satisfied with the current usage of the DS.

The key findings from this research effort include effort estimation as a recurring task that appears in four different levels of the project. Effort estimation takes place on: project level, DS specification level, DS development level and functional testing level. Estimations usually go through 3 phases, starting with a ballpark estimation. The ballpark estimation evolves into a baseline estimation and the baseline is updated once more detailed estimations are available. Effort estimation seems to work well as it is happening right now. All different occurrences of effort estimation could have added up to a big deviation in total project effort, but our findings indicate this is not the case. Many review- and approval-cycles are combined with a vast amount of software development experience, which leads to fairly good results when it comes to effort estimation. It can be concluded that in this project there is no urge to improve effort estimation, the current practices apparently work well enough.

This research makes two types of contributions: from a theoretical perspective, it contributes to the emerging research on effort estimation in agile, distributed, large-scale projects. In large, agile projects, delivery stories support effort estimation practices and reduce the need for communication. From a practical perspective, this research offers practitioners knowledge about the use of agile artefacts in effort estimation and the value of learning-oriented estimation.

Outline thesis

1 Introduction

Software effort estimation is not only a core task in any software development project, accurate effort estimation is a key factor for software project success. The importance of a reliable basis for making decisions has only increased with the rapid growth in the demand for high-quality software. Combined with the increased complexity of software and the global trend towards distributed projects, software planning and management is essential but at the same time increasingly difficult.

The introduction of agile software development has increased the difficulty that is related to accurate effort estimation. In agile projects, the requirements are developed along with the project and only sketched in a rough manner. This is completely different from traditional software development projects where requirements are developed at the start of the project. This is why it is interesting to know how effort estimation works in a large-scale, outsourced and agile project. This study is concerned with the state-of-the-practice of effort estimation in such a project, in order to understand current effort estimation practices and identify possible causes for inaccuracies in those estimates.

This master thesis project is an extension of an existing research collaboration between the University of Twente (UT) in Enschede, the Netherlands and the Tata Research Development & Design Centre (TRDDC) in Pune, India. The Requirements Engineering team of TRDDC is working together with the Information Systems Group of the University of Twente in an academic collaboration in the field of Agile Software Development. Both research departments are interested in the practice of capturing requirements and estimating effort in agile projects.

The following paragraphs will elaborate on requirements in agile software development, which gives a scope for the project and motivation for the research goal and research questions. This chapter continues with the theoretical and practical significance of this research and concludes with an outline of this master thesis.

1.1 Capturing requirements: User Stories to Delivery Stories

User stories are used in agile software development to help define the functions that a system must offer. User stories are a concise way of capturing requirements, with a focus on the 'who', 'what' and 'why'. Mike Cohn describes a format of "As a <role>, I want <goal/desire> so that <benefit/business value>" [25], which can be extended by mentioning the priority of the story and some acceptance tests on the back of the card.

Delivery Stories (DSs) were developed within the vendor as an extension on User Stories as a means to organizing an agile work style in very large software projects, so-called "Agile-in-the-large". DSs capture the functionality of a story and add details that are needed to plan its implementation. According to a paper by Daneva, van der Veen et al. [53], these include: functional requirements, design specifications, security requirements, dependencies, test scenarios, effort estimation and risk. Other inputs that are used are non-functional requirements and external dependencies.

1.2 Research goal and research questions

Requirements are very important to help cost estimation of software development projects, especially agile projects. Shari Lawrence Pfleeger [65] mentions that for most projects, the biggest component of cost is effort. Next to that, effort is also the cost component with the greatest degree of uncertainty. Therefore, the ability to accurately estimate effort would immediately generate business value for any (agile) software development project.

This interview study will focus on the aspect of effort estimation in agile projects that use DSs as a way to document the detailed requirements in large agile projects. The goal is to determine current effort estimation practices and the possible causes of inaccuracies in effort estimation in a leading large company engaged in outsourcing-contract-based software delivery. This study is part of a larger research initiative in which a future study will follow up on this research topic and will include effort estimation in agile projects that use user stories. The combination of these two research studies is expected to generate valuable insight in effort estimation practices in agile projects. To this end, the central research question in this thesis is:

> How does effort estimation currently happen in agile projects that use Delivery Stories?

This central question in decomposed in the following sub-questions:

RQ1: What is known in published literature about the challenges in requirements-based effort estimation in agile, distributed, and global projects?

RQ2: How does the vendor organize agile projects and who gets involved in the effort estimation process as part of this?

RQ3: How are delivery stories used, at what point of time, and by whom in support of the EE process?

RQ4: What problems are caused by inaccuracies in effort estimation in agile projects, if any? If no problems occur, what does the team do to prevent problems from happening?

1.3 Theoretical Significance: Contribution to Knowledge

This research looks at effort estimation in agile, distributed, large-scale projects. This is recognized as a relatively new field of research [67]. Therefore, the main theoretical significance of this research is to contribute knowledge to research on large-scale, distributed and agile projects. More specifically, the objective of this study is to contribute to knowledge on the value of specific agile artefacts (DSs) and how such artefacts can be used to support the communication between software professionals.

1.4 Practical Significance: Contribution to Agile RE Approaches

Research into new or improved software development methodologies are an on-going effort, not only in research but also in the software engineering practitioners' community. This has fundamental practical significance because of the high failure rate of software development projects. Since the publication of the Agile Manifesto in 2001, Agile approaches to software development have increased in popularity. Insight in the use of agile approaches, especially in large and distributed projects, is of great value to software development professionals. This study presents an analysis of

the use of agile techniques for effort estimation in a large and distributed project, which can be used by practitioners to consider their own implementation of agile techniques.

1.5 Thesis outline

The outline of this thesis reflects the phases of this research. This chapter was an introduction into the topic and the goal and significance of the research. The next chapter provides the context of the case study organization and the project itself. Chapter 3 elaborates on the related work. This section is divided into 5 topics, being: agile software development, scaling agile, distributed agile software development, agile requirements engineering and agile effort estimation. Chapter 4 introduces the research approach. The research methodology was based on the 7 stages of interview investigation [48], which was also used to structure the chapter. Chapter 5 presents the results, organized according to the categorization of the codes that resulted from the interviews. Chapter 6 discusses those results and chapter 7 elaborates on the validity of this research. Chapter 8 provides the conclusions that can be drawn from this research. The final chapter is a reflection of this research project.

2 Context

A qualitative interview study [48] was conducted in which several practitioners within the same project were interviewed. The case study investigated how effort estimation (EE) was done in a large agile project. This chapter presents the context of this research, starting with a description the organization where this research was performed. This section concludes with a description of the project that provided the practitioners to interview for this study.

2.1 Tata Consultancy Services

This case study research was executed in a project of a multinational consultancy organization: Tata Consultancy Services (TCS). TCS offers information technology (IT) services, business solutions and outsourcing services. It was founded in 1968 and is now a leader in the global marketplace and among the top 10 technology firms in the world [1]. One of the main advantages of TCS is the focus on innovation. TCS has a global network of 19 Innovation Labs that collaborate with technology partners and universities to research key emerging trends [2]. The Tata Research Development & Design Centre is one of these Innovation Labs.

2.2 Tata Research Development & Design Centre

The Tata Research Development & Design Centre (TRDDC) was established in Pune, India in 1981 by Tata Consultancy Services (TCS). The TCS Innovation Labs - TRDDC is one of India's premier research centres and houses the largest R&D facility among all TCS Innovation Labs[2].

TRDDC creates tools and processes that simplify the development, maintenance, and management of large IT and engineering systems. The research is divided into 3 distinct areas: Software Engineering, Process Engineering and Systems Research.

The Software Engineering lab consists of different research groups that seek to improve various phases of a software development lifecycle. The Requirements Engineering (RE) research group, that I was a part of, belongs to this Software Engineering lab.

This research group in TRDDC collaborates with researchers from the University of Twente (UT) in the Netherlands, more specifically, with the UT's Information Systems Group. These two research groups share research interests in the area of RE in Agile Software Development projects. These mutual areas of interest have resulted in academic collaborations in the area of client-driven Agile Requirements Prioritization.

As a continuation on the existing research in the area of Agile Software Development, this research aims to gain insights in the process of Effort Estimation (EE) in large agile projects. Collaboration with TRDDC created the opportunity to examine the current EE practices in very large, distributed software development projects.

[2] http://www.tcsinnovations.com/labs/technology-labs-/trddc

2.3 Project context

The project that was used for this case study is part of a typical outsourcing project organization [37]. It consisted of 3 releases and the project was aimed at replacing an existing enterprise system for a large client in the insurance business. At the time of the research, the second release was being finalized and the managers were busy preparing for the third release. The first release took 18 months and had around 55.000 person days of effort. The effort involved in release 2 was similar to the first release, but this was executed in less than 12 months. This is explained by the amount of people that were working for the project, between 200 and 300 for release 1 and a peak team size of 400 people for release 2.

The whole development was done in an onshore-offshore model. The vendor company workforce was distributed between 2 sites, where about 90% of the resources was located offshore and only 10% onsite at the client. There were different channels of communication available between the sites, from video conferencing to telephone and mail. These communication channels were complemented by short term visits between sites. Communication with the customer mainly happened through video conferencing and the other way of communication was during two-monthly workshops with the customers.

All case study participants had different roles in the project. The researchers talked to a business analyst, a tester, a designer, a design lead, a tech lead, the lead solution architect, the delivery manager (who started in this project as a scrum master), the test delivery manager, a delivery assurance facilitator and the programme manager.

3 Related work

This chapter contains background information on literature that is related to this research. Sections 3.1, 3.2 and 3.3 introduce agile software development, agile-in-the-large and distributed agile software development. Sections 3.4 and 3.5 discuss agile RE and agile cost & effort estimation.

3.1 Agile Software Development

The term 'agile' by itself means that something is flexible and responsive, so 'agile methods' can be defined as the "*ability to survive in an atmosphere of constant change and emerge with success*" [11]. It is difficult to give one good definition for agile software development, but all agile methodologies emphasize close collaboration between the programmer team and business experts, face-to-face communication which is considered to be more efficient than written documentation, frequent delivery of new deployable software with significant business value, self-organizing teams and ways to deal with inevitable requirements changes [9]. Put simply, agile development is a different way of managing IT development teams and projects.

The movement of using agile methods in software development already has a huge impact on how software is developed around the world [30]. In 2001, the official definition of Agile Software Development was written down by a group of noted software professionals. The 'Manifesto for Agile Software Development' [7] reads as follows:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- ***Individuals and interactions*** *over processes and tools*
- ***Working software*** *over comprehensive documentation*
- ***Customer collaboration*** *over contract negotiation*
- ***Responding to change*** *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

This manifesto shows the main difference between traditional development (the items on the right) and agile development (the items on the left). The principles behind the agile manifesto [8] highlight the importance of satisfying the customer through: continuous delivery of working software, handling changing requirements, effective and efficient communication between stakeholders, self-organizing teams and simplicity (maximizing the amount of work not done). These agile principles are applied by means of agile methodology: iterative development in the form of sprints, regular meetings (daily scrum, estimation, planning and review meetings, etc.) and agile artefacts (among others a product backlog and sprint backlog).

Agile methods seem to have serious challenges, such as the difficulty of close customer collaboration and the unsustainability of an on-site customer's role for long periods and the difficulty of introducing agile methods into large and complex projects [31]. However, agile methods also have many advantages and can improve job satisfaction, productivity, and customer satisfaction [31]. Because of these advantages, companies are adopting more and more agile practices while searching for solutions for the limitations.

An annual survey on the "State of Agile Development" [81], conducted in 2011, writes that more than 80% of respondents said that their organizations have adopted agile practices within their

software organizations. Nearly half of the responses mentioned that their companies have been practicing agile for over 2 years. This survey shows that interest in Agile Software Development practices are relatively new but ever-increasing in software organizations.

Regardless of the size of projects, the interest towards agile approaches arises mainly from the same needs, but the actual implementation is different [12]. Agile practices require much more tailoring in large companies than in smaller ones [52], and the application of agile on a larger scale can cause challenges that are specific to large agile projects [44]. The next section discusses these challenges of agile-in-the-large.

3.2 Scaling Agile: Agile-in-the-large

Agile methods hold the promise of rapid and efficient software development [12]. Organizations adopt agile practices to become more competitive, improve processes, and reduce costs [14]. Despite the fact that many agile methods seem to be best suited for small and medium projects [19], larger companies also try to gain advantage from the adoption of agile practices [81]. However, large organizations face an additional challenge in the integration of agile practices with existing standards and business processes [14].

Large, complex organizations and projects naturally have many interdependencies that require a large amount of coordination. Therefore, less costly forms of coordination like standardization and planning are easier to control in such situations. Research indicates that most people in large projects and organizations find it hard to implement mutual adjustment coordination on a large scale. Agile methods depend on such mutual adjustment, which is why these methods are usually not preferred for large, complex projects or mature organizations [14].

Mature organizations face another complication that is related to their IT governance frameworks such as ITIL, CMMI or COBIT [14]. These frameworks ensure alignment of IT with business goals and provide structure to IT development and management processes. In a typical agile environment, structure established by a governance framework might hinder project progress [20].

3.2.1 Large-scale development in general

Large projects were very often developed using intensive analysis, design and documentation. Such rigorous development processes were believed to be the only way to deliver the software in these large projects [47]. This approach is being challenged by today's business context, since business change is deemed vital to an organization's survival and evolution in order to adapt to its environment and Ktata and Lévesque mention some of their main concerns for this reengineering [47]:

- **Uncertainty and rapid change** – being too slow to accommodate this uncertainty and rapid change leads to obsolete software and customer dissatisfaction.
- **Obsolescence** - When deployed, many software solutions are already obsolete merely because they do not fit to the new business reality of the host organization. This is caused by the development time and conflicting stakeholders' expectations.
- **Difficulty of software valuation** –Business value is not always tangible and very subjective. The true value of a piece of software can only be evaluated after the software is delivered and it has been used for a while.

Koehnemann and Coats [44] describe several characteristics and challenges of large projects:

- The first challenge is **the overall scope and complexity** of large projects. Large projects typically involve broad sets of both technical and domain knowledge skills and as such, no particular individual will grasp the entire project or even a majority of it.
- Large systems require **documentation** to communicate these system complexities.
- Large projects usually involve **subcontracts** for building or integrating (substantial) parts of the system. This involves team members from different organizations and has consequences for the concept of "team".
- **Reporting, governance and compliance** place additional burden on large projects. For contractual and potential legal reasons, large projects must track the work implemented, when, by whom, the effort it took, etc.

Because of these characteristics, it can be very difficult to apply agile practices to large projects and organizations. For example, agile projects do not use comprehensive documentation, while large projects require documentation to communicate the complexities. This results in a situation where a balance is needed between the limited documentation of agile projects and the extensive documentation of large projects, which is difficult, but not impossible [53].

3.2.2 Agile/hybrid methods in large organizations

Many companies are experimenting with the application of agile practices to projects that involve hundreds of developers and testers that are spread around the globe [3, 12, 14, 15, 32, 34, 44, 47, 77]. They all make adaptations to agile practices or create hybrid approaches to make agile work in their organization.

A study by Koehnemann and Coats [44] presents agile practices that the authors have seen scale successfully to large systems and they provide tips about how to apply these practices in large organizations. The authors mention a valuable lesson learned: "*A best practice is not to use the term* [Agile Methods] *and instead describe the practices as a more common sense, efficient approach for achieving the same development goals*" [44]. Despite some inherent challenges, the authors believe that it is possible to apply agile practices to large systems.

Opposed to that, Barlow et al. [14] state that a purely agile methodology is not suited for general use at large, mature organizations. The authors recommend the implementation of a traditional/agile hybrid solution that will enable project teams to take advantage of the organization's maturity in software development while gaining advantages of agile development such as adaptability to changing requirements.

3.2.3 Challenges related to scaling agile

It is becoming clear that much has been written about adopting agile software development within a large organisation. Next to literature on agile methods that would be suitable, many authors mention challenges and limitations to scaling agile.

Ktata and Lévesque state that misunderstandings between stakeholders are often the reason for the delivery of low value software to businesses [47]. A key aspect in development is to make sure that there is a common understanding of the new technology within all stakeholder groups [3].

While applying agile to large projects, it is possible to break up projects in sub-teams [32] or adopt a 'scrum of scrums' approach [47]. These methods are being used in large projects, but they create another challenge in applying agile practices; the communication between teams.

Elshamy and Elssamadisy [32] try to reduce the need for communication and propose to start by conquering the core project and only divide the project work afterwards. With this method, a significant portion of the architecture and business value is delivered before the work is divided and this creates a higher success-rate of small sub-teams to grow the project into a functional software system. However, an organized way of dividing the work will help to reduce communication, but there will always be a need for good and clear communication between development teams to coordinate the work. Communication is also necessary to identify and prioritize the stakeholders' most success-critical expectations and potential sources of business value [47].

In research on the limitations of agile software processes [80], Turk et al. mention the following limitations:

- Limited support for distributed developments.
- Limited support for sub-contracting.
- Limited support for development involving large teams.
- Limited support for developing safety-critical software.
- Limited support for developing large, complex software.

According to Ktata and Lévesque [47], main reasons for these limitations could be summarized in two main categories tightly related to the applicability of agile practices in large-scale development:

1. Co-location and Face-to-face collaboration issues (Requirement engineering problems)
2. Weaknesses in the decision making process (governance problems)

The next section will continue on co-location and face-to-face collaboration issues, by discussing the field of distributed agile software development as a special case of agile-in-the-large.

3.3 Distributed Agile Software Development

The previous section showed that applying agile on a large scale can create issues with regard to distribution: specifically, co-location and face-to-face collaboration issues. In any large project, some form of distribution of work and teams is necessary to manage the work. For years, many organizations have taken advantage of doing distributed software development projects, which involves cooperation and collaboration between teams located at different locations [23].

It is possible to define three levels of distribution, i.e. local, distributed and global [22]:

- **Local** teams are located at the same site and could meet daily for face-to-face meetings.
- **Distributed** teams do not have the ability to meet personally very often, but they can compensate through technological means, e.g. telephone and video conferencing, to have synchronous communication.
- **Global** teams are located around the globe and have very few overlapping working hours during the day. Communication tends to occur primarily through asynchronous means such as email and file sharing.

Distribution has many advantages. Reduced cost is a main reason for moving into distributed projects [78], but another driver is higher flexibility of competences and resources [28] of talented knowledge workers around the world. However, it is considered unavoidable for software projects with geographically, temporally or socio-culturally dispersed teams to experience unique pressures and challenges [74]. These challenges are related to (asynchronous) communication [74, 78], a reduced media richness [26, 45], coordination and collaboration [74] and a lack of trust [23].

3.3.1 Distributed Agile

"As information technology's role in the modern economy has grown in importance, software developers have found themselves confronted with the challenges of exceptional complexity" [74]. Many software developers have welcomed agile software development [74], as a way to deal with complexity of changing requirements, stakeholder needs, software features and functionality [7, 8]. Agile development promises an increase in productivity [31, 69, 74], innovation [74], quality and value of the software developed [69]. Other benefits of agile development are increased employee satisfaction [74] and customer satisfaction [31].

Distribution promises organizations the benefits of sizable cost reductions [69, 78], higher flexibility of resources and competences to obtain extra knowledge around the world [28, 74], and many other advantages such as: reducing time-to-market, increasing efficiency, improving quality, expanding through acquisitions and reaching proximity to market [74].

Combining distribution and agile is a very attractive possibility, especially in fast-paced and constrained commercial software engineering projects [69]. However, it can be very challenging to apply agile methods in distributed software projects, because the nature of agile projects with co-located team members is quite the opposite of that of distributed teams located at different locations [23, 78]. Table 1 shows some of the fundamental differences between agile and distributed software development [74].

Table 1: Characteristics of agile versus traditional distributed software development [74]

Characteristics	Agile Development	Distributed Development
Communication	Informal Face-to-face Synchronous Many-to-many	Formal Computer-mediated Often asynchronous Tunnelled
Coordination	Change-driven Mutual adjustment, self-management	Plan-driven Standardization
Control	Lightweight Cross-functional team	Command-and-Control Clear separation of roles

A comparison of the characteristics of agile versus traditional distributed software development with the earlier section on agile-in-the-large in mind, shows that agile-in-the-large would fit somewhere in between. Large agile projects are by nature distributed to a certain level and next to the informal agile characteristics they also need some of the formal approach of traditional distributed development.

