The impact of agile principles and practices on large-scale software development projects

av

Lina Lagerberg & Tor Skude

LIU-IDA/LITH-EX-A--13/006--SE

2013-02-14
Examensarbete

Effekten av agila principer och praxis i storskaliga mjukvaruutvecklingsprojekt

av

Lina Lagerberg
Tor Skude

LIU-IDA/LITH-EX-A--13/006--SE

2013-02-14

Handledare: Kristian Sandahl, IDA, Linköpings universitet
Daniel Ståhl, Ericsson

Examinator: Pär Emanuelsson, IDA, Linköpings universitet
The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Lina Lagerberg and Tor Skude
Abstract

Agile software development methods are often advertised as a contrast to the traditional, plan-driven approach to software development. The reported and argued benefits on software quality, coordination, productivity and other areas are numerous. The base of empirical evidence to the claimed effects is however thin, and more empirical studies on the effects of agile software development methods in different contexts are needed, especially in large-scale, industrial settings.

The purpose of the thesis was to study the impact of using agile principles and practices in large-scale software development projects at Ericsson and it was carried out as a multiple-case study of two projects. One of the projects had implemented a limited number of agile software development practices and was largely plan-driven, while the other project had fully adapted its organization and product design for agile software development. Propositions of possible effects of the use of agile principles and practices in the two projects were generated by a literature review. Empirical data was then collected from online surveys of project members, internal documents, personal contact with key project members and a collection of metrics, to study the presence of the proposed effects.

The study was focused on eight different areas: internal software documentation, knowledge sharing, project visibility, pressure and stress, productivity, software quality and project success rate.

Agile principles and practices were found to:

- Lead to a more balanced use of internal software documentation, when supported by sound documentation policies.
- Contribute to knowledge sharing
- Increase project members’ visibility of the status of other teams and the entire project.
- Increase coordination effectiveness and reducing the need for other types of coordination mechanisms.
- Increase productivity.
- Possibly increase software quality.

Additionally, the study showed that internal software documentation is important also in agile software development projects, and cannot fully be replaced with face-to-face communication. Further, it was clear that it’s possible to make a partial implementation of agile principles and practices, and still receive a positive impact. Finally, the study showed that it’s feasible to implement agile principles and practices in large-scale software development. It therefore contributes to understanding the effects of agile software development in different contexts.
Table of contents

Foreword

1 Introduction..................................................................................................................2
  1.1 Agile software development..................................................................................2
  1.2 Agile transformation at an Ericsson product development unit...........................3
  1.3 The research context - Ericsson and PDU X..........................................................4
  1.4 Thesis purpose ....................................................................................................5
    1.4.1 Definition of agile principles and practices....................................................6
    1.4.2 Limitations....................................................................................................6

2 Theoretical background..................................................................................................7
  2.1 Agile principles and practices .............................................................................7
    2.1.1 Scrum and XP ..............................................................................................7
    2.1.2 Definition of agile practices .........................................................................8
  2.2 Studies on Ericsson ...............................................................................................10
    2.2.1 The effect of moving from a plan-driven to an incremental software development approach with agile practices..........................................................10
    2.2.2 Experienced benefits of continuous integration in industry software product development: a case study ....................................................................10
  2.3 Earlier research on the impact of agile principles and practices on software development ....11
    2.3.1 Communication............................................................................................11
    2.3.2 Project visibility .........................................................................................12
    2.3.3 Internal software documentation................................................................12
    2.3.4 Knowledge sharing.....................................................................................12
    2.3.5 Coordination ..............................................................................................13
    2.3.6 Job satisfaction...........................................................................................14
    2.3.7 Pressure and stress ....................................................................................14
    2.3.8 Project success rate ...................................................................................15
    2.3.9 Productivity ...............................................................................................15
    2.3.10 Software quality .......................................................................................16
  2.4 Scaling up agile......................................................................................................17
    2.4.1 Communication and coordination in large-scale projects................................17
    2.4.2 Knowledge sharing in large organizations ..................................................18
    2.4.3 Large-scale implementation and employee satisfaction ................................18
    2.4.4 Large-scale productivity, software quality and project success-rate ..........18
    2.4.5 Architecture is important in scaling up agile methods ......................................18
  2.5 Synthesis of earlier research .............................................................................19

3 Conceptual framework.................................................................................................22
  3.1 From theory to conceptual framework .................................................................22
  3.2 The conceptual framework...................................................................................22
    3.2.1 Description of conceptual framework - Effects ...........................................24
    3.2.2 Description of conceptual framework - Outcomes ......................................25

4 Research questions ....................................................................................................27

5 Research design .........................................................................................................29
6 Case A: The plan-based project ................................................. 41

6.1 Empirical findings ..................................................................... 41

6.1.1 Data sources ....................................................................... 41

6.1.2 Case description ................................................................... 41

6.1.3 The software development process in Project A – a waterfall approach ........................................................................ 42

6.1.4 Description of Survey A respondents ................................... 43

6.1.5 What agile principles and practices are implemented in Project A? ......................................................... 45

6.1.6 Internal software documentation in Project A ...................... 48

6.1.7 Knowledge sharing in Project A ........................................... 50

6.1.8 Project visibility in Project A ................................................ 50

6.1.9 Pressure and stress in Project A .......................................... 51

6.1.10 The level of coordination effectiveness in Project A ............. 51

6.1.11 Productivity in Project A .................................................... 52

6.1.12 Software quality in Project A ............................................ 55

6.1.13 Project A success rate ....................................................... 55

6.2 Data analysis – what impact does the use of agile principles and practices have in Project A? .............................. 56

6.2.1 Representativeness of the survey respondents ....................... 56

6.2.2 Experiences and perception of agile principles and practices ........................................................................... 57

6.2.3 Impact on internal software documentation .......................... 57

6.2.4 Impact on knowledge sharing .............................................. 59

6.2.5 Impact on project visibility .................................................. 59

6.2.6 Impact on pressure and stress .............................................. 60

6.2.7 Impact on coordination effectiveness .................................... 60

6.2.8 Impact on productivity ....................................................... 61

6.2.9 Impact on software quality .................................................. 62

6.2.10 Impact on success rate ..................................................... 63

6.3 Case A Conclusions ................................................................. 64

7 Case B: The agile project .............................................................. 65

7.1 Empirical findings ..................................................................... 65

7.1.1 Data sources ....................................................................... 65

7.1.2 Case description ................................................................. 65

7.1.3 The software development process in Project B - an agile approach ................................................................. 66