3.3.2 Potential issues

Because of the fundamental differences between agile and distributed development, conflicts exist between the assumptions underlying the two ideas [69]. Agile distribution is possible, but it is not without challenges.

Teams will have to deal with globally distributing work, because companies attempt to maximize the benefits of such global distribution. This results in a multi-site and multi-cultural working environment, where differences in culture, communication, working practices and vision can become barriers to implementing agility in distributed environments [75].

The geographical separation of stakeholders and teams also makes it difficult to follow many of the fundamental concepts promoted by agile approaches [74]. Some issues that arise when agile is used in global projects are: Team organization, Inter-team communication, Legacy processes, Process refinement and Integration issues [13].

To make agile work in a distributed environment, it is important to tailor agile practices and compensate the lack of co-location and face-to-face interaction through innovative information technology and communication tools [74]. In successful agile and distributed projects, new challenges are recognized and solutions and methods are put in place to overcome these challenges [69].

3.3.3 Overcoming distribution issues

Communication is often mentioned as one of the big limitations of distribution [45, 69, 70]. It is sometimes argued that agile practices can solve these communication problems as long as the agile practices are adapted and augmented to counter the challenges of distributed teams [70].

Other authors also reason that agile processes are very effective in addressing the needs of large distributed projects as long as incremental improvements are used to create successful applications or systems [13]. Table 2 contains an overview of the impact of agile techniques on global projects.

Table 2: How agile affects global projects [13]

Agile Practice	Impact on global projects
Frequent, short iterations with 'demo-able' results	Avoid excessive up-front planning and design work Better handling of unstable requirements Help manage customer expectation
Product owner	Provide a representative for all stakeholders Keep features prioritized according to changing business reality
Continuous Integration	Avoid late discovery of integration issues Improve system level understanding and testing in the project
Develop storyboards and prototypes	Storyboards and prototypes are a powerful way to collaborate with remote domain experts on features. Help test new technology before introducing them in projects
Access to domain experts is critical	Ensuring access to domain experts is critically important for clarifying both the project goals and specific features or user stories

While it can be challenging to combine the principles of agile and distribution, the benefits and rewards of distributed agile are potentially immense. Agile processes bring a certain level of productivity, high value and efficiency, while distribution offers scalability, cost benefits and development size [69]. There are also intangible benefits to the combination of agile practices and distribution: software engineers from different places can work together in the intense way that belongs to agile practices, with the corresponding learning, improvement and eventually greater quality [69].

"*Effective distributed agile development is at its core about finding alignment between the values and principles of agile manifesto and the values and practices of the organization adopting it*." [75]

3.4 Agile Requirements Engineering

Earlier sections already highlighted that organizations adopt agile practices to be more competitive, improve processes and reduce costs [14]. Agile methods can be used to speed up software development [12], which helps to cope with the rapidly changing business environments in which most organizations operate. However, it is exactly this environment that is challenging traditional requirements-engineering (RE) approaches [24].

3.4.1 Requirements Engineering

Requirements are the basis for every project. Requirements define what the stakeholders – users, customers, suppliers, developers, businesses – in a new system need from it and what the system must do in order to satisfy that need [36]. RE is the process of establishing those needs, both the services that the customer requires from a system and the constraints under which it is developed and operates [27].

The RE process consists of five main activities [46]: Elicitation, Analysis and Negotiation, Documentation, Validation, and Management. In traditional RE methods, work begins with the elicitation and documentation of a "complete" set of requirements. This will be followed by documentation of the architectural and high-level design, development and inspection [27].

The main goal of a RE process is creating a system requirements document for knowledge sharing [27]. The aim of RE is to help to know what to build before system development starts in order to prevent costly rework [64]. Once these requirements are communicated and agreed upon between stakeholders, the requirements drive the project activity. Requirements therefore form the basis for: Project planning, Risk management, Acceptance testing, Trade off, and Change control [36].

Traditional RE has some reported challenges, namely Overscoping [17], Communication gaps [18], Keeping the Software Requirements Specification (SRS) updated, Development Work Monitored from Requirements and Manual Selection of Requirements for Products [16]. Next to these challenges, the traditional way of RE has been challenged by the rapidly changing business environment in which most organizations operate today [68]. This changing environment is one of the reasons that companies make the move to agile software development and with that they need to consider RE from an agile perspective.

3.4.2 Move towards agile RE

RE and agile approaches are often seen as being incompatible. RE often relies heavily on documentation for knowledge sharing and agile methods focus on face-to-face collaboration between developers and customers to share knowledge [64]. The requirements in traditional RE are documented in specific requirements artefacts by RE specialists. This is usually done in a phase that is separated in time from design and development. In contrast, the detailed requirements in agile RE are defined gradually in interaction between the customer and the developers [16].

It has been suggested by proponents of agile methods that because of constant interaction with the customer throughout the agile development process, teams are likely to achieve higher customer satisfaction [68] in addition to speeding up the development process [12]. However, agile RE is often less formal and requirements are not always documented [16]. This informal nature of agile RE practices may be considered unacceptable [63]. Agile methods and RE are pursuing similar goals in key areas like stakeholder involvement. The major difference can be found in the emphasis that is put on the amount of documentation needed for a successful project [64] This is supported by Ramesh et al. [68] in the table below that shows traditional and agile approaches for RE practices.

Table 3: Traditional and agile approach for RE activities [68]

RE activities	Traditional RE approach	Agile RE approach	Agile practices used to support the RE activities
Requirements elicitation	Discovering all the requirements upfront	Iterative: requirements evolve over time and are discovered throughout the development process	Iterative RE Face-to-face communication
Requirements analysis and negotiation	Focus on resolving conflicts	Focus on refining, changing and prioritizing requirements iteratively	Iterative RE Face-to-face communication Constant planning Extreme prioritization
Requirements documentation	Formal documentation contains detailed requirements	Little formal documentation. Focus on the essential instead of volume.	Face-to-face communication
Requirements validation	Focus on the consistency and completeness of requirements document	Focus on value creation by ascertaining whether the requirements reflect current user needs	Review meetings Face-to-face communication

Table 3 shows that even though the activities are still the same, the focus of and the specific way to execute these RE activities are different. An important difference can be found in the amount of documentation that is needed for each type of RE approach. To support the RE activities without generating (too much) formal documentation, several agile practices are used. The most important agile practice is the use of frequent feedback cycles with client involvement using face-to-face communication, which decreases the need for formal documentation and increases the customer involvement [64]. Another practice is the use of iterative RE. Instead of discovering all requirements upfront, the iterative aspect of agile RE lets requirements evolve and new requirements can be discovered throughout the development process. According to De Lucia and Qusef [27] "*the main difference between traditional and agile development is not whether to do RE but when to do it*."

3.4.3 Agile approaches from an RE perspective

Agile RE focuses on essential documentation instead on the volume of the documentation, with user stories [4] as the most visible representation of requirements in many agile projects (see section 2.4). Defining requirements with user stories is a way to facilitate the communication between business and engineering roles, and increase the probability of capturing and meeting the customers' expectations [16].

Next to the user story, there are several RE techniques and practices that are used for agile approaches to RE [64]. These techniques include intensive customer involvement, interviews to provide access to the needed knowledge, prioritization of features that deliver the most business value, a limited amount and scope of documentation which increases the chances that the documentation can be kept up to date when the software is changed and frequent review meetings and acceptance tests for requirements validation.

"*The processes used for agile RE vary widely depending on the application domain, the people involved and the organization developing the requirements*"[27], but Cao and Ramesh have identified 7 agile RE practices that are used in business environments [24]. These practices are summarized with an overview of the benefits and the challenges in table 4.

Table 4: Agile RE Practices (based on research by Cao and Ramesh [24]

Agile RE practices	Benefits	Challenges
Face-to-face communication over written specifications	• Customers can steer the project • Decreased need for time-consuming documentation and approval processes	If high-quality interaction is not possible, this approach poses risks for inadequate or wrong requirements
Iterative requirements engineering	• Improves relationship with customer • Requirements are clearer and more understandable	• Cost and schedule estimation • Minimal documentation • Non-functional requirements
Extreme requirements prioritization	Involved customers generate a better understanding of priorities	Continuous reprioritization can lead to instability
Managing requirements change through constant planning	The early and constant validation of requirements largely minimized the need for major changes	This could lead to redesign of the architecture after early cycles, which could add to project cost
Prototyping	Improves communication with customers to validate and refine requirements	• Maintaining or evolving prototypes is difficult. • Could lead to unrealistic expectations of customers
Test-driven development	Tests capture complete requirements and design documentation that are linked to production code	A major challenge to TDD adoption is that developers aren't accustomed to writing tests before coding
Use review meetings and acceptance tests	The review meeting help to obtain management support by providing frequent updates on project status and progress	Implementing acceptance testing is difficult due to the difficulty of access to the customers who develop these tests

The findings of Cao and Ramesh support Bjarnason et al [16], who wrote that agile might solve some challenges of traditional RE, but it also creates new challenges. One of those challenges are problems with cost and schedule estimation [68], which will be further discussed in the next section.

3.4.4 Agile RE practices in large-scale environments

Agile methods can play an important role in the management of large projects [27]. Larger parts of the project can be decomposed and the sub-components can be developed by agile teams in other time zones and other countries, provided that frequent communication and self-organization are established.

As mentioned in the earlier section, agile practices remedy several challenges and issues that are related to traditional RE in large-scale software development, but they also create some new challenges. For example, Bjarnason et al. [16] point out that transferring an organization to agile RE practices in itself is a significant challenge that requires major mind-set changes.

3.5 Agile Effort Estimation

Software development effort mainly consists of human effort that is needed for high-level design, detailed design, coding, unit testing, integration testing and customer acceptance testing. Effort can be seen as a surrogate for software development cost, because cost for the effort of personnel is the main cost in software development [10].

Software development effort estimates are the basis for project bidding, budgeting, planning [35],process and productivity improvement, change management [79], investment decisions, risk analysis and expectations management [62]. There are a lot of methods which can be used for effort estimation, a few examples that are used frequently are [40, 62, 66]:

Parkinson's estimation	Estimation based on the available resources.
Top Down estimation	Start estimating using the general functionality.
Bottom Up estimation	Start by considering the estimation of the required system components.
Estimation by analogy	Estimation based on a comparison with similar projects.
Expert judgment	Estimation based on intuition and experiences from other projects.

Good effort estimation is critical to any project. Pessimistic budgets and plans can lead to lost business opportunities, while over-optimism can result in significant losses [38, 39]. However, it is difficult to provide accurate effort estimates. One review study mentions that 70-80% of software development projects overrun their estimates with an average of 30-40% extra effort [60].

The quality of the outputs of the estimation process is heavily dependent on the quality of the inputs [57], which means that it is difficult to get an accurate estimate at the beginning of a project (when the value of an accurate estimate is the greatest). Other reasons for inaccurate estimates can be categorized as either technical or human [54]. Technical factors are concerned with models and methods for estimation that were described earlier. Human factors include: practitioners optimism [41], unrealistic self-confidence, hidden agendas [42] and organizational issues such as problems created by management and politics [55].

3.5.1 Estimation in distributed projects

Project distribution has intensified over the past years, where reducing development costs was one of the driving reasons for this development practice. This explains why software development effort estimation has a very important role in achieving a reduction of costs [51]. Accurate effort estimation is considered crucial to software development project success, especially in globally distributed projects [61]. Despite this importance, a limited amount of research on effort estimation in distributed projects is available.

A study by Peixoto et al. [51] studied the choice of specific estimation techniques in globally distributed projects. They concluded that people select the estimation methods based on what they know and not based on what is best for the project. Another main conclusion was that even teams in the same country, organization and development methodology choose different estimation techniques to estimate effort.

More recent research by Ramasubbu and Balan [67] continues on their earlier research on aspects of globally distributed software projects. The researchers state that early stage project cost estimation remains a significant challenge. Such early stage effort estimations are crucial for the profitability of projects, especially in a fixed price contracting model [67]. The researchers identified problem root causes of poor project effort estimation at their case study companies:

1. Missing information
2. Simulation capabilities
3. Lack of experience with current tools

The solution to these root causes was the introduction of "*learning-oriented cost estimation*", which is "*a case-based reasoning approach to tap into existing organizational memory (past project experiences available across firms) to help less experienced managers learn from other projects*" [67]. This solution is general enough to be applied to other distributed software development projects. As can be seen in chapter 5, this solution was also applied in the project that was used for this research.

3.5.2 Estimation in agile projects

The application of effort estimation methods in agile projects is a very important task, but also a very difficult one. Classical estimation methods need well defined requirements. Agile methodologies don't offer well defined requirements at the start of a project [71]. In agile software development projects the requirements are developed along with the project and only briefly written down in user stories [5]. The user stories may change during the project and such change requests are an important challenge. Methods for agile effort estimation must support the brief and ever changing requirements in such projects.

An effort estimation method that is typical for agile projects is planning poker, which belongs to a set of group estimation techniques [59]. Planning poker is used as a method to guide estimation meetings with multiple stakeholders. The group discussion in planning poker helps to identify activities that individuals might overlook. This results in more accurate estimates and reduces the over-optimism that is typical in expert judgment-based methods [56]. A study by Mahnič and Hovelja on the accuracy of planning poker showed that expert estimates from a planning poker meeting were much closer to the actual effort spent. The planning poker estimates tended to be more accurate than the statistical combination of their individual estimates [56].

Research by Børte et al. [21] argue that social interaction plays a crucial role in software effort estimation. The researchers call the step from reasoning to decision-making the“ magic step” and believe social interaction might be the answer. Currently, individual knowledge is the dominating orientation for the specification of the work that is needed to solve tasks. This knowledge becomes activated by the participants during social interaction, which is why planning and estimation should be understood as a social practice.

Planning poker is one way to deal with the difficulty of brief and changing requirements. Another way to deal with difficult effort estimations is to work with more feedback and change. Estimations should be executed frequently, but within a very short time [71]. This would provide more precise predictions as a project evolves and more information becomes available.

4 Research approach

This section covers the selection and the use of research methodology in this research project. It will continue with a description of the design of the interview protocol and the execution of the interviews. After an explanation of the capturing and analysis of data, this section will be concluded with an elaboration on the structuring and modelling of the results.

4.1 Research methodology

This research project adopted a case study research approach. A case study gives you a chance to analyse one aspect of an individual unit [58], usually a real world problem. There are different types of case studies, depending on the purpose of the research. There are exploratory, explanatory and descriptive case studies [83]. This thesis research is constructed as an exploratory case study, which is a type of research that is suitable for a problem that has not been clearly defined. Results of exploratory research can provide substantial insight into a given situation [76]. This type of case study is most suitable for this research, because this is an initial research to look for ways in which effort estimation is currently being done in a large-scale, agile project. In this type of research you start with collecting the data after which structured analysis will help to make sense of that data. The goal is to come up with a model to view this data.

There are different strategies for the selection of samples and cases [33]. A random selection would allow for generalization for the entire population, while information-oriented selection maximizes the utility of information from small samples and single cases. In the case of information-oriented selection, "*cases are selected on the basis of expectations about their information content*" [33]. This research used information-oriented selection because the scope was limited to agile projects. With this type of selection in mind, there are different types of case study designs, from single-case to multiple-case designs. Single-case designs are vulnerable, because you will put "*all your eggs in one basket*"[82]. But even though multiple-case designs may be preferred, Yin [82] states that the single-case study is an appropriate design under several circumstances. One of these circumstances is when the case represents a *unique* case where a new phenomenon is researched. During this research, the vendor company only had one agile project in which DSs were used. Because of this unique case, the researchers decided to use a single-case design.

In this case study, the team of researchers used a multi-method approach, with interviews to collect the data and a Grounded Theory approach to analyse the data. The data analysis was done manually, without the help of specialized software programmes. There are many more possible sources of evidence in case study research [83]. Due to practical limitations data gathering was done only by means of interviews. However, for the purpose of getting an overall picture of how effort estimation is embedded in the software development process, we don't think that studying additional documents would have yielded essentially different insights."

This interview research was organized into discrete steps, using the 7 stages of interview investigation according to Kvale and Brinkmann [48]: thematizing, designing, interviewing, transcribing, analysing, verifying and reporting. The next subsections will explain the research methodologies in more detail, using these 7 stages.

The research team split up the work as follows: the author of this thesis designed the questionnaire in collaboration with a TRDDC researcher. The questionnaire was reviewed by two senior researchers at UT and piloted with one interviewee. Two TRDDC researchers helped to gain access to contacts willing to participate in the study. One TRDDC researcher accompanied the author of this thesis in the interview sessions and ensured the scripts reflect what the interviewees indeed have said. The author of this thesis created all transcripts, which were checked by the researcher from TRDDC. The analysis took place in multiple stages. Three researchers (the author, the TRDDC researcher and one supervisor from UT) worked on the analysis in relative isolation, after which they discussed their analytical results over email. The author of this thesis coded all transcripts, the TCS researcher also coded parts of the transcripts and the UT supervisor coded 5 interviews. The author combined all codes to create a final set of categories and codes, creating a mind-map to structure the results. The results, discussion, validation and conclusion were generated and written down by the author and improved with feedback of two senior researchers from UT.

4.2 Thematizing

The first stage of any interview project is *thematizing*. "*Thematizing refers to the formulation of research questions and a theoretical clarification of the theme investigated*" [48]. In this stage it was about the *why* and *what* of the research. The purpose of this research was written down in the research goal (see section 1.1):

> *This interview study will focus on the aspect of effort estimation in agile projects that use delivery stories as a way to document the detailed requirements in large agile projects. The goal is to determine current effort estimation practices and the possible causes of inaccuracies in effort estimation in a leading large company engaged in outsourcing-contract-based software delivery .*

Literature on effort estimation was used to obtain some form of theoretical clarification of the topic that would be investigated. This literature was partially acquired from some researchers with previous knowledge of the effort estimation field. The set of literature was further increased using an unsystematic search for recent publications.

4.3 Designing

Designing the interview involves the "how" of the study [48]. This included both the design of the questionnaire as well as the planning of the interviews with the practitioners.

Interviews as a source of evidence have both strengths and weaknesses. Interviews are targeted and insightful, but weaknesses include a bias due to poor questions, response bias, incomplete recollection and reflexivity [83]. These risks have been addressed in the design of this research.

To prevent a bias due to poor questions, the questionnaire has been designed by 2 researchers (the author of the thesis and the TRDDC researcher) with a different background (where one had more practical experience and the other more theoretical experience). After several iterations of questionnaire design and feedback from senior researchers, the questionnaire was tested in a pilot interview with one practitioner that used to work in the project. It was not necessary to change the questionnaire design after this pilot interview.

Due to cultural differences there was a possibility that a response bias would arise. Even though a response bias mostly occurs in the formulation of a question, it was not clear how much the cultural differences would affect the answers from the respondents if they would answer with what they thought the interviewer wanted to hear. To prevent misunderstandings due to pronunciation-differences as well as the meaning of language (both body language and spoken language), the interviews were conducted by 2 interviewers (with one from the local culture).

To rule out incomplete recollection, the interviews were recoded and both interviewers made notes of striking behaviour or remarkable comments. Reflexivity arises when an interviewee is influenced by the interviewers (or other observers). This was difficult to take into account during the design stage. We became more aware of reflexivity during the interview stage.

It was difficult to plan the interviews with the practitioners and we would not have been able to do this without the involvement of a high-level manager in the organization. This manager organized 2 days of non-stop interviews with participants in the project. We made sure to ask for practitioners with different roles in the project but since these interviews were arranged based on availability of practitioners, we didn't have a say in the selection of interviewees.

4.4 Interviewing

A research interview is an interpersonal situation, a semi-structured conversation between the interviewer and the interviewee [48]. A semi-structured life world interview seeks to "*obtain descriptions of the life world of the interviewee with respect to interpreting the meaning of the described phenomena*" [48]. According to Kvale and Brinkmann, the interview should have a sequence of themes to be covered, with some suggested questions but openness to changes of sequence and the form of questions.