7.1.4 Continuous integration is a key part of the software development in Project B ................................................. 67

7.1.5 Description of Survey B respondents ..................................... 68
7.1.6 What agile practices are implemented in Project B? ........................................... 69
7.1.7 Internal software documentation in Project B ..................................................... 73
7.1.8 Knowledge sharing in Project B ......................................................................... 75
7.1.9 Project visibility in Project B ................................................................................ 76
7.1.10 Pressure and stress in Project B .......................................................................... 76
7.1.11 The level of coordination effectiveness in Project B .......................................... 77
7.1.12 Productivity in Project B .................................................................................... 78
7.1.13 Software quality in Project B ............................................................................... 80
7.1.14 Project B success rate ....................................................................................... 82

7.2 Data analysis – what impact does the use of agile principles and practices have in Project B? ... 82
7.2.1 Representativeness of the survey respondents ..................................................... 82
7.2.2 Experiences and perception of agile principles and practices ............................... 82
7.2.3 Impact on internal software documentation ...................................................... 83
7.2.4 Impact on knowledge sharing ............................................................................. 84
7.2.5 Impact on project visibility .................................................................................. 85
7.2.6 Impact on pressure and stress ............................................................................. 85
7.2.7 Impact on coordination effectiveness ................................................................. 86
7.2.8 Impact on productivity ....................................................................................... 86
7.2.9 Impact on software quality .................................................................................. 87
7.2.10 Impact on success rate ..................................................................................... 88

7.3 Case B conclusions .................................................................................................. 89

8 Cross-case analysis .................................................................................................... 90
8.1 Difference in implementation of agile practices ....................................................... 90
8.1.1 Perception of agile practices .............................................................................. 91
8.2 Has the use of agile principles and practices had an impact on the use of internal software documentation in Project A and Project B? .......................................................... 91
8.3 Is knowledge more easily shared in Project B, compared to Project A thanks to the use of agile principles and practices? .................................................................................................................. 93
8.4 Has the use of agile principles and practices made Project B more visible to team members, other teams and project management, compared to Project A? ......................................................... 95
8.5 Has the use of agile principles and practices increased the pressure and stress employees feel in Project B, compared to Project A? .................................................................................. 96
8.6 Has the use of agile principles and practices increased coordination effectiveness in Project B, compared to Project A? ...................................................................................................... 97
8.7 Has the use of agile principles and practices increased productivity in Project B, compared to Project A? .............................................................................................................. 98
8.8 Has the use of agile principles and practices increased the quality of the software produced in Project B, compared to Project A? .................................................................................. 99
8.9 Has the use of agile principles and practices made Project B more successful than Project A? ................................................................................................................................. 100

9 Conclusions ................................................................................................................ 102
10 Discussion ................................................................................................................ 104
10.1 Results and research method .............................................................................. 104
10.2 Revised conceptual framework ............................................................................ 106
10.3 Transferability ...................................................................................................... 107
10.4 Ideas for future work ......................................................................................... 110
11 Bibliography .................................................................................................................. 112
12 Appendices .................................................................................................................... 116
   12.1 The systematic literature review ............................................................................. 116
   12.1.1 Study selection criteria ..................................................................................... 116
   12.1.2 Data extraction protocol ................................................................................... 116
   12.1.3 Studies found in the literature review ............................................................... 116
   12.1.4 List of studies .................................................................................................... 117
   12.2 Project A survey questions and result summary .................................................. 119
   12.3 Project B survey questions and result summary .................................................. 124

Table of figures
Figure 1. Synthesis of the empirical evidence on what impact agile principles and practices have on
software development......................................................................................................... 20
Figure 2. The conceptual framework .............................................................................. 23
Figure 3. The definition of coordination effectiveness given by Strode et al. [39] ............... 25
Figure 4. Schematic description of the software development process in Project A............. 43
Figure 5. The survey respondents’ experience with agile and non-agile software development. ......... 44
Figure 6. The functional role descriptions of the survey respondents............................... 45
Figure 7. The survey respondents’ satisfaction with agile practices in Project A.................. 47
Figure 8. The survey respondents’ opinion about the amount of different software documentation items in
Project A............................................................................................................................. 49
Figure 9. Time spent daily on producing internal software documentation in Project A [S-A]............ 50
Figure 10. Accumulated system stability test pass rate..................................................... 55
Figure 11. The software development process in Project B............................................. 67
Figure 12. The survey respondents’ experience of agile and non-agile software development.......................................................................................................................... 68
Figure 13. The functional role descriptions of the survey respondents............................... 69
Figure 14. The survey respondents’ satisfaction with the agile practices in Project B.............. 72
Figure 15. The survey respondents’ opinion about the amount of different software documentation items in
Project B............................................................................................................................. 74
Figure 16. Time spent daily on producing internal software documentation in Project B......... 75
Figure 17. Non-acceptance test pass rate and line coverage ........................................... 81
Figure 18. Acceptance test pass rate .............................................................................. 81
Figure 19. Revised conceptual framework ....................................................................... 106
**Table of tables**

Table 1. Main characteristics of Project A and Project B ................................................................. 4
Table 2. An overview of the metrics that were collected and analyzed ........................................ 35
Table 3. Data sources used in the case description of Project A .................................................... 41
Table 4. Implementation of agile practices in Project A ................................................................ 46
Table 5. Work item cycle times in Project A .................................................................................. 53
Table 6. Cycle times for bugs and trouble reports in Project A ..................................................... 54
Table 7. Restricted features in one release of Product A ............................................................... 56
Table 8. Number of features in one release of Product A .............................................................. 56
Table 9. Data sources used in the case description of Project B .................................................... 65
Table 10. Agile practices implemented in Project B .................................................................. 71
Table 11. Project B work item cycle times .................................................................................. 79
Table 12. Work item cycle times for the feature teams in Project B ........................................... 79
Table 13. Bug cycle time in Project B ......................................................................................... 80
Table 14. Agile practices in projects A and B ........................................................................... 90
Table 15. The number of respondents, standard deviation, mean, median, minimum and maximum value for all questions in the Project A survey ........................................................................ 119
Table 16. The number of respondents, standard deviation, mean, median, minimum and maximum value for all questions in the Project B survey ................................................................ 124
Foreword

This report is a result of a master’s thesis conducted at Linköping University in cooperation with Ericsson during the fall and winter 2012 - 2013. The report was produced with readers within Ericsson as the primary target audience, but we hope that it can bring insights for anyone interested in agile software development. We find the subject of agile software development very interesting, and it has been a privilege for us to be able to study it in detail in an industrial setting.