In order to comply with these recommendations, the questions in the interview protocol (see Appendix A) were ordered to different themes. Each section of the interview contained suggested questions and hints for follow-up questions if applicable. The questions were complemented by precise script, which was read to each participant during the interview to introduce the new themes and the accompanying questions. The interview guide was not always strictly followed, because some questions could not be answered by every interviewee. The two interviewers (the author of the thesis and the TRDDC researcher) decided during the interview which questions to ask, when to ask one or several follow-up questions and when not to ask a question.

To establish a good contact with the interviewee, each interview was introduced by a briefing. During this briefing, the interviewers introduced themselves, explained the purpose of the interview, asked for permission to use a sound recorder, and so on. The interviewers finished each interview with a short debriefing, where the interview was rounded off and the interviewee had the opportunity to say something more or to ask questions.

As mentioned before, these interviews were conducted across cultures and this "*may involve different norms for interaction with strangers concerning initiative, directness, modes of questioning, and the like*"[48]. The interviewer that was in a foreign culture already spent some time in this new culture, but to prevent the influence of verbal and non-verbal factors [43] a second interviewer (from that culture) was present.

The manager that arranged the interviews with the practitioners expressed the wish to sit in on the interviews to observe and learn. This might have influenced the interviewees, but it is not possible to determine how.

4.4.1 Conducting the interviews

In total, 11 interviews were conducted by the 2 interviewers with participants in different roles in the project (see table 5). All interviews were conducted face-to-face, on-site of the project. The interviews took around 40 minutes, with a few outliers as can be seen in the table. The reason for these outliers can be found in the roles of the participants in the project. In case of participant 2 and 3, the interviewers concluded during the interview that the interviewees had never been aware of any effort estimation. It was still possible to ask questions about general processes within the project, but the duration was considerably shorter because a large part of the questionnaire was not applicable for these interviewees.

More information on the roles of different stakeholders that participated in this project and the stakeholders that were mentioned during the interviews can be found in appendix B.

Table 5: Overview of the interviews

Participant	Participant Role	Experience	Duration	Words
0	Business Analyst	*[3]	45:49	4178
1	Business Analyst	4y	52:31	5569
2	Tester	1y 8m	06:30	803
3	Designer DSs	2y	09:34	1143
4	Tech Lead (=Design Lead)	2+ y	37:15	4448
5	Tech Lead	4y	31:37	3655
6	Lead Solution Architect	15y	38:14	5383
7	Delivery/Release Manager / SCRUM master	14+ y	39:59	4846
8	Test Delivery Manager	17+ y	23:26	2342
9	Delivery Assurance Facilitator	8y	18:13	2677
10	Programme Manager	15y	49:39	5934

4.5 Transcribing

There are two important requirements for transcribing [48], recording the interview and making sure that the recorded interview is audible to the person that is making the transcript. All interviews in this research were recoded using the audio-recording option of a smartphone. The interviews were performed in a closed-off conference room and the only disturbances on the recordings were the door opening and closing.

The recordings were transcribed by one of the researchers and checked by another to make sure everything was written down correctly. The interviews were transcribed in a partial verbatim style, because in a complete formal, written style the interpretation of the transcriber could influence the original meaning of what was being said by the interviewees. The interview transcripts are available in a separate document.

[3] This participant does not have any recent experience in the project, but was involved in the project during the first release. This interviewee was selected for the pilot interview because she could answer the questions based on her earlier experience and she was located in an office in the same city as the researchers.

4.6 Analysing

To organize the interview texts and find meaning into what was said, this research used the approach of coding and categorization. Coding involves "*attaching one or more keywords to a text segment in order to permit later identification of a statement, opening it for quantification*" [48]. The goal of coding is to compare codes and develop categories that capture the data, leading to saturation when no new insights and interpretations emerge from further coding [48]. Coding can take the form of categorization, when the codes are put in categories to create an overview of the main themes. Codes play an important role in Qualitative Data Analysis (QDA) within the Grounded Theory Approach [29].

Grounded Theory is a methodology, a way of thinking about and conceptualising data [49]. The purpose of Grounded Theory is to develop theory inductively [48], which describes processes such as those in organizations [6]. Grounded theory is often used in social science disciplines, but it is also an excellent method for studying software engineering [6]. Grounded Theory was used in this research because this approach is useful in areas that have not been previously studied.

QDA can be described as a process of *Noticing, Collecting and Thinking* about interesting things with a process that is iterative and progressive, recursive and holographic [73]. *Noticing* includes interviewing to collect data as well as coding the data. In the first stage of data analysis, coding resulted in dozens of codes without any consistency between different interviews. *Collecting* and sorting the pieces of the puzzle is the next step, for which it is possible to use a "grounded theory" approach. During the coding of the first interviews 3 researchers worked together to code transcripts. Comparing the coded transcripts led to a more complete overview of applicable codes, which increased internal validity. For the first interviews mind-maps were used to create overviews of (the relation between) the codes within each interview. This resulted in some preliminary categories in which the codes were organized. At this stage there was still no real consistency between different interviews, although a big set of codes reappeared in every interview.

Data analysis is more than coding and sorting, *thinking* is important to examine everything that has been collected, look for patterns and to make discoveries [73]. *Thinking* can also be supported by a grounded theory approach. After 6 interviews it was clear that no new codes would appear in the remaining interviews. The mind-maps of these interviews were combined by one researcher to create one 'master'-mind-map, with a final set of categories and codes. The complete mind-map can be found in appendix C. These codes were used for the remaining interviews and the categories were used for the last stage in this research approach: reporting.

4.7 Validating

Validation of any research is important to judge the quality of research designs [82]. There are several principal issues of validation of interview research, concerning objectivity and reliability [48]. Validation is mentioned separately because of its importance, it does not belong to a separate stage in the investigation. Each stage in the interview investigation entails certain validity issues, which have been addressed by the validations in each of the subsections of this research approach.

Four approaches to validity are common to case studies: construct validity, internal validity, external validity and reliability [82]. The possible threats to validity of the observations and conclusions will be discussed further in section 7 in this thesis.

4.8 Reporting

There are some common issues regarding research reports that should be considered while working on a report [48]. To guarantee the ethics of reporting, informed consent was taken into account before the interview situation and confidentiality was reckoned with during transcription of the interviews.

It was suggested that interview reports would become more readable if the final report was taken into consideration from the very start of the research [48]. The standard format for reporting interviews (introduction, method, results, and discussion) was used to structure the report. The results were structured using the codes that surfaced during the analysis. The codes were categorized and visualized in a mind-map, in which the categories led to the corresponding subsections of the results as seen in figure 1.

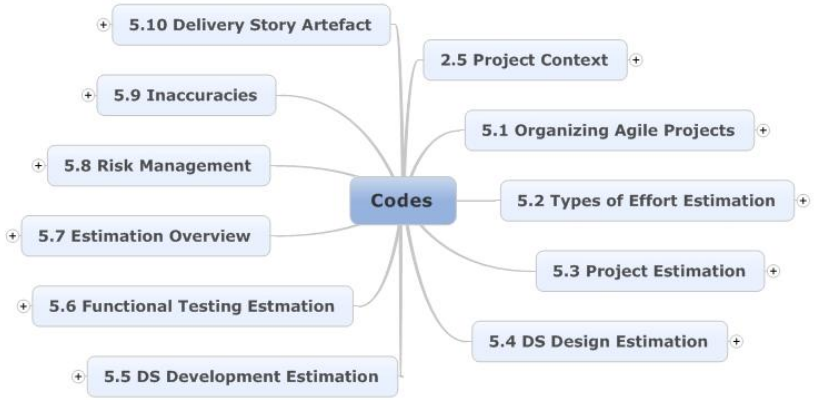

Figure 1: Code categories

5 Results

This section presents the results of the data analysis. It is structured according to the categorization of the codes as it was described in the previous section.

5.1 Organizing Agile Projects

Part of this study was to get an overview of the way the project was organized. The vendor company first started with an integration project at the client. At that time they were not part of development. From the day that the vendor company got involved in the development, they started with an agile project approach. This section describes different steps in the project, which will help to put the results in the next sections into context.

5.1.1 Creating agile artefacts

Agile artefacts are an important part of the agile project approach. The agile artefact that has been used in this project is an elaborated version of the user story artefact as described in section 2.4. At the beginning of this project the client created elaborate user stories and delivered these to the vendor. The lead solution architect gave a good view on the way (elaborate) user stories are produced (see quotation 1).

> *The way USs are produced at the client is… it's a process decomposition with different levels of process. […] So each process is made up of activities. Each of those activities, or a set of activities, is taken up to form a user story. […] How a EUS [elaborate user story] is produced is the … a user goes into the activity and say, defines all the business rules. That is a detailed US.*

Quotation 1 – P6

These user stories were taken as a starting point for the design, which was added to the stories. This turned them into DSs, an extension on user stories as a means to organize an agile work style in very large software projects (also see section 2.4).

> *Now, delivery story is the next step where we try to look at it in two aspects, functional and technical. Functional defines the translation of the requirements in US into the system. It's the same function but translated into a system view… […] Then comes the technical specification. Technical specification is where you look at all the technical aspects…*

Quotation 2 – P6

The functional specification is done by the business analyst team. Once the functional specification is firmed up, it gets approved and the design team gets into the technical specification and the test scenarios. If the design team encounters contradictions in the functional specifications they will go back to the BA and ask for clarifications. If the BA team are not aware of the issue it is even possible to go back to the Subject Matter Experts. Once the design is finalized, the DS moves on to the test scenario's that will complete the DS. The technical specification is more like a high level design, it gives several components: short procedures and screens to be developed, services and entities to be created and the components to be updated. Figure 2 contains an illustration of the process of creating agile artefacts, including all actors that are involved in the process.

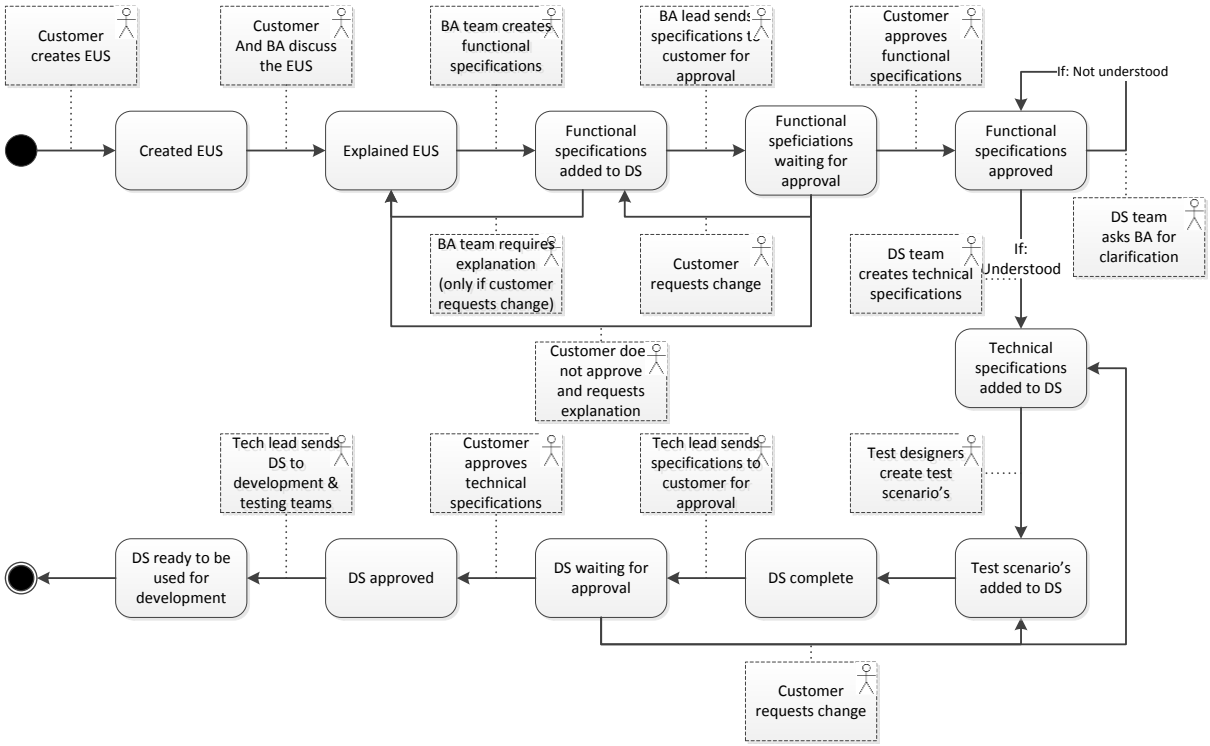

Figure 2: Creating a DS

Every user story has a corresponding DS. To capture all details in the DS, every component will be put into a subsection of the DS.

> *What we do is, inside the DSs we try to create a sub section for each component at a high level. For example, if there is a UI and it's a complex UI, we try to break it into tasks and there will be a separate section for each task.*

Quotation 3 – P7

The DS artefact, used as design document and for estimation and planning, is only used for the core development functions of the project.

5.1.2 Applying agile

The project started from scratch with an "*approach to try it out and then form the base line and then start from there*" [P7]. To create a solid baseline for this project, a 3 month planning phase was included. During this planning phase DS samples were created. These samples were then used to conduct a pilot scrum.

> *Before the actual agile had started, we had conducted a pilot agile for about one month. To find out how it works, are there any barriers and to find out the average velocity of the scrum. […] We had taken a variety of user stories, easy, medium and complex, and conducted those.*

Quotation 4 - P0

The experiences from the pilot scrum were used to organize the agile project. As mentioned in section 2.5, this project will be finished after 3 separate releases of which 2 releases are now completed. A graphical representation of this project from the beginning until now is shown in figure 3.

Figure 3: Agile project organization

Every release consisted of several collections. A collection consists of three 2-week sprints and a 2-week consolidation period. That period is used to finish up any loose ends and make sure that the result at the end of the collection is deployable. The amount of collections in a release can change, but at the end of a collection there is always some deliverable to the customer.

For each of the layers in the figure there are different people involved in the project. The programme manager is responsible for the complete project, which includes all the peripheral systems next to the core development. The lead solution architect is also working on the complete project, consolidating estimates and planning. For every release there is a release manager, who is responsible for the delivery of that specific release. Collections do not always have to be executed in sequence, some collections can also be developed in parallel. Therefore, each collection has its own team of designers, developers, business analysts and testers.

5.1.3 Approach

The project organization as discussed in the previous section is based on an agile approach to software development. However, this is also a large scale project that does not allow a complete agile environment. Agile-in-the-large requires a certain level of documentation and the distribution of developers and customers will add to a more formalized communication compared to traditional agile. During the interviews it was pointed out that this project is not completely agile.

> *"What we are doing here is not a complete agile environment, this is a hybrid model and we are doing this for a large scale program actually. Typically agile is used for product development, which suits here, but then considering the client requirements, constraints what they mean through… in this project before we started engaging. We suggested a hybrid approach. It's a combination of agile and large program development. Some other things we do, it will not follow the strict agile principle…"*

Quotation 5 – P7

In the first collection of release one, the development teams did start with estimation based on story points.

> *what happened during the collection 1 of release 1, we did started estimating during the sprint using the story points, just to see how it goes. So at that point, after a couple of sprints we stopped that practice because it was taking a lot of time: the entire team sitting and doing the estimation… and the reason for that was… we were not achieving the velocity we wanted to achieve. The team was just focusing on the estimations… Ideally it should be done in the first half of the sprint, the first day of the sprint, but then the team was taking almost 2 or 3 days to form an estimate and then coming up and saying I can only finish 6 or 7 stories…*

Quotation 6 – P7

This was not helping the project, since the price and schedule were already fixed.

> *we could achieve only 60 or 70 % of our target during the first 3 collections, so we were lagging behind. So we changed our approach of making that estimation…*

Quotation 7 – P7

The next sections will contain more elaborate descriptions of the approaches to estimation in different parts of this project. However, based on what has been written down in this section it is already possible to point out that the use of agile artefacts and the project organization are definitely part of the agile approach to the core development of this project.

Peripheral functions

This thesis is about estimating effort in agile projects, but this thesis would not be complete without an explanation what is not part of this research. Even though an agile/hybrid approach has been chosen for the core development, there are other things that have to be considered for complete project estimation. These functions do not use the DS concept, for example, reports, correspondences, all the peripheral systems and interfaces. These functions are all part of the total project effort, but they have nothing to do with the agile approach of the software development. There are more components that are not directly related to the DS, as mentioned by the programme manager.

> *[…] when you deliver a program, it has a lot of components as well as like… You'll have a build and deployment team, you'll have an infrastructure team, you'll have a quality team, you'll have… So those are all things which is not directly DS.*

Quotation 8 – P10

It is important to mention that non-DS-related effort contributes to a larger part of the total project effort compared to DS-related effort, as explained by the lead solution architect in the following quotation.

> *Again, all the work that we are doing, let's say release one and release two were about 50 and 60 thousand person days of effort. Out of which the estimations based on DSs contributed to about 16/17 thousand person days. So you should also understand that there is a lot of other work components of estimation at Company X, which is not directly linked to DSs.*

Quotation 9 – P10

This participant continued to explain that all these other components are done using a sort of waterfall approach. Since this research focuses on effort estimation using agile artefacts, estimation of those other functions is outside of the scope of this research and won't be discussed further.

5.2 Types of Effort Estimation

During the interviews it became clear that 'effort estimation' had a different meaning for different participants in the study. In the previous section it was mentioned that estimations are important to come up with project cost and schedule at the beginning of the project as well as planning during the project. These different levels and stages of the project all require different types of estimations and approaches to estimation. This section aims to clarify the relation between different types of effort estimation, which will provide the reader with some context for the next sections.

5.2.1 Levels of Estimation

Estimation in this project is divided into two different levels. Before the project start, a ballpark estimation was used to determine project price, cost and scale. The elaborated user stories provide enough information to discuss the high level scope of the project.

> *we first get involved in the scoping session and this is where we will get to know the high level scope for that program or project.*

Quotation 10 – P8

During these sessions, the programme manager sits down with the delivery managers of each stream.

> *… the programme manager, the delivery manager from each stream, like we have different streams, development teams basically and testing teams.*

Quotation 11 – P8

After the scoping session, the delivery managers were able to provide ballpark estimations for their stream and these estimations were used to create the overall ballpark estimation. This estimation had to be approved by the customer before the project could be activated.

Once the project was activated, a sample of DSs was created and it was possible to determine a baseline estimation of the effort needed to execute this project. This baseline estimation was created using baseline estimations from all streams of the project. This estimation was again approved by the customer.

The DS design (both the sample and the stories during project execution) also has to be estimated. As soon as DSs are created from the elaborated user stories, they can be used as input for detailed development and functional testing estimations. These estimation figures are used to keep the baseline estimate updated.

Figure 4 illustrates how these different types of estimation relate to each other, the project phases and the agile artefacts.