We would like to direct a special thank you to our supervisors Kristian Sandahl at Linköping University and Daniel Ståhl at Ericsson, and to our examiner Pär Emanuelsson at Linköping University and Ericsson for their continuous support and help. We would also like to thank the members of Project A, Project B and PDU X for their cooperation with our study. Finally, we would like to thank everyone else who helped us with our thesis.

Linköping 2013-02-14

Lina Lagerberg and Tor Skude
1 Introduction

The report is commenced with a background to the studied problem, introducing some of the issues concerning large-scale, agile software development. The two studied projects, Project A and Project B, are presented, the corporate context is described and agile software development methods are introduced. Finally the purpose of the thesis is defined along with the limitations made.

If you are working in the software engineering industry you probably haven’t missed the word ‘agile’, a small word that incorporates numerous values, principles and practices for the development of software [1] [2]. Agile software development methods are often advertised as a contrast to the traditional, plan-driven approach to software development [3] and the reported and argued benefits are numerous. Agile software development methods are claimed to increase software quality [4] and reported to correlate positively with project success rate [5], make the employees more satisfied [6], improve communication [7] and coordination [8] and increase productivity [9], to name just a few. However, challenges and limitations with agile development methods are also raised by researchers: the lesser emphasis on documentation could lead to inferior knowledge management in the long term [10] and the focus on constantly delivering results could both increase the pressure on the software professionals [8] [11] and create architectural problems [12].

In general the base of empirical evidence to the claimed effects is thin, and more empirical studies on the effects of agile software development methods are asked for by several researchers, especially on which effects that could be expected in a large-scale, industrial setting [3] [12].

In the multinational telecommunications company Ericsson there is an extensive interest for agile practices and principles and agile transformation programs are executed on several levels in the organization. Ericsson is facing fierce competition from new competitors, low operating margins and a constantly evolving marketplace. By implementing agile methods the company hopes to confront these threats and increase its efficiency and effectiveness, reduce its time to market, retain a serious customer focus and create internal synergies. But little is known whether this is possible to achieve with the implementation of agile principles and practices. This thesis aims to show what effects the use of agile principles and practices have had on two Ericsson software development projects.

1.1 Agile software development

One definition of agile software development that is used inside Ericsson is that of the Agile Alliance. The Agile Alliance is a nonprofit organization committed to advancing agile principles and practices [1]. They describe agile software development as the collection of new methodologies that emerged in the late 1990’s that:

“[..] emphasized close collaboration between the programmer team and business experts; face-to-face communication (as more efficient than written documentation); frequent delivery of new deployable business value; tight, self-organizing teams; and ways to craft the code and the team such that the inevitable requirements churn was not a crisis.” [1]
The Agile Alliance further defines an agile software development as software development that follows the values and principles cited in the Agile Manifesto [1]. The Agile Manifesto is a collection of values and principles created in 2001 and its authors believe that the software development is improved with the application of these principles [13]. Throughout the years several development methods have emerged implementing the agile principles and making them more concrete, giving practices for software professionals to follow and also combining them with other philosophies for effective product development, such as Lean and Kanban. Examples of such methods are Extreme Programming [14] and the project management method Scrum [15]. The Agile Alliance has composed a list of 60 agile practices [16].

There are fundamental differences between the agile approach to software development and the traditional, plan-based one. Agile methods are built on the assumption that the world is unpredictable and therefore aims at being adaptive, flexible and responsive, while traditional methods aims at optimizing the development through a well-planned and formalized process [3]. Another difference is the sequence of events; in agile software development design and implementation are inseparable and done iteratively, while design precedes implementation in the traditional software development process. Testing is also done earlier and more often in agile software development. The waterfall model is one of the first described models of traditional, plan-based software development and many other models, such as the V-model, are adaptations of it [17]. In the waterfall model the steps of the development process - requirement analysis, system design, program design, coding, unit and integration testing, system testing and acceptance testing - are done in separate, consecutive, steps where one step must be finished before the next one begins. Associated with each step are milestones and deliverables that could be used for project management. The criticism of the waterfall model includes that it handles change badly and that it’s not very well adapted to the problem solving process software development really is.

1.2 Agile transformation at an Ericsson product development unit

One of the units inside Ericsson where an agile transformation is ongoing is the software Product Development Unit X (PDU X). This department runs, among several other software development projects, Project A and Project B. At a quick glance, these projects may appear similar; they both employ hundreds of software professionals with similar backgrounds, they are located at the same Ericsson development site and they both develop large, complex software products with equivalent functional requirements. But with a deeper look at the internal project organizations and the ways of working, the differences are major.

Project A, on the one hand, develops a product (Product A) that has been under development for many years and follows a traditional, plan-based development process that could be characterized as the waterfall model. Separate, functional departments are responsible for the different phases in the development process and extensive, document-intense handovers are necessary between these. Throughout the years, Project A has struggled with challenges in integration, dependency management and flexibility. In an attempt to overcome the challenges Project A has introduced several agile principles and practices into their product development

---

1 The names of organizational units, projects, products and people have been altered to protect Ericsson’s intellectual property.
process during the last couple of years. For instance, the work in the design and implementation phase is now in several teams done in iterations, continuous integration and test automation have been implemented in parts of the project and the Scrum methodology is used by several teams. However, there is no end-to-end implementation of agile software development throughout the whole project; up-front plans are still used extensively, the development teams are still separate, handling separate phases in the development process and testing and integration is to a large extent still done manually.

Project B, on the other hand, had the great opportunity of starting fresh with both product architecture and organization and has accordingly taken the agile transformation several steps further than Project A. The project’s purpose is designing, developing and deploying a product (Product B), which in the long run can replace Product A. The project was launched in the beginning of 2012 and the whole project follows a truly agile development method inspired by Scrum, uses cross-functional teams and implements continuous integration and test automation to a large extent. The hope with the new ways of working, according to project management, is to be able to develop an equivalent product with fewer people in a more efficient way. Hopefully many of the challenges present in Project A are also overcome in Project B. But whether this has been achieved or not has not yet been investigated. No studies on the implemented agile practices in the two projects have been done.

The different characteristics of Project A and Project B are summarized in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product age</strong></td>
<td>Older than 10 years</td>
<td>Approximately 1 year</td>
</tr>
<tr>
<td><strong>Project members 2012</strong></td>
<td>420 (+ 480 external consultants)</td>
<td>120</td>
</tr>
<tr>
<td><strong>Total number of teams</strong></td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td><strong>Team size</strong></td>
<td>1-90 persons</td>
<td>3-13 persons</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>Separate subprojects for System Analysis, Design and Integration &amp; Verification</td>
<td>Cross-functional teams. Separate Continuous Analysis and Release Verification units.</td>
</tr>
<tr>
<td><strong>Development team’s responsibility</strong></td>
<td>Horizontal: architecture layers</td>
<td>Vertical: features</td>
</tr>
<tr>
<td><strong>Process model</strong></td>
<td>Plan-based, waterfall</td>
<td>Scrum</td>
</tr>
<tr>
<td><strong>Iteration length</strong></td>
<td>1 month</td>
<td>3 weeks</td>
</tr>
<tr>
<td><strong>Product release frequency</strong></td>
<td>9 months</td>
<td>3 weeks (potentially)</td>
</tr>
<tr>
<td><strong>Integration frequency</strong></td>
<td>Once per month</td>
<td>Continuously</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Component-based</td>
<td>Service-oriented</td>
</tr>
<tr>
<td><strong>Adoption of agile practices</strong></td>
<td>Moderate</td>
<td>High</td>
</tr>
</tbody>
</table>

### 1.3 The research context - Ericsson and PDU X

LM Ericsson, later Telefonaktiebolaget LM Ericsson, was founded in Sweden in 1876 [18], and has since then grown to be a global telecommunications equipment company with more than 100 000 employees world-wide [19]. The largest share of Ericsson’s 227 billion SEK sales comes
from fixed and mobile telecommunications infrastructure equipment and software, including 2G, 3G and 4G mobile voice and data (broadband) base stations. Software revenue accounted for 24% of the total revenue in 2011, making Ericsson the world’s fifth largest software company [20]. Ericsson also provides services to its customers, telecom operators, with global services accounting for almost 100 billion SEK in sales. The company is number one in the world in mobile networks and in telecom services. Ericsson is divided into four business units, where Networks is the largest one.

Project A and Project B both belong to the Product Development Unit X (PDU X). PDU X is responsible for the design, development, deployment and maintenance of predominantly software systems, i.e. it is not involved in hardware development. The unit belongs to the Development Unit Radio, which in turn belongs to the Business Unit Networks. In parallel to the Development Unit Radio there is also a Product Area Radio, which is responsible for sales and distribution. The product area is often the purchaser organization to the development unit, specifying high level product requirements. In this sense Project A and Project B have both internal and external stakeholders.

Up until November 2012 PDU X was organized into a line organization with twelve lines. Six of these were supporting lines, with responsibilities such as Systems, Customer Support and Integration & Verification, while six of them were responsible for the development of different product families. Today the unit has instead organized its products into requirement areas including all steps of the development process, to support the agile software development methods it strives to use. Separate lines for Continuous Analysis and Continuous Integration are still present, but their roles are of a more supporting kind than before. In the PDU X unit there are also supporting units for Financial Control and Human Resources. PDU X has its head quarter in Europe, but does also have development offices around the world.

During the course of writing this thesis, there have been layoffs and changes in Ericsson’s organization, in PDU X and in the rest of the world. We have strived to keep the descriptions up to date.

1.4 Thesis purpose
The expectation inside PDU X is that the agile principles and practices implemented in Project A and Project B has had a positive effect on the software development. The expectation is also that the even more agile way of working in Project B has led to even more improvements in Project B, compared to Project A. But the effects are unknown. The purpose of this thesis is therefore to investigate how the use of agile principles and practices have affected the software development process and the products in Project A and Project B.
The purpose of this thesis is thus to answer the question:

**What impact does the use of agile principles and practices have on the large-scale software development projects Project A and Project B?**

The purpose will be answered through a multiple-case study of Project A and Project B. The two projects are chosen because of the different extents to which they have adopted agile principles and practices. Project A is considered to represent a more traditional, plan-driven development process, with fewer implemented practices, while Project B represents a more agile approach with more implemented practices. The supposition is that the two projects, Project A and Project B, differ sufficiently in their approach to software development that the impact of agile software development will be possible to study by a comparison between the two projects.

### 1.4.1 Definition of agile principles and practices

Since PDU X follows the definition given by the Agile Alliance their definition is considered suitable also for this thesis. Thus we define:

- An agile software development as a software development that follows the values and principles stated in the Agile Manifesto.
- An agile software development method as a software development method that follows the values and principles behind the Agile Manifesto.
- The agile principles as the twelve principles stated in the Agile Manifesto
- An agile practice as a practice that implements the values and principles behind the Agile Manifesto.

### 1.4.2 Limitations

The focus of the thesis is on internal factors. This means that the design, implementation, integration and testing phases are the main phases under study, not the pre-study and requirement analysis nor the deployment, usability and acceptance testing or maintenance. In consequence there is a focus on how agile principles and practices have affected the efficiency of the development, not the externally focused partner of efficiency – effectiveness. Customer interactions and feedback are not investigated.

Ericsson is a huge company with many software development units. We will only study the agile practices and methods implemented in PDU X and only in the two chosen projects. However a discussion of the general applicability of the results to other projects inside Ericsson and even to other companies is made in chapter 10.

There exist multiple factors that may influence the effects and outcomes of a development process. In this thesis we study the ways of working.
2 Theoretical background

The report’s theoretical framework is initialized with a definition of the agile practices treated in the report before earlier studies on agile methods conducted at Ericsson are presented. Earlier research on the impact of agile principles and practices on software development projects in other organizations is also presented, followed by a discussion on large-scale implementation of agile principles and practices. The chapter concludes with a synthesis of earlier research.

2.1 Agile principles and practices

The Agile Manifesto lists a set of values upon which agile software development relies. Along with these values, there is also a set of principles. Principles are “domain-specific guidelines for life” [14, p. 15], showing how the values can be applied in different areas. Principles also act as a bridge over the gap between values and practices, bringing the values into concrete, specific application.