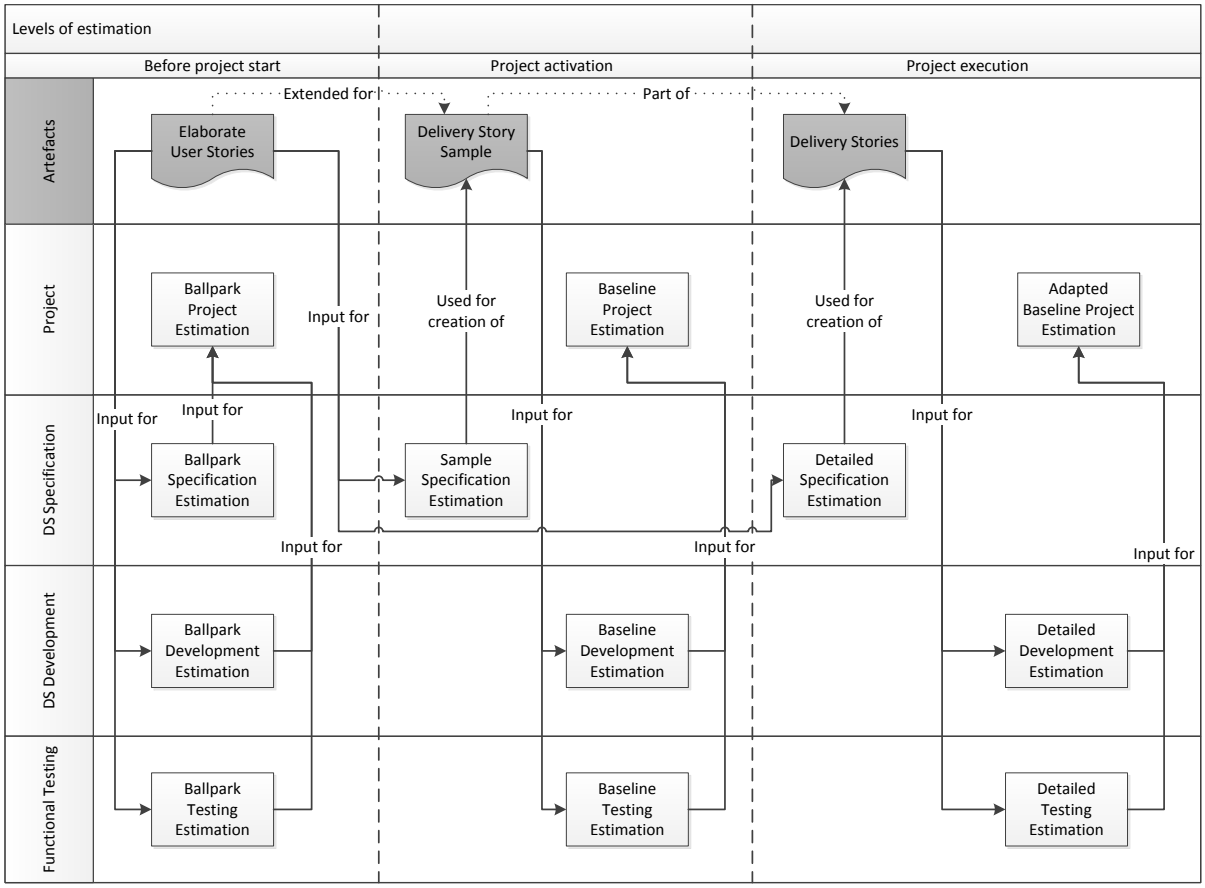

Figure 4: Levels of estimation

It has to be noted that this figure is a very concise representation of the actual situation, focused on the core scrum effort. This figure is only meant to illustrate how the different types of estimation within the actual development are related. For example, the specification estimation is only done when the DSs are going to be created. These stories are not all written at the start of the project. This is an on-going process, where the DSs will be ready in time for the development of a specific collection.

5.2.2 Estimation overview

The previous sections have described all different occurrences of estimation in this project. These occurrences have been modelled to create one overview of estimation, as can be seen in figure 5.

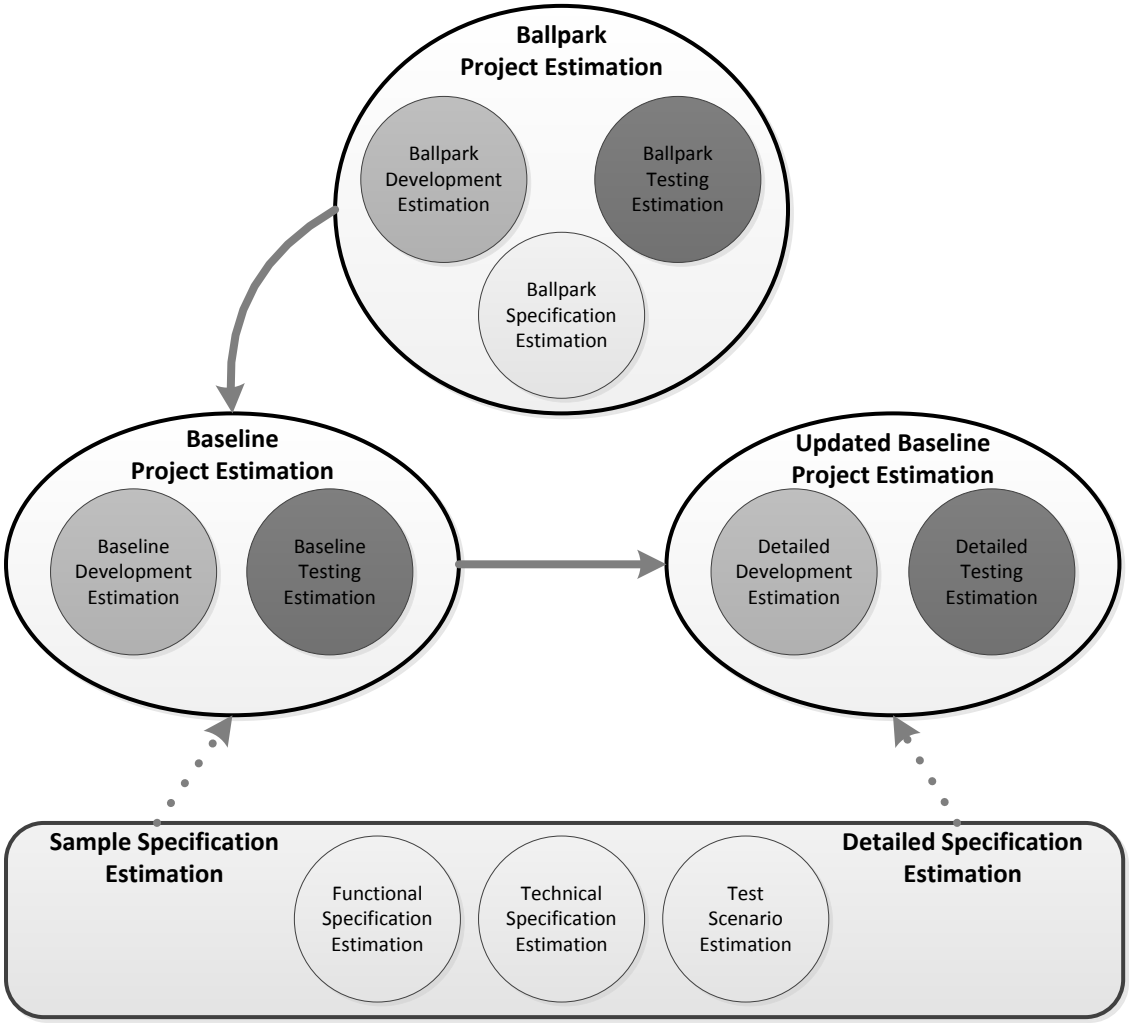

Figure 5: Estimation overview

The big ellipses represent the different stages of project estimation. The ballpark project estimate is based on the ballpark estimations on development, testing and design. The baseline project estimate is based on the baseline estimations from development and testing. The detailed development and testing estimations are continually used to create the updated baseline estimation.

There is a clear link between the project estimations, which is why the arrows are prominently visible in the figure. Similar links (arrows) between the sub-estimates were expected and they might exist, but there was no evidence that the sub-estimates are actually linked. Because of this reason, these arrows are not included in the figure.

The baseline estimations are based on a sample of DSs and extrapolation, while the detailed estimations are based on the complete set of DSs. Every time DSs are created the estimation process is the same. DS specification estimation is always based on separate estimations for the design of functional specifications, technical specifications and test scenarios.

5.2.3 Evolving estimations

The explanation of the different levels of estimation shows that there are many different types of estimation in this project. The development and the testing estimates specifically use the agile artefacts for their estimations and these estimations are then used for the project estimation.

These estimations evolve from a ballpark estimation to a baseline estimation and are finalized in detailed estimations. In all these cases, the delivery manager is responsible for the ballpark estimation, the baseline estimation is done by a team lead and the detailed estimations are done by the people who will be working on development and testing, as shown in figure 6.

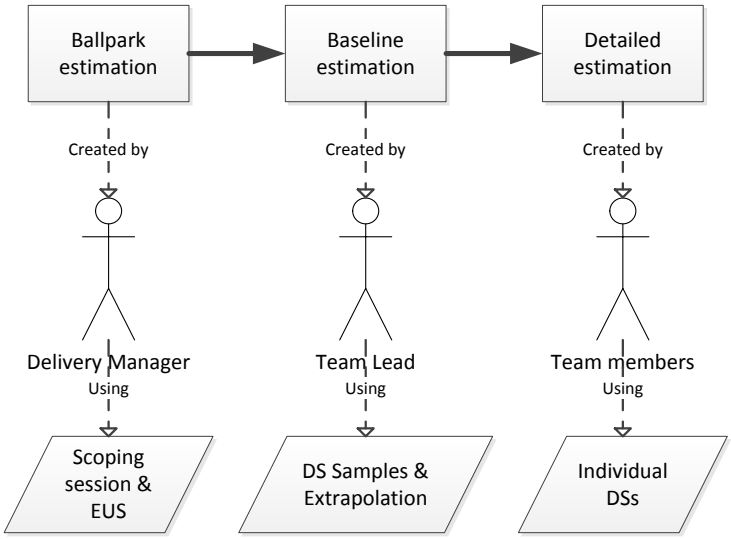

Figure 6: Evolving estimations

The estimation that is done for the DS specification is different from figure 6, because this process of estimation did not change once the project moved on to a new phase. Specification estimation repeats itself every time new requirements are available.

The next sections will elaborate on the different types of estimation and how these fit into the big picture that was presented in figure 4.

5.3 Project Estimation

The highest level of estimation is project estimation, which is the sum of core development effort and all peripheral functions (see section 5.1.3). This section will only elaborate on project estimation related to the core development functions in this project.

5.3.1 Project approval

During the initial estimation for the project, the vendor received user stories as a sample of the requirements. Based on that sample, the vendor understood the complexities of the business functionalities. That sample was discussed in a scoping session and afterwards all delivery managers estimated and extrapolated the sample to arrive at a ballpark estimate to give a project price, cost and scale.

Once the ballpark estimation was approved by the client, the user stories in the sample were used to create DSs. This sample of DSs was the basis to do the next set of estimation samples. A pilot scrum of the sample resulted in estimation guidelines that were to create a baseline estimate "*to come up with a concrete plan in terms of project cost and schedule*" [P7].

The baseline estimation was also shared with the customer, after which the project could move on to project execution. During the execution phase, as soon as more DSs were created and taken up for development and testing, the baseline estimate was updated continually using detailed estimates of the development and testing effort.

The lead solution architect will use the scrum effort as one of the inputs to consolidate all estimates for the project. These estimates will be shared with the customer to keep them informed about planning.

> *Ok, I consolidate all the estimates for Company X. So this scrum effort is one of the pieces that I consolidate along with all the others.*

Quotation 12 – P6: Lead Solution Architect

The solution architect receives all the estimates and will make sure that the estimates are not outrageous. If he identifies items that are totally out of line, the solution architect will go back to the designer to ask for an explanation and (if necessary) change the estimate.

5.3.2 Estimation at project level

Based on the information from the interviews it is possible to model effort estimation at project level. The model in figure 7 shows the process of project estimation as described in this section. This is focused on the core scrum development effort, including the actors that are involved in this process.

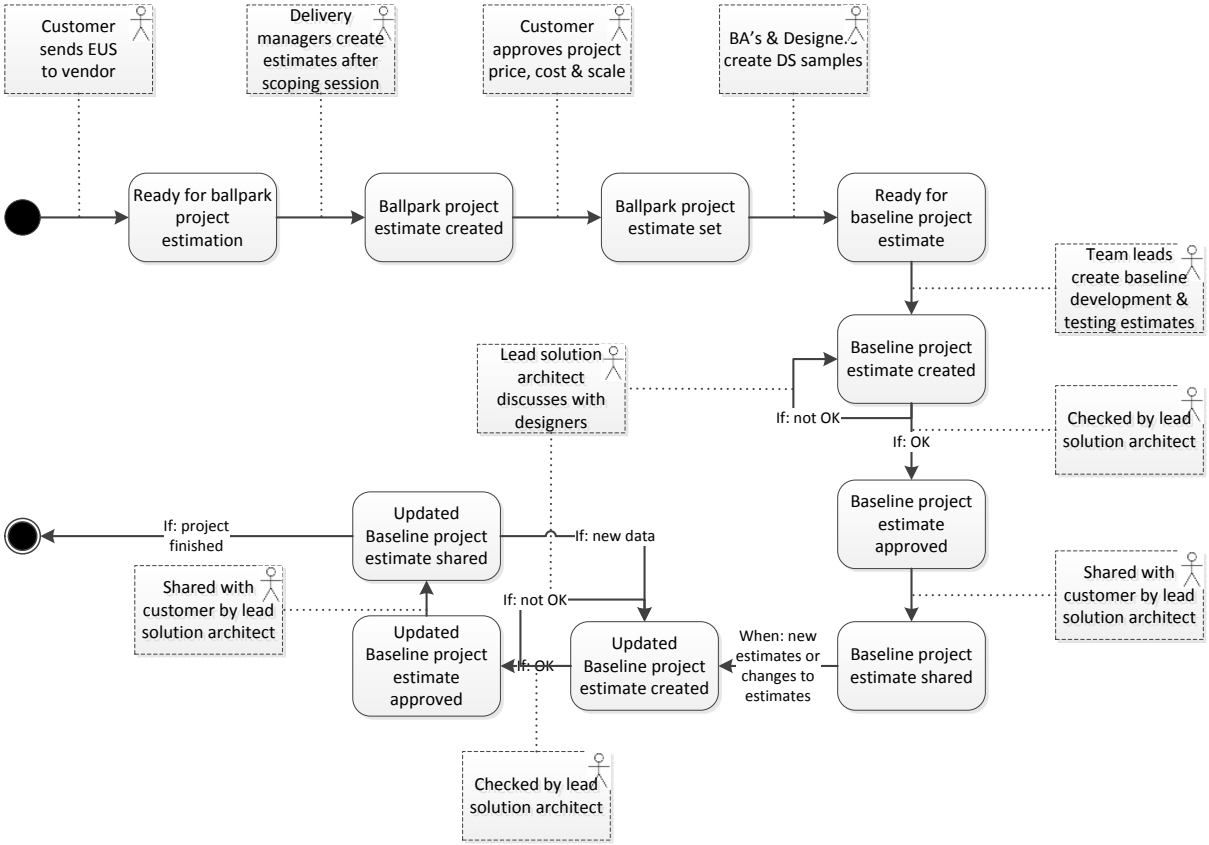

Figure 7: Estimation at project level

During the start-up of the project and the creation of the DSs, the contract was based on Time & Material. Based on the created DSs and a more precise baseline estimate, the contract changed to fixed price. In the course of the project execution the baseline estimate is constantly updated if necessary, these changes will be used for planning and they will not influence the project price.

5.4 DS Specification Estimation

At the beginning of this chapter it was explained how agile artefacts were created in this project, from Elaborate User Stories (EUSs) to Delivery Stories (DSs). These DSs are used as design document for development, but they are also useful for the estimation of the project effort (see previous section) and the estimation of development and testing effort (see next sections). It was mentioned that it takes a significant effort to create the DS artefact, more than was anticipated at the start of the project. (Inaccuracies in effort estimation will be discussed in section 5.8.)

> *…the effort we estimated for the delivery stories itself we underestimated actually. But this has nothing to do with the effort estimation for the development.*

Quotation 13 – P10

Several participants specifically talked about estimating the effort needed for their part in the creation of the DS, which has nothing to do with the effort estimation for development. This section elaborates on DS specification estimation.

5.4.1 Delivery Story Specification

Based on the elaborate description of the creation of agile artefacts in section 5.1, it is possible to create a concise view of the different steps in the specification of a DS. Figure 5 is a simplified version to illustrate all stages of DS specification, including the actors that are part of this design. To keep this figure accessible, the model does not contain all information and approval loops that are included in figure 8.

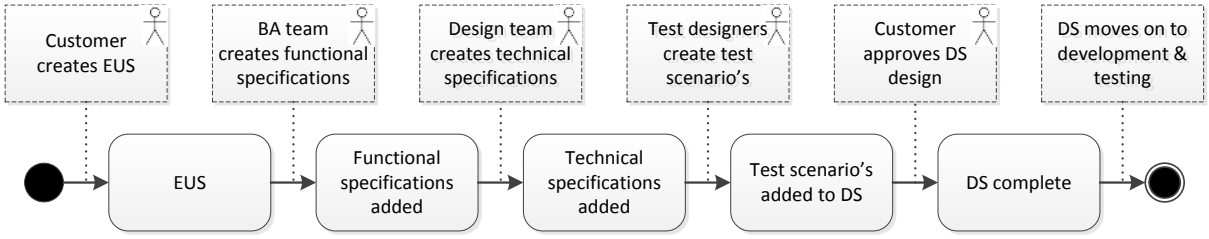

Figure 8: DS specification

In this simplified model it is clear that there are 3 stages in the design where something is added to the DS. The total effort to create a DS can therefore be divided into 3 separate estimations that can be added up to one DS specification estimate.

5.4.2 Functional specification estimation

In order for the BA team to understand the requirements, they will discuss the EUS with the customer. The BA team can use this knowledge to categorize the EUS, to determine the complexity of the story. The EUS contains all the business rules that are used for this categorization into simple, medium and complex.

> *So let's say there is a delivery story ABC, it is divided into simple, medium and complex and this basically drives the… this is not… it does not drive the effort that you put, but it helps you understand that… how big or… it's just the requirement for that particular… let's say the ABC story is simple, so then we'll see… we'll have a very simple requirement… let's say: this is the feel and this should have the ability to enter this… So we know that the effort required is 1 pd [person day]*

Quotation 14 – P1

The number of business rules in the EUS is used as a guideline to size the difficulty of a story.

> So it is based on the number of BR's, Business Rules. […]If it is 3 or 4 business rules then it is simple, till 10 we used to mention it is medium and above 10 it is large. Simple, medium, complex.

Quotation 15 – P4

The business analysts (BAs) are the first to work on the DS. The BA team has to estimate the effort needed to write the functional specifications. If some requirements are similar to ones that have already been estimated, the BA team uses the experience from earlier releases to help them estimate the effort needed.

> *So henceforth, if we have similar requirements we will put in this much effort. We just know that by doing this repetitively, for this will be required this much effort so that's how it is.*

Quotation 16 – P1

If the requirements are about a new functionality, the functional specification estimate is set during an estimation meeting where at least a BA lead and a lead designer will sit down and discuss the effort needed.

> *So I sit with my business analyst lead and the lead designer and we discuss, ok, this is a new functionality that is expected.*

Quotation 17 – P1

Such a meeting will take approximately 15-30 minutes per story.

> *In estimation meetings: the design lead and the corresponding designer is required and the BA lead and the corresponding BA. Sometimes we will have a discussion with the SME [subject matter expert] directly for some critical requirement. The SME from the client side will provide the EUS, the elaborated user stories they will provide. So for some critical requirements we will check with them.*

Quotation 18 – P4

In case of new functionality, the estimation is done by business analysts and designers to ensure a complete understanding of the requirements. The designers will have all the technical knowledge, which will be combined with the functional knowledge of the BAs. The BAs are in-between the SME and the designers, so they will have a better understanding of the business than the designers.

> *See, I mean, since the business analyst will have a functional approach on the things and the designer will have the perspective on the architecture… and from the development… the person who will be actually developing the scrum. So they interact with both of them, they understand that. So they would know how much effort technically is involved.*

Quotation 19 – P1

When the functional specification estimate is finalized, it is consolidated and written down in a master sheet that is kept by the project delivery manager. This is an on-going thing, as soon as a requirement is estimated, it is recorded and then sent for approval.

> *… even if there is 1 person day effort it has to go through the approval process, I mean… from our side we say this, but then a formal mail is sent by the delivery manager and it has to be accepted there. And then the new requirements come to us.*

Quotation 20 – P1

Once the estimates for the functional specification design are approved, the BA team will continue to write the corresponding sections of the DS. When the functional specification is completed, the draft document will be sent to the design lead for the technical specification estimation and to the team that will do the test-estimation. A graphical representation of the functional specification estimation can be found in figure 9.