There is no evident list of agile practices that has been adopted as the industry standard: the list of agile practices presented on the Agile Alliance website contains 60 different practices [1], but in a systematic literature review from 2010 only 17 agile practices were found to be commonly mentioned in research publications [2]. It rather seems like each organization will apply the set of agile practices suitable for its needs and constraints [5]. It’s also possible that organizations practice agile software development unknowingly. In a survey conducted by Abbas et al [5], 59% of the respondents said that they didn’t follow an agile software development method, but only 34% didn’t use any agile practice.

The list presented below is a summary of practices treated in this thesis. The list is not exhaustive with regards to agile software development in general. As many of the practices come from Extreme Programming (XP) and Scrum a brief description of these methods is also presented.

2.1.1 Scrum and XP

Scrum

Scrum is mainly a project management method [3]. It advocates an iterative and incremental development process [15]. One iteration is called a sprint and driving the sprints is a list of requirements, gathered in a product backlog. A Scrum project contains only three roles [15]. The Product Owner represents the stakeholders of the project and the system and is responsible for prioritizing the requirements in the product backlog. The Scrum Master is responsible for the Scrum process, to adapt the general Scrum methodology to the needs of the project and to educate the project members in the Scrum process. Finally the Team is responsible for developing functionality. The teams should be self-organizing, self-managing and cross-functional [15].

Extreme programming

Extreme programming (XP) is a software development method originally created by Kent Beck and defined in his book Extreme Programming Explained, published in 1999 [14]. XP focuses on programming techniques, communication and teamwork. The methodology contains a body of practices, many of which have been incorporated into what is today considered agile software development practices.
2.1.2 Definition of agile practices

In this thesis, we have used the following definitions of agile software development practices.

**Backlog** - A backlog is a list of requirements or features to be done [16]. It can be associated with a single iteration, or the project as a whole. Items in the product backlog are ordered according to priority. Items from the product backlog are broken down into smaller parts (work items) and moved to the sprint backlog during iteration planning.

**Collective ownership** - The most common form of collective ownership is collective code ownership. This means that every programmer in a team has access to and is expected to make changes in the code where he or she finds it is necessary [14]. No specific developer should own any part of the code by being the single person allowed to edit it. Another interpretation of collective ownership is that the whole team should commit to the team goals, and that the responsibility for reaching the goals ultimately lies with the team itself, and not with any individual team member [16].

**Continuous integration** - Continuous integration is the practice of continuously incorporating the work progress of everyone in a project [16]. This includes build automation and automated quality control.

**Daily meetings** - Agile development teams should have daily, short meetings to keep everyone up to date with the current progress and what others are doing [16]. Meetings are often held every morning. A standardized form of daily meeting is the daily stand-up, or the daily scrum, which is held with the participants standing up to limit the time [2]. During a daily stand-up, each team member informs the rest of the team what progress has been made since the last meeting, what is planned for the day and if there are any impediments.

**Demo** - At the end of an iteration, a working product incorporating the work that was completed should be demonstrated to the Product Owner and other stakeholders [15]. Also known as sprint review.

**Empowerment** – Empowerment refers to that agile development teams should be autonomous, self-organizing and self-managing [15].

**Frequent releases** – Frequent releases is when working software is frequently released and handed over to a customer or customer representative [16].

**Information radiators** – An information radiator is a highly visible physical display of updated information about the software development project [16]. A task/scrum board is an example of an information radiator. Another example could be a flatscreen on a wall that displays the latest test results.
Iteration planning – Iteration planning is done to decide what should be done in the coming iteration. This often includes preparing a sprint backlog, breaking down requirements into smaller work-parts, estimation of effort as well as planning what features should be included in coming releases [15].

Iterative and incremental development – is when software development is done in iterations, often called sprints. In each sprint, visible functionality – increments - is added to a working piece of software [16].

On-site customer – This practice means that a customer should be available on-site, working with the developers [2]. The customer is able to help with clarifying requirements and prioritizing the backlog.

Open office space – Is when agile development teams are located together in an open environment that enables face-to-face communication [2]. A large room with small cubicles suits these criteria. There should also be a communal area, and space for information radiators.

Pair programming – Pair programming is the practice of two programmers using a single computer with a single mouse and keyboard at the same time [14]. Programmers take turns using the keyboard, switching at least a couple of times per day. The programmer that is not using the keyboard reviews the code as it is typed in [2].

Refactoring – Refactoring means to improve the internal structure of source code, without causing any external changes to behavior [16].

Retrospectives – Retrospectives are meetings held at the end of an iteration to reflect on what went well and what could be improved for the next iteration [16].

Sign up – When deciding who should do what, the practice of sign up means that teams members sign themselves up for tasks, instead of being assigned by someone else [16].

Simple design – Is when developers use the simplest possible design instead of trying to predict future needs [2].

Sustainable pace – Working in a sustainable pace means that a team should strive to work at a pace that can be sustained forever [16].

Task board – A task board is a place where tasks and their progresses are visualized [16]. An example of a task board is a white board which is divided into three columns: to do, in progress, and done. Tasks are represented by post it-notes and are moved along the board as they progress.

Whole team – The practice of whole team means that an agile software development team consists of people with a broad variety of skills, including designers, developers and testers [2]. Teams that incorporate this practice are also denominated cross-functional.
2.2 Studies on Ericsson

Several studies on the effects of agile principles and practices have previously been conducted at Ericsson. In this section, a summary of the findings of two earlier studies is presented.

2.2.1 The effect of moving from a plan-driven to an incremental software development approach with agile practices

This study was conducted by Petersen and Wohlin at an Ericsson product development site in the end of 2007 [21]. The researchers studied the effect of moving from a plan-driven process to an incremental development approach with a three month development cycle. The new software development model realized the agile practices of *iterative and incremental development, frequent releases, product backlog* and *frequent integration*. The authors found that several weaknesses with plan-based software development were improved or amended by the use of agile practices.

The *incremental development* process enabled a higher release frequency, which was believed to benefit the organization by leading to an earlier return on investment. There was also a clear improvement in the reduction of waste, i.e. less work that didn’t contribute to customer value, due to a better identification of requirements. Furthermore, software quality was improved due to early testing which led to early fault detection and feedback from testing, which in turn caused a lower fault-slip-through. It was also believed that the higher degree of traceability created incentives for teams to deliver high quality work. The maintenance effort, which had been constantly rising when using the old approach, settled at a constant level. There was also a reduction in the need for documentation, since much of it was earlier required for hand-overs from phase to phase. Parts of the documentation could be replaced by direct communication as project teams focused on several phases.