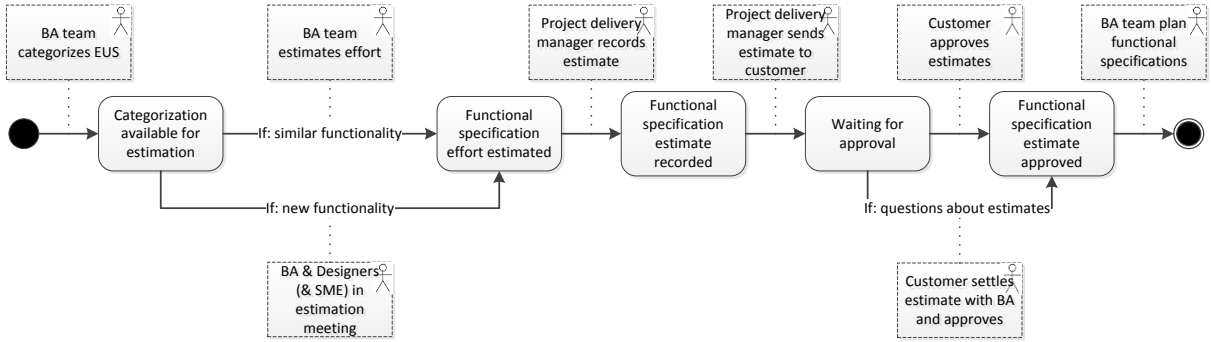

Figure 9: Functional specification estimation

5.4.3 Technical specification estimation

At the time the design lead receives the draft document, she will check who is working on a particular design and forward the story to that designer, who will do the corresponding effort estimation.

> *we are estimating how long I would take to design it, and we will be giving it to our lead. We will be telling them how long it would take for us to proceed with that design [of the DS artefact]… how much time it would take to provide it accurately and all that. Only that we will be telling the lead.*

Quotation 21 – P3

In order to determine the time the designers will need, they use the functionality as a guide.

> *The main thing would be the functionality, at least the part if it is existing or not […] If it is a complete new thing, then we would be taking more time… so depending on that factor we would be telling them actually.*

Quotation 22 – P3

The designers will also use existing development from project as inputs to their estimates.

> *when we are doing it for the next fund… that time we will check the existing development. It might be different from whatever we have written, so then we will refer only to the development what we have done. So it is based on delivery stories (and) what is developed previously.*

Quotation 23 – P4

If the designers do not understand the requirements completely, they can request an estimation meeting with the BA team.

> *mostly we will not have estimation meetings for all the delivery stories, only the stories where it is… the requirement is very tough, we will have a meeting… half an hour. […] Only 2 times we had estimation meetings… that time we will not fully discuss about how much pd we need… they will explain us the requirement… to understand the requirements only we will have meetings. And after, based on the understanding, we will provide the pd impact.*

Quotation 24 – P4

The designers can also check with the developer, to make sure they understand how it works.

> *sometimes they [the designers] will check with the developer for something how does it work, how to implement it… and if it is a new requirement they will come to me [the design lead] and we will have a discussion and all.*

Quotation 25 – P4

Once the designers are done with their estimation, they will send it back to their lead. The design lead will cross-verify the estimates.

> *I will compare it with the EUS, whatever we have received, and their comments, and based on that I will consolidate the original pd [person day] impact.*

Quotation 26 – P4

When the design lead has a consolidated person-day impact, she will send it to the resource manager. The resource manager is the manager for the design phase and she will sometimes review the estimates, to check if they are not outrageous.

> *So if it is exceeding, then we'll sit and clarify what… I mean, once again we will check where we can reduce or where we can increase… that kind of meeting only we have with the manager.*

Quotation 27 – P4

Finally, the resource manager will send the estimate to the customer for approval. The technical specification will only be written after the effort estimation has been approved.

> *… we will send that effort estimation to the business analyst and they will forward it to the SME, corresponding SME's. so based on their input they will deliver the document to us and only after that we will start working on it.*

Quotation 28 – P4

Figure 10 illustrates the estimation of the effort needed for the technical specification.

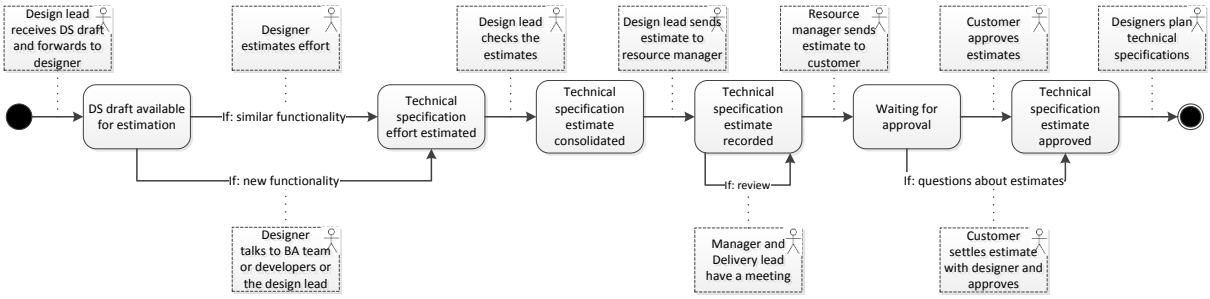

Figure 10: Technical specification estimation

5.4.4 Test scenario estimation

As soon as the technical specification is approved by the customer, the resource manager will contact the test scenario team and send them the stories to get a person-day impact.

> *So… the same lead, they will forward it to them [the testing team] and they will provide [the testing-estimate] separately.*

Quotation 29 – P4

The testing team will create testing-scenarios based on the requirements in the document.

> *So we will analyse what exactly and where exactly in the screen or at what level is that happening and then we'll just make the scenario's for that. So once we get the documents from the designer we will be putting our part in that and we will deliver it. And it is like… we have to deliver it within some period of time.*

Quotation 30 – P2

The testing-team lead is responsible for creating an estimate for this part of the DS artefact. Once the effort has been estimated, the lead will send the estimates to the resource manager.

> *… after the development effort [of creating the delivery story specification]… directly the resource manager will contact the customer*

Quotation 31 – P4[4]

This is the last time the customer will approve any estimations related to the design of the DS artefact. Once this estimate has been approved, the test scenarios are created and the DS is completed. The creation of the test-scenario estimate can be seen in figure 11.

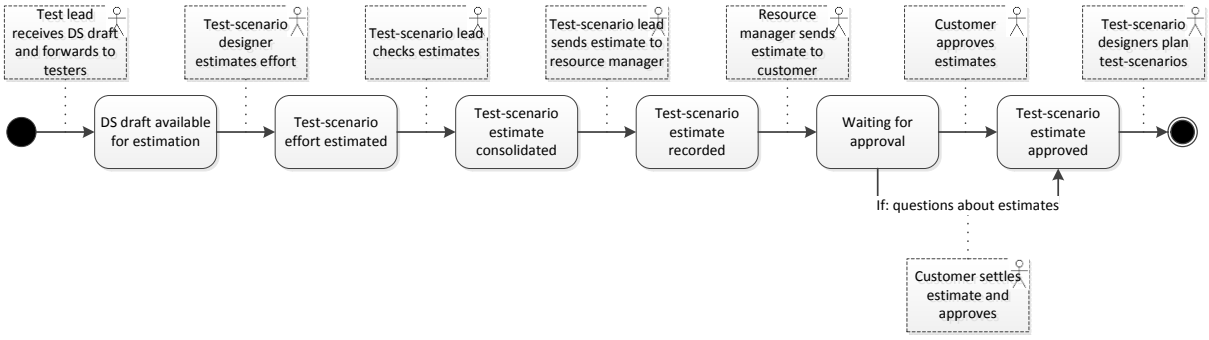

Figure 11: Test-scenario effort estimation

[4] Please note that this participant was the design lead. She is talking about the effort to 'develop' the delivery story specification, which is further referred to in this thesis as creating/designing the delivery story to prevent confusion in terminology.

5.4.5 Change requests

Every estimate, in each of the 3 stages of design, has to go through an approval process and the customer has the possibility to ask questions about the estimates and request changes if required. The customer can disagree with an estimate and in that case there will be a discussion to settle on an estimate that works.

> *we [the BA team] get delivery of requirements on an on-going basis… so, let's say, from the customer side there is decided a ceiling, that only this much effort would be allowed… and, as and when we start approaching that, the customer becomes more and more stern, understandably, on setting the pd's that we give. So they discuss with us: "Can you please tell me why this much effort is required, because there was a story X that we discussed last month and that had only half the effort. The same thing I'm asking you for over here, why are you asking for double the effort?" And then we had to discuss and then we have to initially agree, and then only it goes.*

Quotation 32 – P1

Once an estimate is approved, the design of the DS moves to the next stage and the estimation information can be used for planning purposes.

5.4.6 Estimation at specification level

Based on the information from the interviews it was possible to model effort estimation at each of the three DS specification phases. The model in figure 12 shows how each of these estimation processes fit into the creation of the DS artefact as shown in the model in section 5.4.1. Approval-loops have been omitted to improve readability of this model.

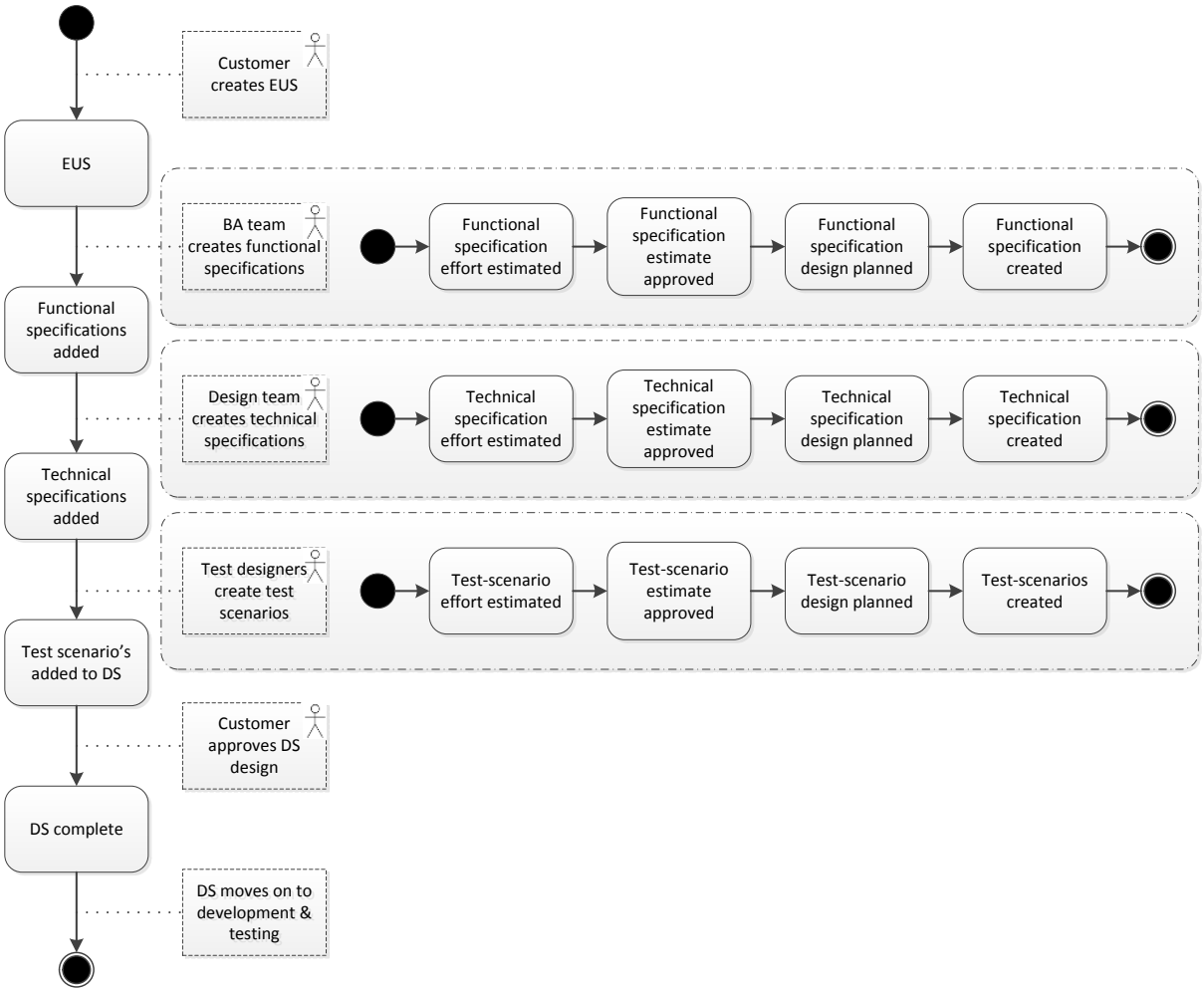

Figure 12: Estimation at design level

5.5 DS Development Estimation

Once a DS has been approved by the customer, the design is completed and can be taken up by the development team. Each individual DS will be estimated in order to determine the effort and create a planning for development.

> *Now, at this point of time we do individual estimates. […] We take every DS and estimate it.*

Quotation 33 – P6

To determine which DSs should be estimated (and developed) first, it was important to distribute scope and create a sequence. At the beginning of the project, before starting the development for the DSs, there was a planning phase. During this phase, the requirements were analysed.

> *There we did a lot of analysis of the requirements, we identified dependencies, we identified the sequence, which domain can run parallel and what has to go in sequence. So the DS development is focused on that sequence, […]*

Quotation 34 – P7

The sequence was used to determine which stories to take up for development. Decisions about scope were taken in consultation with the customer.

> *So that inputs will be taken jointly by client and the vendor company, so if you are saying this is the scope…the scope distribution in different buckets will be done with the help of clients so their input should be required for that.*

Quotation 35 – P10

The scope distribution will help to determine which stories are more important to develop first.

5.5.1 Baseline development estimation

Once the first sample of stories have been approved, it was possible to create an initial baseline estimation using the DS sample. Initial estimations were mainly being driven by the guidelines, based on the number of business rules.

> *Yes, we do, we have the guidelines as I said, based on the number of business rules, what sort of… is it a new screen, how many tabs are there in that screen, how many store procedures I need to create for that, is it a batch functionality, that kind of guidelines are there. So that's what drives the estimation.*

Quotation 36 – P7

These guidelines are used to categorize the stories into simple, medium and complex and very complex.

> *Typically it's 3 days, 5 days, 7 days for a simple, medium, complex. For a very complex one it is roughly 40pd, 50pd and we have what we call global stories, which cuts across a lot of domains, a lot of stories, […] , to be used in any number of places so we call them global and the estimates for them are very high actually, 60PDs compared to 3PDs.*

Quotation 37 – P7

These complexities are used for the next set of estimation samples.

> *Initially we just extrapolated based on a sample… this is the complexity for a sample, 30% simple, 40% complex, that kind of complexity.*

Quotation 38 – P7

These estimates help to firm up the estimations for development.

> *So the second set is where the estimation gets firmed up, it's more a baseline estimate rather than a ballpark estimate. The first one which we did during the requirements, before we even started the project, is a ballpark. The second one which we do here as part of the Ds is more a baseline estimate.*

Quotation 39 – P7

Based on these numbers, the baseline development estimates are discussed with the customer.

> *At the time that we write the design document, we'll show this design document and say that it can take this much person days.*

Quotation 40 – P5

The product owner works together with a business analyst to prioritize the stories and pick stories for development.

> *During prioritizing, I [as business analyst] was there and the product owner was there.*

Quotation 41 – P0

Once prioritizing is completed, the business analyst takes the stories to the tech lead for development.

5.5.2 Detailed development estimation

When the DSs have been sent to the tech lead, he will start a sprint by estimating all DSs.

> *Suppose the sprint is of two weeks, at the beginning of the sprint we will take up all these items that were assigned to us and we will estimate it.*

Quotation 42 – P5

The initial baseline estimates will give an idea about the high level estimate, which will be analysed further during an estimation meeting using the actual DS artefact and the experience of the developers.

> *So there [during estimation for development] they use the component details, complexity, guidelines, experience so they are able to estimate.*

Quotation 43 – P7

Before the estimation meeting, the tech lead will have a discussion with the BA to talk about the functional domain and determine which set of developers will work on it.

> *so I just assign them, this [DS] has come for this domain and you [the developer] know this domain better than anyone so you can sit and review it.*

Quotation 44 – P5

In an estimation meeting, the tech lead sits down with the developers. The tech lead will try to negotiate with them, to make sure that everything will be delivered in a good time, but also with proper code. A business analyst is also present to share the knowledge about the functionality.

> *… with help of the business analyst (BA) we will just see… this design is here for us, this functionality is here for us and this part of the DS will/can take this much of person days. We'll just set the estimate first as a high level thing and then based on that we will divide work items. Like these are the items based on one DS, based on the complexity and the design structure, we'll divide it into the work items and then we'll again analyse it so that we can give a correct estimation for the project.*

Quotation 45 – P5

The scrum master will be present at the meetings to guide the estimation process. He is the one to decide which methods are used for effort estimation, but at this point there are no agile approaches (like planning poker) used to guide estimation.

> *[planning poker] is what we used for a couple of sprints in collection 1, but then we didn't continue or follow that method. […] It didn't work out well and it was taking a lot of time for the team to do that estimation… […] so we had to stop that. Our approach now is: we do the estimate, this is whatever goes into it, whatever it takes to complete, so the team has to work extra hours to finish, so that's the approach taken.*

Quotation 46 – P7

The BA is not part of the actual estimations, but only there to explain the DS and any answer questions the developers might have about the functionality.

> *I [business analyst] was present there to explain the functionality to the development team…*

Quotation 47 – P0

In most cases the DS contains all the elements that are needed to create an estimate, but if the developers have some doubts about the story there is a possibility to talk to the architects.

> *If here are some scenario's, some functionality, some architecture we have some doubts about, we will just quickly get back to the architecture and we'll talk with them and tell "this is the design we have got, is this as per agreed? Is this reviewed by you, can you explain it further? Do we need to do any other things in that?".*

Quotation 48 – P5

When the complete story is estimated, the tech lead will write down the estimate in the product backlog, which functions as a master sheet. Each tech lead is assigned to a scrum master and the estimate will go to that scrum master for a review of the estimation.

> *When we are done with our estimation we will pass this on to the scrum master. Basically the scrum master will be there also at these meetings, but finally he will review with a person who will be negotiating and who will say "ok, this is that thing".*

Quotation 49 – P5

The scrum master was already present at the estimation meetings, but he is also the one that will take the estimate to the customer. Based on the estimations the customer sees the delivery dates for these particular development activities.

> *we will get back to the customer as well… we'll say "This is a breakup of this DS", so he [the customer] will be looking at it and he will approve it and negotiate as well.*

Quotation 50 – P5

As soon as the estimate has been approved by the client and the scrum master, the stories will be frozen and the team can go ahead with development.

> *They would freeze the user stories before they would go into the scrum, no changes were accepted once it went into the scrum […]*

Quotation 51 – P5

5.5.3 Review on different levels

There are several moments in the development process that the estimates are reviewed. The initial estimates are discussed with the customer to get the approval to continue with the next step. After the estimation meeting and a review by the scrum master, the scrum master will review the estimate with the client to reach an agreement.

> *We review to agree with the client and the vendor company and after that some review of the progress against the estimation…*

Quotation 52 – P10

When the estimates have been approved by the customer, the development can start and the scrum master will report it to the scope manager, who moves the estimates up to all levels of the higher management.

> *The delivery manager will write it up and the scope manager will review… then the program manager will verify, then...our <…> management also… so it will have at least 4 or 5 reviews…*

Quotation 53 – P10

The vendor company uses a tool to keep track of the project progress and the estimate is also recorded in this tool by the project management.

> *a whole tool is there, in which we will define our cost, budget , as well as efforts, estimates and all those things… and the progress has to be measured across that. Any deviations , slippage positive or negative is going to impact our ROI.*

Quotation 54 – P10

Figure 13 presents an overview of actors that are involved in reviewing the development estimate. All actors with grey heads review the development estimate at some point, although the estimate is already set in a discussion between the scrum master and the customer.