Nonetheless, some issues remained after the migration, and others were caused by the new ways of working. Testing could overrun a development cycle, with the consequence that some packages had to wait for the next release. There were still problems with test coverage. The development cycle didn’t allow enough time for quality testing on the system level. The organization into many small teams increased the need for communication and coordination. An anatomy plan was used to address this issue by providing a structure of the system and its dependencies and decide on the order of subprojects. The release project, responsible for making shippable, customized products, was involved too late in the process. This hindered the product from being viewed in a commercial perspective. The implementation of the practice *whole team* with small, cross-functional teams required the team members to have a very broad knowledge. This was an ample change from the previous approach, where people were highly specialized.

2.2.2 Experienced benefits of continuous integration in industry software product development: a case study

This study, made by Ståhl and Bosch [22], included four different Ericsson product development projects with different experiences of and approaches to *continuous integration*. The authors could through quantified data from 22 interviews conclude that continuous integration contributes to improved communication both within and between teams, improves project predictability as an effect of finding problems earlier and that it contributes to increased developer productivity as an effect of facilitating parallel development in the same source context. Enough support could not
be found for the formulated hypothesis that continuous integration should also contribute to increased productivity by decreasing the work of compiling and testing locally before checking in.

2.3 Earlier research on the impact of agile principles and practices on software development

This section presents what impact the use of agile principles and practices have been found to have in other software projects, albeit not large-scale ones. The summary constitutes the findings from the literature review. A description of how the literature review was conducted is available in section 5.2.2.

2.3.1 Communication

Communication is important in software development projects to manage dependencies between actors in the process, help people avoid conflicts and achieve goals [7]. It’s also an important contributor to effective coordination [7]. McHugh et al have expressed that “Agile practices require constant interaction and frequent communication” [23, p. 73] in general, and there are several studies, including that of McHugh, that show how agile software development practices also improve communication compared to plan-driven development. The main contribution of agile practices is creating forums and instruments for formal and informal communication.

The practice of open office space has been found to improve the informal communication within teams and reduce the need for formal documentation [7] [21]. The higher degree of informal communication also facilitates problem solving, which contributes to the achievement of iteration goals [7]. However, the open office environment increases the risk of overhearing conversations among other team members and can thereby cause a loss of attention [7] [24]. Individual desks separated by half-height glass barriers are a way to reduce this risk and yet facilitate effective communication [24].

Iteration planning and demos are practices associated with formal communication that have been showed to, in a good way, increase the understanding of requirements and features and handle dependencies between the two. The keeping of a product backlog is also enhancing communication around features and requirements [7]. Daily meetings are formal communication forums that increase communication frequency and can be used for solving and clarifying problems [4]. But the time for technical analysis and discussions during iteration planning sessions and demos is not perceived as enough to solve all issues, especially not as the complexity of the product grows [7].

Nevertheless there are studies that describe communication problems also in agile software development teams [7] [11] [25]. Lee and Xia [25] describe, for example, how team diversity (whole team) can cause conflicts and slow down communication. Retrospective meetings after each sprint/iteration have been reported as an efficient way of improving the implementation of agile practices in a project [7]. But it has also been reported that this practice failed to capture and deal with problems not adherent to the development process, i.e. communication problems and
conflicts present in the development team [11]. McHugh et al [23] report how issues only sporadically were brought up on retrospectives and that they seldom generated follow-ups.

2.3.2 Project visibility

Another aspect of communication is communication of the project status, plans and goals to project stakeholders, such as organizational management, project members, other organizational units, customers and users. In agile software development methods this is often referred to as project visibility. The definition of visibility in Encyclopedia Britannica is the “capability of being readily noticed” [26].

Several agile practices have been shown to have a positive influence on project visibility. Iteration planning increases the project team’s awareness for the goals and plans for the next iteration, as does open office space [7]. Demos are suggested to improve the customer’s visibility of project plans and status [7]. Visible task boards give an overview of the project’s status at a quick glance for both the team and passers by, such as management and customers [7]. Continuous integration gives testers/quality engineers updates on the status of the end product [7]. Iterative development, with a constant fixing of defects, is reported to give a better overview of remaining work [4] and to give more frequent feedback of project status to managers than a plan-driven process [7]. However, some studies also report problems with estimation in agile software development projects, for example [11] and [23], why the project status shown may not always be correct.

2.3.3 Internal software documentation

Although the Agile Manifesto [13] doesn’t state that an agile software development process shouldn’t generate any documentation, agile methods are usually associated with less documentation than traditional ones [10]. In the literature review we found indications that the produced documentation in agile software development projects may not be enough. In a survey study [10], the respondents felt that the amount of internal documentation generated in their projects wasn’t enough. As opposed to the Agile Manifesto the agile practitioners also perceived internal documentation in their projects as important or even very important. The survey included 79 software engineering professionals and 8 teams in 13 different countries and the majority of respondents spent less than 15 minutes daily on writing documentation. In the case studied by Pikkarainen et al [7] the testers, who were separated from the development team, reported that the Scrum methodology used by the development team didn’t produce enough technical documentation for them. To solve this, the development teams organized workshops were the stakeholders, e.g. testers, could participate and then generate the documents themselves [7]. See chapter 2.3.4 below for further discussion on how software documentation may affect knowledge sharing.

2.3.4 Knowledge sharing

Several agile practices are associated with knowledge sharing and learning. Open office is claimed to improve developer learning by improving communication [24]. Retrospectives, daily meetings and iteration planning create open forums for creating and sharing knowledge [23]. Whole team, widens the developers’ knowledge, but decreases the need for specialization [21]. The iterative development process, where project progress is measured continuously and feedback on project status is given to both team members and stakeholders, can increase the awareness of the capabilities and knowledge of the team. McHugh et al [23] give an example from one of their case studies “The
[Product Owner] can quickly ascertain whether team members are knowledgeable and competent to deliver what they promise, which increases trust through good will and more realistic perceptions of the team members’ individual capabilities” (p. 74). This statement is supported by Strode et al [8], who claim that the availability of team members, for example when they are sitting close to each other in an open office environment or are members of a whole team and can substitute one another increases the awareness of who knows what.