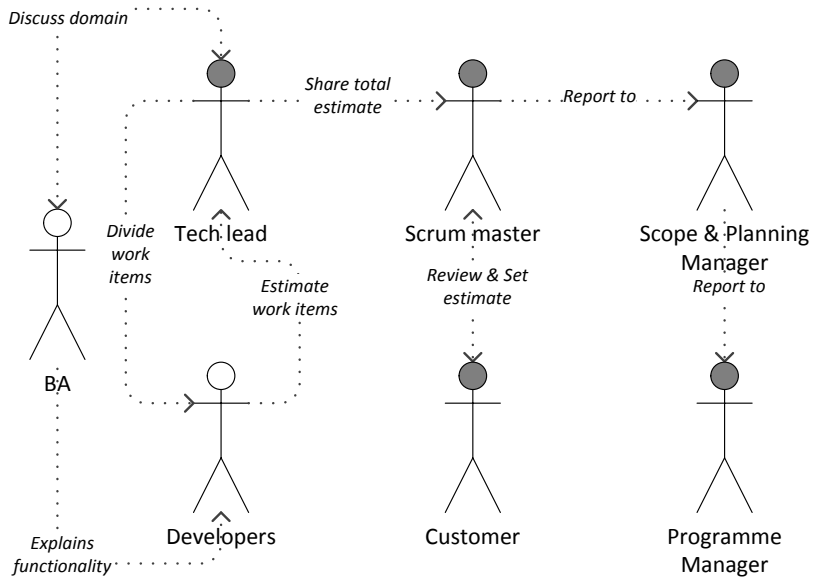

Figure 13: Reviewing the estimate

5.5.4 Estimation at development level

Based on the information that was gathered during the interviews, it was possible to model the process of estimation at the start of development. Figure 14 shows the different stages of estimation at development level. Customer-approval-loops have been omitted to improve readability of this model.

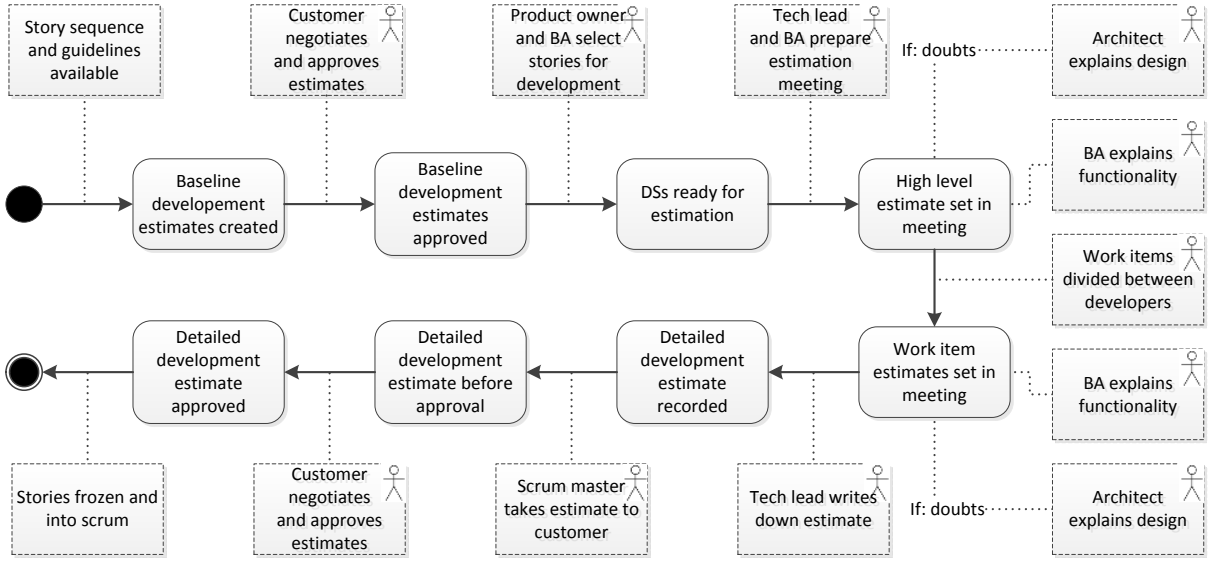

Figure 14: Estimation at development level

5.6 Functional Testing Estimation

One of the characteristics of an agile approach to software development involves the completion of deployable, tested software at the end of a cycle. This means that functional testing is an essential part of the development process and this also includes estimating the effort needed to write test cases to support project planning.

At the beginning of the project, the test delivery manager will create a ballpark estimate.

> *so for ballpark kind of estimate we go with the industry standard, a thumb rule, like I said… 30 to 40%.*

Quotation 55 – P8

This is 30 to 40% of the development effort as estimated at a high level.

5.6.1 Baseline testing estimation

After the test delivery manager has done the first overall estimation based on the scoping session, there will be no further estimation meetings. The next level of estimation is done by test leads. This is a high level solution estimate that is used as a baseline testing estimate.

> *After analysing the DSs, we convert that into a… number of test cases. We have sort of a baseline complexity, like we take 30% complex area, another 40% in medium and 30% in simple stories. And we have, with the past experience, we have developed one delivery story of complex is about converted into 7 to 10 test cases, that kind of thing. We use that as a unit when we derive the estimation for… when we estimate the number of test cases per delivery story… and then add up a total number of DSs given in the scope and multiply that into a number of test cases per DS.*

Quotation 56 – P8

This estimation is done separately, without any estimation meetings. This is possible because there are estimation templates available where the estimates are recorded.

> *Actually, we have estimation templates, we do that. There is no meeting as such.*

Quotation 57 – P8

Once the test lead is done with the baseline estimate, the test manager will validate this estimate.

> *Yes, review… if the test lead does that, the test manager will validate that. But no other estimate review..*

Quotation 58 – P8

5.6.2 Detailed functional testing estimation

A short time before the test design activities, a detailed estimate has to be created for planning purposes.

> *It is done within the team and as part of our update we will update the management team with the total amount of test cases that we will need.*

Quotation 59 – P8

Each team member will work on the estimates. They will read all the documents and try to understand the process.

> *So once we understand that, we come to know that all the input documents of details and we will go ourselves and read that completely and try to understand in that process… in that process we will individually interact with the key people, like SMEs or delivery story designers and that… then we will get the clarification and do our estimation.*

Quotation 60 – P8

Next to the subject matter experts and the designers, the team members of the testing team can also post queries for the business analyst as a means of clarification.

The detailed testing estimates are recorded by the test manager or the test lead in estimation templates. These estimates are purely meant for planning purposes.

> *we do estimation and then provide it to the customer. Because this is a fixed price project basically, so the customer… that level of estimation… the customer is not really bothered. It is for our planning purpose that we do it.*

Quotation 61 – P8

5.6.3 Estimation at testing level

Based on the information that was gathered during the interviews, it was possible to model the process of estimation for functional testing. Figure 15 shows the different stages of estimation at testing level, from baseline estimation to detailed estimation.

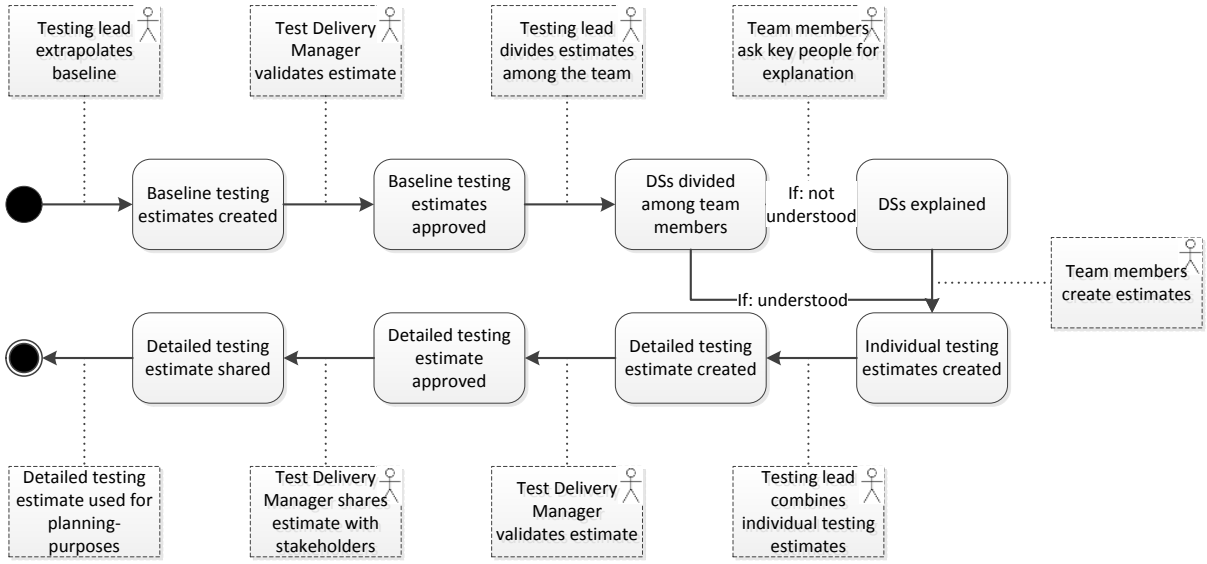

Figure 15: Estimation at testing level

5.7 Risk management

Participants deemed risk management as an important part of the conversation on EE. In this project risk has been mitigated by the combining estimation approaches to get the best possible estimation. Other issues that have been addressed involve dealing with change, dependencies between stories and uncertainties in estimation.

5.7.1 Estimation approaches

A combination of estimation approaches can help to improve estimates and prevent risk. Participants all mentioned different approaches to estimation, depending on the available information and the preferences of the people involved. However, there are some recurring approaches to project estimation. One participant gave a good overview and described three different aspects of estimation that were reflected in all other interviews:

1. Experience and requirements
2. Components
3. Resource loading

Experience and requirements are used for a first estimation. Some people described their gut-feel as an important part of this type of estimation, other participants mentioned the use of estimation guidelines. These guidelines are created once the DS sample is completed.

The number of components of a design is another way to break down estimation. If the build is broken down into components (DSs and even the sections in these stories) it is possible to put numbers to the size of the build.

The third aspect of estimation is a resource loading, assigning resources to activities. Comparing the estimated effort with the available effort until a certain deadline will help to revise estimates if necessary.

All three aspects of estimation are not required to achieve a good estimate, but combining any of these approaches may well lead to better estimates. Participant 6 described why he used all of these views to end up with his project estimate and planning.

> *There is none of them I feel are fool proof, anything can go wrong, but if you have 2 or 3 different ways to look at it you can get a picture that you are comfortable with submitting to the customer. Again, it's all estimation, so there is no right or wrong there, it's the best you can do. But you follow different views based on the estimates you have.*

Quotation 62 – P6

5.7.2 Change management

Many change requests can have a big impact on the project duration.

> *so 1 to 2 months of overall slippage will happen… I'll not say slippage, it's additional CR's [change requests], additional requirements to cope with…*

Quotation 63 – P10

The managing of change, requirement creeps, scope creeps and requirement gaps resulted in some overshooting during the first release.

> *Release 1 was slightly over-shooted because of this… not able to manage or handle the scope creep and the requirements changes properly. But it was not because that… it wasn't estimated properly. I'll say it would have been done as long as we would have managed our team management… specific to requirements to get the scope right properly.*

Quotation 64 – P10

Therefore a good change management process is important as a means of risk mitigation.

> *We had come up with a process, if there was a change with a total effort of more than 4 hours, it had to go through a formal change management process. Otherwise we used to accommodate that change within the sprint.*

Quotation 65 – P0

Any change to a user story would be considered as a spike.

> *Any change to a user story, we used to consider that as a spike, as an add-on. The product owner would not change the user story completely, whatever part, if there was a change he would add it as a spike and that would again go back to the product backlog and again re-prioritized and then based on that we would take it up in the next sprint.*

Quotation 66 – P0

The stories were frozen before they would go into the scrum, no changes were accepted to a story once it was put into the scrum.

5.7.3 Dealing with dependencies

Dependencies between stories can be difficult to deal with, but they are important to keep in mind. The first dependencies were identified during the planning phase before the project started, the DSs are prioritized based on this information.

> *There we did a lot of analysis of the requirements, we identified dependencies, we identified the sequence, which domain can run parallel and what has to go in sequence. So the DS development is focused on that sequence […]*

Quotation 67 – P7

Despite this sequence, it is still possible that people have to revisit stories that have been finished in the past because of dependencies. These dependencies are captured in the functional specifications in the DS artefact.

> *We have a section, it calls of what are all the dependencies. Even the story mentions all the dependencies which get captured in the functional spec […] That's how they understand whether there is an implication in terms of unit testing, because if it is touching a core component, it has to be tested with all others. So that plays a key role in the estimation.*

Quotation 68 – P7

The business analyst will also keep the developers informed about dependencies.

> *Dependencies are taken care of by the BA because he knows all the dependencies between functionalities, so he is the person who will tell us that some part of the functionality is dependent on a requirement that is already there or that has to come later on.*

Quotation 69 – P5

Dependencies are already covered as part of the complexity of the DSs.

> *a definition of a complexity could be OK. If… say, a DS is actually touching 5 additional DSs, then it has to be complex. There are several parameters and one parameter could be this, so if you are naming these ones A, B, C, D and E, then you can say, ok, this DS is a complex one.*

Quotation 70 – P10

But for detailed estimations, the person who is estimating should be aware of the dependencies, to understand the complexity and put that into the estimation process. If the dependencies are not clear, the designer or BA will be asked for clarification.

> *If there is no clarity, we will sit with the individual designer or BA team and try to clarify that and put that into the estimation.*

Quotation 71 – P8

5.7.4 Dealing with uncertainties

It is always possible that there is some level of uncertainty related to levels of knowledge or experience that prevents people to understand the complete picture. One way to deal with uncertainty is to leverage the knowledge of more senior team members and external experts for help.

> *I have people who have much more experience than I have, I go to them.*

Quotation 72 – P1

But, if the uncertainty is not related to the experience of the participant, it is also common practice to talk to people with more knowledge about a topic. These people can be the business analysts, the DS designers, developers, testers and of course also the subject matter expert, the customer.

> *We ask for inputs from different designers, and the key people who know that domain who have worked or are currently working in that domain. They give more of an impact analysis, that's another way of firming up on the requirements, so we take some more time to understand more of that change or component or development*

Quotation 73 – P7

If there is uncertainty about the effort that is needed, it is common practice to build in a buffer. This is the case with the design of DSs, but also with testing and development. Effort estimation for the DS specification is designed to prevent inaccuracies. Such inflation of an estimate is also used to incorporate any unexpected scenario's that are included in realistic estimation.

> *Suppose I feel like… for a requirement I need to give 2 pd's but I'm not very sure… at that time we'll give 3 pd's… it works like that*

Quotation 74 – P4 – Design Lead

In case there is un-clarity and it is difficult to estimate, there are some standards that will address contingencies.

> *we will also put some contingency to address that… it depends on some previous experience and some standards, 10 to 15% kind of contingency.*

Quotation 75 – P8 – Test Delivery Manager

> *We have a contingency, a two week period, where if there is any kind of estimation slippage… they would be taking care of it. I know that this is a two week period after the third sprint, that is a two week consolidation period…*

Quotation 76 – P9 – Delivery Assurance Manager

The two week consolidation period acts as a buffer, which represents 30% of each collection. This means that the EE process already has a legitimately built-in distortion of 30%.

5.8 Inaccuracies

Despite different approaches to estimation and other ways to reduce risk , it is still possible that inaccuracies in estimation occur. Changing teams could be a reason for deviations between estimated effort and actual effort, because of differences in the skills of employees. However, it was pointed out that in this project they always try to maintain the same teams until a domain or process is finished.

> *People move on, but we plan, we always keep a pool of resources who can work along with the team, at least a month in advance, get the knowledge, so… It's probably… 6 months in advance, we try to prevent this.*

Quotation 77 – P7

Despite the estimation guidelines, experience of the estimators and measures like maintaining the team composition, there are still some cases of overestimation and underestimation.

5.8.1 Overestimation

Overestimation does not happen very frequently. If a story is overestimated, it has to do with very big or complicated stories.

> *Overestimation I doubt we had any examples actually… Maybe one or a few instances, but that's more on the global stories. Global stories… we didn't have any visibilities on them, so we assumed an average of 60PD kind of estimate and… it took more, a bigger timeframe actually.*

Quotation 78 – P7

A good visibility of the required effort can be caused by a lack of documentation, but the opposite is also possible. The lead solution architect talked about a case where a very complicated legal document was attached to the story. When people are unsure about big stories, they tend to overestimate to be on the safe side.

> *one example of a 150 person days because… there was a change request they had inserted an attachment which actually contains a legal document. […] it's a 60/70 page document attached to the US… […] They got scared of that document, maybe about 10 pages is where the actual stuff is. […] So those were the cases where we have seen 100+ PDs [person days] in the estimations because people were unsure of what that thing [the legal document] is.*

Quotation 79 – P6

5.8.2 Underestimation

Underestimation happens more often and often has to do with changes during the project.

> *under-estimation… in the previous project… for various reasons the timelines were changed but we continued with the same assumptions on that and then at the end we sort of… we ended up in additional effort. That is happening here [in the current release] also now.*

Quotation 80 – P8

Next to sudden changes, inaccuracies can also arise when a senior developer creates an estimate based on his own experience while the actual development takes more time for a junior developer.

> *Underestimation… yes… that happened for sure. Because… like I mentioned to you, sudden changes in requirements and then… senior developer is actually estimating it and then a junior developer is actually developing it…*

Quotation 81 – P9

Change management is very important, because changes during the lifecycle that were not included in an updated estimate resulted in additional effort.

> *main reasons for the change in estimation of over or wrong estimations is… baseline of the input are not happening, which means… the requirement for example… […] we take at that point in time some baseline, but during the course of further progress in the lifecycle it keeps changing. [….] we do the initial estimation and overcome that, but at the end or halfway through our testing, it still gets changed and we don't refactor. And another is… unknown items, like dependencies in some cases is not listed on the requirement dependencies… […] That resulted in additional effort and kind of that, this is one of the examples*

Quotation 82 – P8

Participant 8 mentioned that it would be easier to estimate if the input would be complete and fixed before the actual estimations have to be done.

One very clear case of underestimation in this project arose in the DS specification phase.

> *the effort we estimated for the delivery stories itself we underestimated actually. […] That was heavily underestimated actually, initially. And then we had to correct it for release 2. An example for underestimation is delivery story estimation, we almost got it… we had to double the team size within the first 2 collections to catch up with all the backlogs, and even it went beyond schedule and in a sense we had to stretch 3, 4 months extra beyond the actual completion or planned completion date.*

Quotation 83 – P7

Extra effort was required for stories that turned out to be more difficult to design than expected.

> *… for example a case on which I was working where we stumped on the design part. The design part… on the changes that we were expecting that were happening in design… it was much more. […] it says that it has to be developed for let's say 2pd's of effort, but in the end it was about 10 pd's of effort at least.*

Quotation 84 – P1

This extra design effort would not be necessary if domain experts are available at all times, but sometimes they were not available. The business analyst in interview 1 explained that he had to go around talking to different stakeholders to gather the data himself, which took more time than estimated up front.

5.8.3 Dealing with inaccuracies

There are several approaches to deal with inaccurate estimations. One of those approaches involves increasing the team size to catch up with the planning.

> *we had to double the team size within the first 2 collections to catch up with all the backlogs, and even it went beyond schedule and in a sense we had to stretch 3, 4 months extra beyond the actual completion or planned completion date.*

Quotation 85 – P7

During all levels of estimations there are many reviews, which is one of the reasons that inaccurate estimates are caught early.

> *there are cases [of inaccurate estimates], but when they really are out of control it tends to be caught early, because… nowadays even more, there are a lot of people who pay attention to the estimates.*

Quotation 86 – P6

It turned out that estimates are only reviewed during the project. The actual and the estimated effort are not compared once the stories are completed.