Demos could also increase knowledge and understanding as the developers preparing for demonstration have to understand all code the team has developed, including parts they haven’t worked on themselves [8]. However, the reduced emphasis on internal documentation in agile software development may decrease knowledge sharing in the long term, especially in an environment where the team setting isn’t constant; if knowledge isn’t recorded it can be forgotten. Team members may also quit and thereby take undocumented knowledge with them. In a survey study [10] on agile practitioners’ perception on software documentation, the respondents didn’t agree that face-to-face communication is sufficient to capture and sustain knowledge, as verbal communication is more prone to be forgotten, especially in the long term. Mishra and Mishra [24] argue that physical tools like papers and whiteboards should be used for storing information for later use to facilitate corporate learning.

The ability to share and build upon each others’ knowledge requires a degree of trust between team members. If new members are added to a team with a slow progress and problems, the old members can start to defend their previous work instead of sharing knowledge [11]. Daily meetings and retrospectives can, if they work well, increase the trust in a software development team as they require the developers to interact and communicate [23]. McHugh et al [23] found in their study that the team members in three agile projects were not afraid to ask other team members for help, or to offer help and thus demonstrated a high level of trust.

2.3.5 Coordination
Coordination within the team could be enhanced by agile software development methods. Strode et al [8] discuss coordination in terms of coordination strategy, which they define as “a group of coordination mechanisms that manage dependencies in a situation” [8, p. 1230], and coordination effectiveness. The authors have chosen to call it coordination strategy as these mechanisms are, according to them, chosen consciously by project stakeholders. Their proposition is that a coordination strategy consists of three elements: synchronization, structure and boundary spanning, and that these elements contribute to achieve implicit and explicit coordination effectiveness. The authors further describe how several agile practices help to achieve effective coordination by being a part of the coordination strategy. Information radiators and daily meetings help team members to stay up to date with what to do, and what other team members are doing, which is defined as implicit coordination effectiveness. To be located within close proximity of each other, preferably in the same room with adjacent desks, in a so called open office space, could also increase coordination effectiveness, according to Strode et al.

To coordinate effectively the members should additionally have high availability to each other, i.e. be assigned full time to the project [8]. Strode et al also describes how the agile practice whole team, when team members have the expertise and skill to perform the tasks of other team members,
will contribute to effective coordination since it reduces workload bottlenecks and increases the understanding of what other team members do, although they use the term substitutability between team members instead of whole team.

2.3.6 Job satisfaction

Job satisfaction is a complex concept and there is not one single factor deciding the satisfaction a software engineer gets from his or her work. Nevertheless, agile software development methods can be believed to have some influence on the matter. Reports show that the use of agile practices both make software professionals more satisfied with their job [6] and at the same time puts them under more pressure and stress [8] [11]. Malnik and Maurer [6] found that members of agile software development teams were significantly more satisfied with their job than IT industry professionals in general. There were also a significant positive correlation between the experience of agile methods and job satisfaction: the longer the respondents had worked in agile software development teams, the more satisfied they were [6]. The most important factors correlating with the level of agile software development experience, which thereby can be believed to have influenced the higher level of job satisfaction, were the ability to influence decisions that affect the individual, the opportunity to work on interesting projects and the relationships with users/customers [6]. Of these factors the first one can be associated with the agile principle empowerment and the last one with close customer collaboration/on-site customer, while the interestingness of projects is considered to be outside the influence of agile principles and practices. That the use of agile software development methods increases job satisfaction, is supported by studies found in the literature review of Dybå and Dingsøyr [3].

The job satisfaction of employees, if not considered a purpose of its own, has been shown to decrease the level of employee voluntary turnover, i.e. perceived desirability of movement [6]. Job turnover is considered to cause large costs for a software company, including recruiting and training cost, overload on team and loss of customers. Whether or not there is a connection between employee satisfaction and productivity – that happy employees work faster – is not agreed upon by researchers.

2.3.7 Pressure and stress

The agile practice of demo could increase the stress, as new bugs are possibly found and need to be fixed quickly [8]. Also daily meetings can, if there isn’t an atmosphere of trust inside the team, increase the pressure and cause stress [11]. Stray et al [11] explains how the daily meetings inside a project under study became a place where the developers felt the need to defend what they had done between meetings and an obligation to show progress. Sign up can also increase the pressure on team members as they feel obliged to complete a task within a time frame once they have committed to do so [23]. The increased pressure is reported to be self-inflicted by the developers, but intensified in part by an increased project visibility, as everybody is aware of whose responsibility an uncompleted task is [23]. It’s noteworthy that this individual responsibility goes against the principle of collective ownership, where the whole team is supposed to take responsibility for completion of tasks. The explanation could be that there is a conflict between assigning tasks or bugs to individual developers but still expect the whole team to take responsibility for their completion, as reported by McAvoy and Butler [27]. In one of the teams they studied, the root of this problem was the behavior of the team manager, who used the assignment list to follow up on trouble reports by asking the developer assigned to a certain report. In that way the project
manager, unintentionally, created an incentive only for that developer to correct the report and thereby counteracting the collective ownership [27].

2.3.8 Project success rate

Abbas et al [28] found in their survey that organizations with experience from agile software development tended to have a higher project success rate. They have not defined exactly what they mean by project success, but for the purpose of this thesis the two concepts of on-budget and on-time completion have been included in the concept of project success.

By using the practice of iterative development and earlier and more frequent testing the team can avoid large repairs in the end and thereby increase the possibility to deliver the project on schedule [4]. A software team’s response efficiency, which in turn is affected by empowerment, also has a positive effect on a project’s on-time and on-budget completion [25].

On the contrary some agile practices, daily meetings and collective ownership can be considered factors that, under special circumstances, can contribute to maintaining escalating commitment [11]. Escalation of commitment can be defined as “increasing commitment to a failing course of action” [11, p. 153]. This phenomenon has been found to happen in 30-40% of all software projects [29].

2.3.9 Productivity

Agile software development methods are often associated with increased productivity; however we have found it hard to find empirical support for this association. The systematic literature review undertaken by Dybå and Dingsøyr [3] up to and including 2005 found only two studies on the productivity of agile software development methods compared to traditional ones that were conducted in an industrial setting. However, both studies showed an increased productivity: [30] showed a 41% increase in productivity and [31] a 46% increase. Illieva et al [30] investigated the effect of introducing eXPERT, which is a software development method incorporating the practices of XP with the use of time, effort and defect logs. The authors found that using eXPERT increased the productivity in a project with 41% compared to a similar, baseline project. They derive this, in part to the higher amount of newly produced code in the eXPERT project, but do conclude that part of the productivity increase is because of the change in development method. Layman et al [31] report that a 46% increase in productivity was reached over time by adopting the XP methodology. This case study was though done on a team consisting of only 6-10 people, and the authors make no claim that the results should be applicable on larger teams. Worth mentioning is that both studies measured productivity solely in terms of number of lines of code produced per hour.