> *Now I know that the estimations which are captured in the product backlog and the actual estimations… may not actually match. The reason being… one is there are a lot of requirements they miss. Second is… we have not been able to actually compare the actual and the estimated effort… […] We have a lot of practical challenges in doing that, especially because it is an agile project. […] we are trying to come up with a separate… an alternative means of capturing that. But that is again not a primary focus activity, because we are working in a fixed-price model*

Quotation 87 – P9

Improving estimation is not a primary focus activity because of the fixed-price model of this project.

The high level estimations are roughly the same as the average of the detailed estimations, which is also one of the reasons why it is not a high priority to improve effort estimation.

> *What we have seen is, our high level estimations gives [an average of] about 6PDs, with all the detail, the estimations that we have done for every US individually… if you would take the average, it would still be that same number that we guessed in the beginning.*

Quotation 88 – P6

The lead solution architect explained in interview 6 that deviations in estimation are balanced. Underestimation happens more often but doesn't have a big impact. Overestimation can have a huge impact, but it doesn't happen very often. That is why occurrences of underestimation and overestimation will even each other out and it explains why estimation is not an urgent problem in this project.

> *we don't care [about underestimation]. It gets balanced. The average effort is probably 5 to 6 person days for a DS. So how much can it be under[estimated] really? […] Whereas overestimation can be infinitely over 5PD. […] But in general, these two gets balanced. So even if things get overestimated and underestimated, in the end you are roughly ok.*

Quotation 89 – P6

5.9 Delivery Story Artefact

During the first release, around 1100 DSs were created. In case the requirements changed, the DS would also be updated.

> *If the US is going to change then the DS will also have to be updated accordingly and approved.*

Quotation 90 – P10

The DS artefact evolved during the project, improvements based on experiences and changes based on changing needs. At this stage it is complete and contains all the information that is needed for development.

> *the design document is covering all the aspects and components that have to be taken care for, so no, we have not faced that scenario. It [the DS] has covered everything in that [document].*

Quotation 91 – P5

The only thing that could be a reason to make changes to the DS template is to improve testing.

> *if we can bring testing hardly from level 1 to up… it may help I think… better estimate my testing efforts.*

Quotation 92 – P10

To prevent any gaps between stories, a recent change was the creation of a solution framework.

> *[the solution framework] actually helps us to close the gaps between the stories otherwise something will fall between those DSs, it may be an incorrect requirement or an incorrect design, but taking it together, that is a change that we made for release 2.*

Quotation 93 – P6

The DS artefact has the big benefit that it gives a good idea about what the project is going to look like.

> *[The delivery story] gives a visibility at an earlier stage*

Quotation 94 – P7

Because of this visibility, the DS is useful for detailed effort estimation, but it was mentioned that the DSs were not specifically created with the purpose of estimating effort. The details in the DS are a way to capture requirements, simplify development and can also be used for effort estimation.

> *So I just want to make sure that all we understand the same about DSs. It's not a way to estimate. We just take that opportunity, because we are doing the detailed analysis and design we understand a little bit more than what is in the US. Take that opportunity to do the estimation also.*

Quotation 95 – P6

Even though participant 6 doesn't see the DS artefact as a specific way to estimate, one of the business analysts did value the opportunity to do estimation using the DS.

> *Estimation would not be the same without it [the DS].*

Quotation 96 – P1

6 Discussion

This section discusses the findings that were presented in the chapter. Overall observations from this research are structured and discussed in several sub-sections.

6.1 Applying Agile

The objective of this case study was to research effort estimation practices in large agile projects. The project that was selected for this case study was used as an agile project in previous research, but during the course of this research it became clear that this project cannot be labelled as a complete agile project. One of the interview participants mentioned that the project should be viewed as a hybrid project, but the project tends to be even more traditional than the term 'hybrid' would suggest. The software development in this project does meet a few values of the agile manifesto [7], as depicted in table 6.

Table 6: Applying agile

Agile values according to manifesto	Software development in this project
Individuals and interactions over processes and tools	The larger scale of this project caused the level of documentation to increase and the interactions between teams to decrease, which does not meet the agile principle. This research focused on effort estimation practices and it turned out that the processes and interactions to estimate effort are separated from each other and that there is not one estimation technique that was used by every team. This is in line with earlier research by Peixoto et al. [51], which concluded that even teams in the same country, organization and development methodology choose different estimation techniques to estimate effort.
Working software over comprehensive documentation	The short iterations and recurring reviews supported the delivery of working software. However, the delivery story contains a lot of information and individual stories can still represent comprehensive documentation.
Customer collaboration over contract negotiation	The customer was available for questions during development and reviewed deliverables at the end of each 2-month collection. This customer involvement ensures developing something that the customer wants and needs.
Responding to change over following a plan	Because of the short iterations and recurring reviews, the software development was able to deliver working software and respond to change.

Research by Ramasubbu and Balan [67] has indicated that early stage project estimation remains a significant challenge in globally distributed software projects, with missing information as one of the root causes. However, in this case study this is not a problem. The vendor uses "*learning-oriented estimation* [67]", there are effort estimation guidelines in place to help managers with the effort estimation process. Because a lot of information is available, this vendor company has improved estimation practices and at the same time avoided one of the most common problems in globally distributed projects.

In this project only the most useful parts of agile methodology (for this project) were applied, but in some aspects this selective use of agile resulted in a move from agile/hybrid towards traditional software development. One of the characteristics of agile is the use of estimation meetings [56, 59], where the actual developers are involved in estimating the effort they think they will need for development. This is a useful practice because it puts a certain level of responsibility at the developers, but it can also take time. Meetings are still used to divide tasks and discuss requirements with development teams, but the high-level project estimation is done by high-level managers. These managers can take inputs from their leads and the team members, but in the end one can question how accurately high-level managers can estimate the actual effort that the practitioners would need. It has to be noted that the company does not strive for precision; buffers were added to deal with estimation inaccuracies.

6.2 Estimation Inaccuracies

Due to the nature of this research and the availability of project data, the number of inaccuracies in estimation turned out non-quantifiable. However, in the interviews it has become clear that the estimated effort can sometimes deviate from the actual effort that was needed.

It became clear in the results that effort estimation within this project is not one estimation, but it consist of many sub-estimations that are used for individual planning and added up for project planning. These sub-estimations of different streams are not linked and the participants view these estimates as separate entities. Sharing knowledge about estimation practices between different streams could be used to standardize and improve the effort estimation processes in this company. This is supported by research by Børte et al. [21], who argue that social interaction plays a crucial role in software effort estimation, because individual knowledge becomes activated during social interaction.

Any ideas about the causes for inaccuracies in estimated effort versus actual effort are purely based on the views and ideas of individual interview participants. It turned out that mainly managers are aware of cases of over-estimation or under-estimation, while the designers, developers and testers are not (made) aware of their own inaccurate estimates. Sharing knowledge and giving feedback about the causes for inaccurate estimates may lead to a better understanding of inaccurate estimates and improve future estimates. More feedback and change is one way to deal with difficult effort estimation [71].

6.3 Estimation Reviews

Estimation reviews are a recurring topic in this research. There are two different types of estimation reviews that can be identified in this project (see the state models of estimation in chapter 5). Managers review the estimates and with regular intervals customers approve estimates. In this case, approval from the customer is also used as some kind of expectation management. Another type of review is a comparison between estimates effort and actual effort, to identify deviations and possible causes for these deviations.

If deviations in estimations occur in this project, they are managed, for example by increasing team size to meet a deadline. Large divergences from the schedule are therefore avoided. However, a certain level of analysis after a deviation has occurred can help the understanding about such irregularities and prevent further inaccurate estimates in the future. Only in a few instances

estimates are reviewed after completion of development, and if they are reviewed it is mainly to keep track of the overall project effort and progress. It was not mentioned by any participant that this company reviews the quality of estimates after the actual work is completed, but there was a specific remark that it is very difficult to capture the estimated versus the actual effort. There are a lot of practical challenges in doing that, because the project is agile and there are continuous requirement changes.

Another way this project deals with the challenges of estimation in this agile project is the adoption of buffers (which is a common practice in outsourcing agreements). This project uses the consolidation period of a collection as a buffer of 33% of the development effort, which is consistent with a study that mentions an average of 30-40% of extra effort on top of the original project estimates [60]. The combination of the buffer and the general findings of managers that over and under-estimation even each other out, results in an accurate estimation average. Despite this overall success when it comes to estimation, one participant mentioned that an analysis of effort estimations, updated effort estimations and the actual effort could be useful to improve future effort estimation. The identification of cases of estimates that were far from the actual effort could be used to prevent future inaccuracies and improve the overall estimation accuracy.

7 Validity

The validity of each of the 7 stages of the research approach has been discussed in chapter 4. In any interview research, there are several principal issues related to validation, concerning objectivity and reliability of the results [48]. There are four common validity concerns to case studies that can be used to test the validity: construct validity, internal validity, external validity and reliability [82]. This section will discuss the validity of this research and finish with some limitations to this research.

7.1 Construct validity

Construct validity is about establishing correct operational measures for the concepts being studied. There are two steps that have to be covered to meet the test of construct validity according to Yin [82]:

1. Select the specific types of changes that are to be studied (and relate them to the original objectives of the study) and
2. Demonstrate that the selected measures of these changes do indeed reflect the specific types of change that have been selected.

This research aims to identify current effort estimation practices within agile projects. An interview study has been designed to identify these practices. This method of data collection was selected because this would guarantee data collection about the actual practices as they were used in the project. It could be argued that this is not a valid measure, given that there is no guarantee that the selected participants will cover the whole story.

To prevent threats to construct validity, participants with diverse roles in the project were selected for this interview study. This created the opportunity to analyse the phenomenon of effort estimation from different perspectives, which helped to create a complete picture. All participants were interviewed using the same interview protocol. For a number of questions, we witnessed practitioners giving similar answers, which made us think that the validity of the results is strong.

7.2 Internal validity

There is an inherent weakness related to the use of interviews to gather data for research. The correctness of this research is guaranteed because every interview was recorded, transcribed and double checked for correctness. The structured questionnaire that was created up front helped to ensure consistency between the interviews. The structure of the interviews also made it possible to compare answers and combine answers to piece together one coherent picture. The internal validity of this research was also improved because coding was done by three researchers, to prevent errors and biases in this study. The researchers worked independently of each other and in two different locations, exchanging and discussing their results over email to arrive at a common understanding of the concepts. However, the completeness of this research cannot be guaranteed due to some limitations to this research (see section 7.5).

7.3 External validity

External validity deals with the problem of knowing whether a study's findings are generalizable beyond the immediate case study [82]. There are different frameworks available to justify the generalizability in information systems research [50, 72].

The data that was gathered during this research can be generalized to observations about effort estimation practices [50]. It is possible to do an analytic generalization of results from this single study [72] if there are similarities between relevant attributes of this case study project and other projects. The conclusions that have been derived from this research are likely to happen in similar projects in the same company, but also in similar other companies. Similar projects refer to agile projects in a large, distributed and outsourced environment on a fixed-price contract base. Similar companies refer to multinational IT offshore outsourcers with a high CMM level.

7.4 Reliability

The objective of reliability is to be sure that if a later investigator followed the same procedure as described in this thesis, the later investigator should arrive at the same findings and conclusions [82]. The elaborate research approach as described in section 4 is a documentation of the procedures that were followed and are a way to ensure reliability of this case study.

7.5 Limitations

It is not possible to guarantee completeness of the research results. There are several limitations that might have influenced the data collection.

There were limitations to the time that was available for the interviews. All 10 interviews were executed on 2 consecutive afternoons. There was no time available for a short analysis of interview results immediately after every interview, which made it impossible to use a grounded theory approach to possibly adapt the interviews by including emerging questions.

There were also some limitations regarding the availability of participants. It was not possible to talk to all important leads that might have different insights on the topic of effort estimation. Participants were therefore also selected based on their availability during the 2 days that the researchers were on-site. It was not possible to arrange more interviews from a different location because of constraints on communication possibilities.

The manager that arranged the interviews with the practitioners expressed the wish to sit in on the interviews to observe and learn. This might have influenced the interviewees, but it is not possible to determine how. Since both interviewers were relatively junior compared to some of the managers, it was useful that a senior manager helped to decrease the influence of hierarchy differences. However, with the younger participants, the interviewers did feel that the presence of the manager might have influenced the answers.

8 Conclusions

At the beginning of this research project, a central research question was formulated: How does effort estimation currently happen in agile projects that use Delivery Stories? This research question was decomposed in the following sub-questions, which have been answered using a literature study and an elaborate interview study. This chapter concludes with a few words on the contribution of this research.

(RQ1) *What is known in published literature about the challenges in requirements-based effort estimation in agile, distributed, and global projects?*

The overview of the related work in chapter 3 discusses all aspects of large, distributed and agile software development projects in detail. There are challenges related to agile projects, scaling those projects and dealing with distribution of such projects. Other challenges that have been mentioned are related to agile RE in large-scale environments and effort estimation in distributed and agile projects.

(RQ2) *How does the vendor organize agile projects and who gets involved in the effort estimation process as part of this?*

The vendor has organized their agile project based on a planning phase, which included a pilot scrum. The experiences from this planning phase helped to plan the different releases. Releases consist of several collections and a collection consists of three sprints and a consolidation period. Another remarkable aspect of the application of agile in this project is the use of the DS artefact. The DS is a design document which incorporates all components necessary for development and all participants in this study are satisfied with the current contents of the DS. One of the advantages is that communication between the developers and the customer is greatly reduced, because the designing of the document is already based on interaction with the client. When the designers of the DS get their queries answered, the developers don't need to ask those questions any more. Therefore, it is easy for the developers to take this document and continue with development.

(RQ3) *How are delivery stories used, at what point of time, and by whom in support of the EE process?*

After a thorough analysis of the results, it was possible to answer this question. It turned out that effort estimation is a recurring task that appears in different levels of the project. Effort estimation takes place on project level, DS specification level, DS development level and functional testing level. Estimations usually go through 3 phases, starting with a ballpark estimation done by managers. The ballpark estimation evolves into a baseline estimation and the baseline is updated once more detailed estimations are available. Detailed estimations will be created by the people that are actually designing the DS or developing based on that DS.

(RQ4) *What problems are caused by inaccuracies in effort estimation in agile projects, if any? If no problems occur, what does the team do to prevent problems from happening?*

To answer this question, the first step was to identify inaccuracies in effort estimation. However, it has proven to be difficult to determine concrete causes for inaccurate estimates. Such analysis could help to improve effort estimation in future projects. Even though the vendor company is mature and has all the data that is needed to analyse any diversions between estimated effort versus actual effort, such analysis has not been executed. On the other hand, it is not a top-priority for the vendor, because data analysis is not necessary if there is no urgent need for improvements.

Effort estimation seems to work like it is happening right now. All different occurrences of effort estimation in different levels of the project could have added up to a big deviation in total project effort, but this is not the case. This project deals with the challenges of estimation by adopting the use of buffers. Next to the buffers, many review- and approval-cycles are combined with a vast amount of software development experience. This leads to fairly good results when it comes to effort estimation. Therefore, it can be concluded that the vendor is not engaging in improving effort estimation because this project does not urgently need better effort estimation practices.

From a theoretical perspective, this research makes a contribution to the emerging research on effort estimation in agile, distributed, large-scale projects. In large, agile projects, DSs support effort estimation practices and reduce the need for communication. From a practical perspective, this research offers practitioners knowledge about the use of agile artefacts in effort estimation and the value of learning-oriented estimation.

9 Reflections

While working on the last finishing touches of this master thesis, it was time to write down some reflections. There are a lot of things to reflect upon, but I'm not writing a book, so let's keep it close to this thesis.

For me, this research reflects cultural awareness. It was a process that had already been put in motion during my Erasmus exchange in Stockholm. During my first month I already experienced a melting pot of cultures, people from around the world worked together in an intensive language course to learn about Swedish culture and even more about each other's culture.

My cultural awareness was enhanced by a course on "Global IT Management" that I selected last minute after another course had been cancelled. Cultural differences were one of the main topics of that course and I never realized how useful that knowledge would be until Maya told me about a master thesis research that would take place in India.

Despite the fact that I didn't know that much about India at that time, I felt privileged that she told me about this master thesis research and even more so when I learned that I was selected for this amazing opportunity to experience a different culture.

No matter how much I prepared myself, by reading books and watching (travel) documentaries about India, I experienced a huge culture-shock when I touched down in Mumbai. It was my first time outside of Europe and the colours, the traffic, the slums, the noise, the smells… everything was different from everything I ever experienced before. However, after the initial shock, it turned out that it was easy to adjust to a new lifestyle and a new culture. Which does not take away the fact that every day, from the day of my arrival until my departure, I encountered things that surprised me. Luckily, when I didn't understand something, there was always somebody willing to explain.

I definitely have to mention that cultural and language differences have influenced both the researcher and this research. The Dutch approach to tasks and questions is usually a very direct one, which means that even a toned –down version of this approach is much more direct than Indians will ever be. Language also proved to be prone for misunderstandings, but extra explanations would usually resolve these misunderstandings as soon as they appeared. My conclusion is that even if everybody that is involved is culturally aware, it is still difficult to take all the cultural influences away. I could go on and on about this topic, but I believe that the cultural influences in this research could be the topic for another 6 months of research.

I feel rich and truly blessed that I had the opportunity to finish my university studies in such a special and spectacular way. These last few months have been the most educational period of my entire studies and I will appreciate and value these experiences for the rest of my life.

9.1 Acknowledgements

This thesis would not be complete without explicating my gratitude to everybody that was involved in this master thesis research.

I would like to thank the Tata Research Development & Design Centre for offering this research opportunity. Specifically, I would like to thank Smita Ghaisas, my supervisor at TRDDC, for her help, support and especially her patience. She did everything and more to help me feel at home at the office and in Pune. I also want to thank Preethu Rose, whose work effort and input helped to shape this research. Next to her input in this research, I'm very grateful that she helped me to understand more about Indian culture. I would also like to thank the interview participants for their time and input. Furthermore, I want to express my gratitude to all other kind people I've met at TRDDC, for their support when I needed it and a pleasant working environment in general.

I also want to thank Klaas Sikkel and Maya Daneva, my supervisors from Twente University, for their help, advice and never-ending support. Their feedback and insights have helped me to continually improve my thesis.

Finally, I want to thank my family, for supporting me when I decided I wanted to go to India, during my stay and especially for their support and inspiration during the last few months, when it wasn't always easy to stay focused on the goal of graduation.

References

[1] *About TCS*. Tata Consultancy Services, City, 2012.
[2] *Innovation*. Tata Consultancy Services, City, 2012.
[3] Abdelnour-Nocera, J. and Sharp, H. *Adopting Agile in a Large Organisation - Agile Processes in Software Engineering and Extreme Programming*. Springer Berlin Heidelberg, City, 2008.
[4] Abdullah, N. N. B., Honiden, S., Sharp, H., Nuseibeh, B. and Notkin, D. *Communication patterns of agile requirements engineering*. City, 2011.
[5] Abrahamsson, P., Fronza, I., Moser, R., Vlasenko, J. and Pedrycz, W. Predicting Development Effort from User Stories. In *Proceedings of the Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement* (2011). IEEE Computer Society, [insert City of Publication],[insert 2011 of Publication].
[6] Adolph, S., Hall, W. and Kruchten, P. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, 16, 4 2011), 487-513.
[7] AgileAlliance *Manifesto for Agile Software Development*. City, 2001.
[8] AgileAlliance *Principles behind the Agile Manifesto*. City, 2001.
[9] AgileAlliance *What is Agile Software Development*. City, 2012.
[10] Agrawal, M. and Chari, K. Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. *Software Engineering, IEEE Transactions on*, 33, 3 2007), 145-156.
[11] Anderson, D. J. *Agile Management for Software Engineering*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 2004.
[12] Auvinen, J., Back, R., Heidenberg, J., Hirkman, P. and Milovanov, L. *Software process improvement with agile practices in a large telecom company*. City, 2006.
[13] Avritzer, A., Bronsard, F. and Matos, G. *Improving Global Development Using Agile: How Agile Processes Can Improve Productivity in Large Distributed Projects*. Springer Berlin Heidelberg, City, 2010.
[14] Barlow, J. B., Giboney, J. S., Keith, M. J., Wilson, D. W., Schuetzler, R. M., Lowry, P. B. and Vance, A. Overview and guidance on agile development in large organizations. *Communications of the Association for Information Systems*, 29, 1 2011), 25-44.
[15] Benefield, G. *Rolling out Agile in a large enterprise*. City, 2008.
[16] Bjarnason, E., Wnuk, K. and Regnell, B. *A case study on benefits and side-effects of agile practices in large-scale requirements engineering*. City, 2011.
[17] Bjarnason, E., Wnuk, K. and Regnell, B. *Overscoping: Reasons and consequences - A case study on decision making in software product management*. City, 2010.
[18] Bjarnason, E., Wnuk, K. and Regnell, B. *Requirements are slipping through the gaps - A case study on causes & effects of communication gaps in large-scale software development*. City, 2011.
[19] Boehm, B. Get Ready for Agile Methods, with Care. *IEEE Computer* 35 2002), 64–69.
[20] Boehm, B. and Turner, R. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, 22, 5 2005), 30-39.
[21] Børte, K., Ludvigsen, S. R. and Mørch, A. I. The role of social interaction in software effort estimation: Unpacking the "magic step" between reasoning and decision-making. *Inf. Softw. Technol.*, 54, 9 2012), 985-996.
[22] Bosch, J. and Bosch-Sijtsema, P. *Coordination Between Global Agile Teams: From Process to Architecture*. Springer Berlin Heidelberg, City, 2010.
[23] Bose, I. Lessons learned from distributed agile software projects: A case-based analysis. *Communications of the Association for Information Systems*, 232008), 619-632.
[24] Cao, L. and Ramesh, B. Agile requirements engineering practices: An empirical study. *IEEE Software*, 25, 1 2008), 60-67.
[25] Cohn, M. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.
[26] Daft, R. L. and Lengel, R. J. Organizational information requirements, media richness and structural design. *Management Science*, 321986), 554-571.

[27] De Lucia, A. and Qusef, A. Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, 2, 3 2010), 212-220.
[28] Dibbern, J., Goles, T., Hirschheim, R. and Jayatilaka, B. Information systems outsourcing: A survey and analysis of the literature. *Database for Advances in Information Systems*, 35, 4 2004), 6-102.
[29] Dierckx de Casterlé, B., Gastmans, C., Bryon, E. and Denier, Y. QUAGOL: A guide for qualitative data analysis. *International journal of nursing studies*, 49, 3 2012), 360-371.
[30] Dybå, T. and Dingsøyr, T. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50, 9-10 2008), 833-859.
[31] Dybå, T. and Dingsøyr, T. What do we know about agile software development? *IEEE Software*, 26, 5 2009), 6-9.
[32] Elshamy, A. and Elssamadisy, A. *Applying agile to large projects: New agile software development practices for large projects*. City, 2007.
[33] Flyvbjerg, B. *Case Study*. Sage, City, 2011.
[34] Gat, I. *How BMC is scaling agile development*. City, 2006.
[35] Grimstad, S., Jørgensen, M. and Moløkken-Østvold, K. Software effort estimation terminology: The tower of Babel. *Information and Software Technology*, 48, 4 2006), 302-310.
[36] Hull, E., Jackson, K. and Dick, J. *Requirements Engineering*. Springer, 2010.
[37] Ilan Oshri, Julia Kotlarsky and Willcocks, L. P. Managing dispersed expertise in IT offshore outsourcing, lessons from Tata consultancy services *MIS quarterly executive*, 6, 2 2007), 53-65.
[38] Jørgensen, M. How to Avoid Selecting Bids Based on Overoptimistic Cost Estimates. *IEEE Softw.*, 26, 3 2009), 79-84.
[39] Jørgensen, M. Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates. *Information and Software Technology*, 52, 5 2010), 506-516.
[40] Jørgensen, M. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70, 1–2 2004), 37-60.
[41] Jørgensen, M. and Grimstad, S. Over-Optimism in Software Development Projects: "The Winner's Curse". In *Proceedings of the Proceedings of the 15th International Conference on Electronics, Communications and Computers* (2005). IEEE Computer Society, [insert City of Publication],[insert 2005 of Publication].
[42] Jørgensen, M., Teigen, K. H. and Moløkken, K. Better sure than safe? Over-confidence in judgement based software development effort prediction intervals. *J. Syst. Softw.*, 70, 1-2 2004), 79-93.
[43] Keats, D. M. *Interviewing: A Practical Guide for Students and Professionals*. UNSW Press, 1999.
[44] Koehnemann, H. and Coats, M. *Experiences applying agile practices to large systems*. City, 2009.
[45] Korkala, M., Pikkarainen, M. and Conboy, K. *Combining Agile and Traditional Customer Communication in Distributed Environment*. Springer Berlin Heidelberg, City, 2010.
[46] Kotonya, G. and Sommerville, I. *Requirements engineering: processes and techniques*. J. Wiley, 1998.
[47] Ktata, O. and Lévesque, G. *Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects*. City, 2009.
[48] Kvale, S. and Brinkmann, S. *InterViews: Learning the Craft of Qualitative Research Interviewing*. Sage Publications, 2008.
[49] Lacey, A. and Luff, D. *Qualitative Data Analysis*. The NIHR RDS for the East Midlands / Yorkshire and the Humber, 2009.
[50] Lee, A. S. and Baskerville, R. L. Generalizing Generalizability in Information Systems Research. *Info. Sys. Research*, 14, 3 2003), 221-243.
[51] Lima Peixoto, C. E., Audy, J. L. N. and Prikladnicki, R. *Effort estimation in global software development projects: Preliminary results from a survey*. City, 2010.
[52] Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kähkönen, T. Agile Software Development in Large Organizations. *IEEE Computer*, 372004), 26–33.

[53] M. Daneva, E. v. d. V., C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, R. Wieringa Agile Requirements Prioritization in Large-Scale Outsourced System Projects: an Empirical Study. *Journal of Systems and Software.*2012).
[54] Magazinius, A. and Feldt, R. Exploring the human and organizational aspects of software cost estimation. In *Proceedings of the Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering* (Cape Town, South Africa, 2010). ACM, [insert City of Publication],[insert 2010 of Publication].
[55] Magazinovic, A., Pernstål, J. and Öhman, P. *Software cost estimation inhibitors - A case study*. City, 2008.
[56] Mahnič, V. and Hovelja, T. On using planning poker for estimating user stories. *Journal of Systems and Software*, 85, 9 2012), 2086-2095.
[57] Malik, A. A. and Boehm, B. Quantifying requirements elaboration to improve early software cost estimation. *Inf. Sci.*, 181, 13 2011), 2747-2760.
[58] Merriam-Webster *Case Study*. City, 2012.
[59] Moløkken-Østvold, K. and Jørgensen, M. Group Processes in Software Effort Estimation. *Empirical Softw. Engg.*, 9, 4 2004), 315-334.
[60] Moløkken-Østvold, K. and Jørgensen, M. A Review of Surveys on Software Effort Estimation. In *Proceedings of the Proceedings of the 2003 International Symposium on Empirical Software Engineering* (2003). IEEE Computer Society, [insert City of Publication],[insert 2003 of Publication].
[61] Muhairat, M., Aldaajeh, S. and Al-Qutaish, R. E. The impact of global software development factors on effort estimation methods. *European Journal of Scientific Research*, 46, 2 2010), 221-232.
[62] Nagar, C. and Dixit, A. Software Efforts and Cost Estimation with a Systematic Approach. *Journal of Emerging Trends in Computing and Information Sciences*, 2, 7 2011), 312-316.
[63] Nawrocki, J. R., Jasinski, M., Walter, B. and Wojciechowski, A. Extreme Programming Modified: Embrace Requirements Engineering Practices. In *Proceedings of the Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering* (2002). IEEE Computer Society, [insert City of Publication],[insert 2002 of Publication].
[64] Paetsch, F., Eberlein, A. and Maurer, F. Requirements Engineering and Agile Software Development. In *Proceedings of the Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (2003). IEEE Computer Society, [insert City of Publication],[insert 2003 of Publication].
[65] Pfleeger, S. L. *Software engineering: theory and practice*. Prentice Hall, 2001.
[66] Pfleeger, S. L., Wu, F. and Lewis, R. *Software Cost Estimation and Sizing Methods: Issues, and Guidelines*. Rand Corporation, 2005.
[67] Ramasubbu, N. and Balan, R. K. *Overcoming the challenges in cost estimation for distributed software projects*. City, 2012.
[68] Ramesh, B., Cao, L. and Baskerville, R. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20, 5 2010), 449-480.
[69] Sadun, C. *Scrum and Global Delivery: Pitfalls and Lessons Learned*. Springer Berlin Heidelberg, City, 2010.
[70] Sandberg, J.-E. and Skaar, L. A. *Getting Communication Right: The Difference Between Distributed Bliss or Miss*. Springer Berlin Heidelberg, City, 2010.
[71] Schmietendorf, A., Kunz, M. and Dumke, R. *Effort estimation for Agile Software Development Projects*. City, 2008.
[72] Seddon, P. B. and Scheepers, R. Towards the improved treatment of generalization of knowledge claims in IS research: Drawing general conclusions from samples. *European Journal of Information Systems*, 21, 1 2012), 6-21.
[73] Seidel, J. V. *Qualitative Data Analysis*. Qualis Research, City, 1998.
[74] Šmite, D., Moe, N. B. and Ågerfalk, P. J. *Fundamentals of Agile Distributed Software Development*. Springer Berlin Heidelberg, City, 2010.
[75] Srinivasan, J. *Preparing your Offshore Organization for Agility: Experiences in India*. Springer Berlin Heidelberg, City, 2010.

[76] Stebbins, R. A. *Exploratory Research in the Social Sciences*. SAGE Publications, 2001.
[77] Sulfaro, M., Marchesi, M. and Pinna, S. *Agile practices in a large organization: The experience of Poste Italiane*. City, 2007.
[78] Svejvig, P. and Fladkjær Nielsen, A.-D. *The Dilemma of High Level Planning in Distributed Agile Software Projects: An Action Research Study in a Danish Bank*. Springer Berlin Heidelberg, City, 2010.
[79] Trendowicz, A., Münch, J. and Jeffery, R. State of the practice in software effort estimation: a survey and literature review. In *Proceedings of the Proceedings of the Third IFIP TC 2 Central and East European conference on Software engineering techniques* (Brno, Czech Republic, 2011). Springer-Verlag, [insert City of Publication],[insert 2011 of Publication].
[80] Turk, D., France, R. and Rumpe, B. Limitations of agile software processes2002).
[81] VersionOne *State of Agile Development*. City, 2012.
[82] Yin, R. K. *Case Study Research: Design and Methods*. SAGE Publications, 2003.
[83] Yin, R. K. *Case study research: design and methods*. Sage Publications, 1994.

Appendices

This section contains:

A. Interview Protocol

The outline of the questionnaire was as follows:

1. Briefing: Introduction (of interviewer, research and research goal)
2. Setting up the project context
3. Effort Estimation (EE) Process (Concrete Cases)
4. Effort Estimation (EE) Process (General Observations)
5. Debriefing

The interview protocol intends to help answer 3 out of 4 research questions:

RQ2: How does the vendor organize agile projects?

RQ3: How does effort estimation currently happen in agile projects that use Delivery Stories?

RQ4: What problems are caused by inaccuracies in effort estimation in agile projects?

These research questions were covered by certain sections in the questionnaire as shown in the table below.

	Project Context	Process of EE	Estimation Methods	Inputs in EE	EE Experiences	EE Inaccuracies	General Observations
RQ2	X	X					
RQ3		X	X	X	X		
RQ4					X	X	X

1. Introduction

Project	
Participant Name	
Participant Role	
Years of Experience	
Date of Interview	
Time of Interview	

<*introduce myself, give background, explain purpose of research*>
"First, let me introduce myself; my name is Renske Vermolen, and I'm a Master's student of the University of Twente, the Netherlands. As part of my graduation project I'm conducting research at the Tata Research Development & Design Centre in Pune. There, we are trying to better understand Effort Estimation (EE) in Agile project that use Delivery Stories (DSs). This interview is part of a research project towards this goal."

"The questionnaire consists of some information about this research, followed by some semi-structured sections. The whole interview should take no longer than 60 minutes. Please rest assured that all findings will be made anonymous and no personal or confidential information will be used in any way. If you feel uncomfortable answering any of the questions, you can always choose not to answer. If you have any questions at any time, please feel free to ask."

"I also would like to let you know how much your contribution matters to me. Without input from professionals like you, this research would be impossible to accomplish. We have no way to learn how things work in real life… For this reason, your participation is very important and I'd like to thank you in advance, for the time you are spending right now with me."

"This interview study will focus on the aspect of effort estimation in agile projects that use delivery stories. The goal is to understand how the current effort estimation practices work and what could be possibly causing inaccuracies in effort estimation. A future study will continue on this research to include effort estimation in agile projects that use only user stories. The combination of these two studies will generate valuable insight in effort estimation practices in agile projects."

<Explain r*esearch questions*>
RQ2: How does the vendor organize agile projects?

RQ3: How does effort estimation currently happen in agile projects that use Delivery Stories?

RQ4: What problems are caused by inaccuracies in effort estimation in agile projects?

"Let's discuss this concrete project where you were involved in the process of effort estimation.

Do you have any questions at this stage?"

2. Setting up the Project Context

To be asked only to the first interviewee (in case all interviewees are from the same project). In case the first interviewee doesn't know an answer to all questions, this section can be repeated at a second interview.

1. How long was the project? (with regard to time)

2. How big was the project? (with regard to number of people)
   Probes
   The size of the developing team? Other people that were involved?

3. What was the nature of the distribution of the project?
   Probes
   How many sites were involved?
   What was the distribution of people among these sites?
   How was the work distributed between them?
   How was the communication organized? (email, phone, video conference …)

4. What was the type of contract? (E.g. fixed-price, per hour, etc.?)

5. How many iterations / sprints did you perform?

6. How did the communication with the client take place?
   Probes
   What communication medium? Frequency of communication?

7. Was the delivery story (DS) artefact already present at the start of the project or was it introduced at a later stage in the project?

8. Did the client consider the concept of DSs useful?

9. How many DSs were created during this project?
   a. Were they isolated?
   b. Were they grouped?

10. Were there a lot of changes to DSs during this project?

3. Effort Estimation (EE) Process (Concrete Cases)

Process of EE

"*A software process includes 3 things: people, artefacts (input/outputs) and resources. This section will elaborate on the process of EE (using DSs). Please try to keep concrete cases in mind*"

1. Could you elaborate on the process of using DSs as a unit of EE?
2. What roles are involved in EE?
3. Who (as in: what roles) provide information as input to EE?
    a. Is this information exclusively based on a DS, or is other information used to complement the DS?
    b. Do you also look at other documents (e.g. rough project plans, etc) when you proceed with the EE?

If not yet answered at 2 and 3, ask 4-6:

4. Is EE done in consultation with the customer?
5. At what point in time does the developer get involved in the EE process, if at all?
6. In which way is the Domain Expert involved?
    a. What inputs is s/he providing to the EE process?
    b. How is his/her input useful?
7. Who records the estimate? At which point in time? Where is this information about the estimates recorded? (Spreadsheet/database/...)
8. Who reviews the estimate? At which point in time?
9. What is the information (about estimates and reviews) used for?
    Probes
    Does someone send the information to other people? For example to higher level managers? Or to other teams?

"*Now I have some more specific questions about your experiences in the EE process*"

10. What is your role in estimation meetings?
11. How long are these estimation meetings on average?
    a. Did you ever experience some very long/short estimation meetings?
12. Did you ever use an approach where effort was estimated assuming idealistic conditions versus realistic conditions?
13. If you have hundreds of stories to use in the EE, where do you start from? (Sequence)
    a. Which story would be the first one to be estimated?
    b. Does such sequencing really matter in your view?

Estimation methods

14. Who decides which EE methods and techniques are used?

15. Do you use any tools for estimation? (E.g. planning poker tool, software based tools, etc.)

16. Did you use velocity charts for EE?
    a. If yes, in what terms is velocity estimated? (Story points / days / hours)

17. How do you size functionality in EE? (numeric sizing / t-shirt size / Fibonacci sequence / dog breeds / ...)

18. Could you possibly tell me more about how you/your team proceeds in order to get estimates?
    b. What do you do first, what second?
    c. Which artefacts do you use?
    d. Who do you ask for inputs/reviews?

Inputs in EE

19. Which of all DS artefacts do you take into consideration while estimating the effort? (GUI screen, test cases, priority, business value, business rules and validations)

    Probe

    If the interviewee mentions specific artefacts such as business value etc., ask him/her about what they mean by that artifact.

EE experiences

20. What were the granularities of the DSs that you worked with? (atomic or modularizable) In case it is modularizable,
    a. Did you experience situations where, based on the estimates, a DS was split into two or more DSs?
    b. If yes, do you take inputs from the previous estimation for making estimations for the new DS?

21. Is your EE process aware of any dependencies among the DS? Does any knowledge about the dependencies really matter for the output for EE?

22. When you face uncertainty in EE, how do you work around it?

    Probes

    What do you do? I guess you do not stop your EE process?

23. Have you ever observed that estimation on a DS on which a senior role was involved was more accurate than an estimate where you had novices?
    a. For which roles did you feel seniority counts?
    b. Could you elaborate more on your observation in your most recent project?

    Probes

    Possible senior roles: senior programmer / a senior business analyst / an experienced Scrum Master / a Project Manager / a Tester

EE inaccuracies

24. How often did teams change in your project?
    a. If yes, have you experienced stale estimations in your project?

25. Could you provide an example from your last project where you observed a concrete case of over-estimation? And of under-estimation?
    a. Would you like to share some of the lessons learned from your EE experiences?
    Probes
    "Could you elaborate on this in more detail?" or "For whom was this important?" or "What would you do differently the next time you go through the same situation?"

4. Effort Estimation (EE) Process (General Observations)

"We would like to conclude this interview with some questions about your general observations of Effort Estimation."

26. In most of the cases some amount of past experience (of similar projects) is taken into consideration while doing EE. How do you make estimates when you have to build something that is totally different from any past project you have worked on (and you cannot rely on past experiences)?

27. What are the benefits with respect to EE that you may have observed using DSs as compared to when DSs are not in place?

28. Have there been situations where you feel that a different approach to DSs (more/different information) could have helped EE?

5. Debriefing

"We would like to thank you for your time and we are very appreciative for your input in our research. You have more experience with the intricacies of EE in real project contexts than we do, therefore we have these final questions:"

29. From your experience with EE, is there something else we should pay attention to in our research?
    Probes
    Is there anything else that you think is important for us to know, something that would help us understand how the EE process works in real projects?

30. Are there any other things you would like to share with us?

B. Stakeholders

This study has mentioned many different participants and stakeholders in this project. To be clear about the roles of different stakeholders, this appendix contains an overview of the most mentioned stakeholders.

Business Analyst (BA)	Responsible for the functional specifications in the delivery story. They also create the detailed estimation for their own work.
Delivery Assurance Facilitator	Involved in the quality management of the software after development.
Delivery Manager	Every stream has a delivery manager that is responsible for the ballpark estimations.
Delivery Story (DS) Designer	Responsible for the design of the technical specifications in the delivery story. They also create the detailed estimation for their own work.
Design Lead	Team lead of the DS designers. Responsible for the baseline estimation for the DS specification.
Developer	Team member of the development team. Uses the individual DS to create a detailed estimation.
Lead Solution Architect	Working on the complete project, consolidating estimates and planning.
Programme Manager	Responsible for the delivery of the complete project.
Release Manager	Responsible for the delivery of that specific release.
Scrum Master	Streamlining the scrum process, guiding estimation meetings. Can be doubled by a delivery manager.
Subject Matter Expert (SME)	The customer. Has knowledge about the requirements and provides the elaborated user stories. Can be consulted for questions about critical requirements.
Tech Lead	Team lead of the development team. Responsible for the baseline estimation of software development.
Test Delivery Manager	Team lead of the functional testing team. Responsible for the baseline estimation of functional testing.
Test Designer	Responsible for the design of test scenarios. They also create the detailed estimation for their own work.

C. Mind-map

The mind-map in this appendix is the complete mind-map of all codes and the categories in which they have been divided.