In our literature review several studies discussed productivity from different perspectives, although none had the explicit research aim to study productivity. No study produced any explicit evidence to an increase in productivity thanks to the use of agile principles and practices, although some studies touched the area. As noted by Peterson and Wohlin [21] in one of the earlier studies conducted at Ericsson, a commonly perceived overall issue with plan-driven software development is the amount of rework associated with requirements change. Introducing agile software development practices made requirements more reflective of customer demand and thus reduced the amount of rework needed. The amount of waste was also reduced. Nevertheless, the authors argued that no increase in productivity could be seen. A study by Li et
al [4] showed that the *iterative development* practice of Scrum resulted in constant system and acceptance testing and argued that Scrum made the development process more efficient by reducing the number of surprises and increasing the control of software quality and release date. However, due to a lack of data, Li et al [4] didn’t manage to measure the defect fixing efficiency.

Lee and Xia [25] conducted a study on agile practices and principles promoting team autonomy (*empowerment*) and team diversity (*whole team*). These two dimensions of agility were found to have an indirect impact on software development productivity, by affecting how the development team responds to changing requirements. Team autonomy makes a team’s response to changing requirements more efficient but less extensive, while a team with a high degree of diversity will respond more extensively to change. Response extensiveness is in this case defined as the proportion of changing user requirements that the team acts on. Because of the complicated relations between team autonomy, team diversity and the productivity of the project as a whole, it can be hard to find the necessary balance between the two team agility dimensions.

In the references of the initially found articles, a couple of additional articles on productivity were found. Some of them reported on a perceived increase in productivity, which couldn’t be backed with quantitative data [32] [33]. A large productivity increase (201%), compared to a productivity baseline, was found by Sutherland et al [9] in a pilot project combining Scrum and CMMI² level 5. The project was considered large by the authors, but actually just involved four people. It is however not possible to attribute the benefit solely to Scrum, since the project did additional improvements.

### 2.3.10 Software quality

Abbas et al [28] found a positive correlation between organizations with agile experience and code quality in their study, but include no discussion on the reasons for this. Surprisingly, few other studies concerning the relationship between agile software development and quality were found in the literature review. The ones that were found showed an increase in perceived quality, but couldn’t back that claim with numbers. One such example is the case studied by Li et al [4] where the authors reported how the *iterative development* and the continuous received feedback through *daily meetings* and *retrospectives* increased the focus and awareness on software quality. But the study didn’t show a decrease in the number of defects introduced by developers, despite the fact that a decreased defect density was the general perception by the developers. The developers believed that the defects should have decreased due to a higher motivation to fix defects earlier and conduct more unit tests [4]. In another study, Strode et al [8] describe how additional bugs sometimes where found during the preparation for *demos*, which could be believed to increase the quality.

As noted above by Lee and Xia [25], team autonomy (*empowerment*) can have a negative impact on a team’s response extensiveness, which in turn may have a negative effect on software functionality. On the contrary, a team’s response efficiency has a positive influence on software functionality.

---

² Capability Maturity Model Integration, a process improvement method [50]
2.4 Scaling up agile

Not much research has been done concerning agile methods in large organizations, with many teams and long projects [3] [12]. Nevertheless we have found some evidence of effects of agile methods in large-scale software development. We present this here, together with a discussion on what to expect of the previously presented effects in a large-scale, industrial context. All the effects presented in the earlier section (section 2.3) are reported in industrial software development projects, but not large-scale ones.

2.4.1 Communication and coordination in large-scale projects

The study by Petersen and Wohlin [21], which was conducted in a large-scale software development project at Ericsson, reported improved communication as one of the benefits gained from using agile principles and practices. Increased project visibility was reported in a survey study from the large Finnish telecom company Nokia [34]. This indicates that better communication and increased project visibility could be achievable also in large-scale implementations of agile software development.

However, communication and coordination do become more complex as the project grows. Practices such as iteration planning and product backlog have earlier been reported to facilitate discussions about requirements and features. But, they are also believed to not be enough to handle the complexity when the number of requirements and features grow [6], as could be expected in larger projects. It’s also argued by several researchers that the long term goals and interactions between features could be hindered by the time-boxed iteration planning sessions [7].

The increased need for coordination, which led to larger management overhead costs found by Peterson and Wohlin [21] raises the question on how effective coordination between teams could be achieved. Peterson and Wohlin describe how one unit inside Ericsson uses a project anatomy plan to handle dependencies between features, which can be seen as an effort to achieve effective coordination in large projects. Strode et al [8] treat the issue under the name of “boundary spanning” and explain how several agile practices facilitate such boundary spanning and thus contribute to an effective coordination. Demos for a group from another unit and ad-hoc planning and breakdown sessions with participants from other units could be seen as boundary spanning activities [8]. Strode et al also saw the advantage of having a coordinator role, which could coordinate between the team and other units or levels in the hierarchy. The authors suggest that the Scrum Master take on this coordinator role.

Strode et al [8] further name two prerequisites for effective coordination inside teams: proximity and availability and it could be argued that the same prerequisites are necessary for effective coordination between teams, i.e. when the units in need of coordination are teams instead of individuals. In that case the teams should be located in close proximity of each other, but with the possibility for focused work inside the team, and the different teams should be available for frequent communication and collaboration. This conclusion is supported by Mishra and Mishra [24] that state that inter-team communication, collaboration and coordination can be helped by separate but adjacent rooms for different teams. Their study was however conducted in a project with only two teams.
Larger organizations and larger projects also tend to have longer iterations [28]. This is somewhat contradictory to the findings of Strøde et al [8], who argues that reducing the iteration length is a way to deal with complexity and maintain effective coordination. Large organizations and projects, where complexity and coordination are two important issues, could therefore be supposed to benefit from shorter iterations.

### 2.4.2 Knowledge sharing in large organizations

Large-scale development projects involve several teams, why inter-team knowledge sharing and learning also become interesting. In the study by McHugh et al [23] members from other teams, for example Quality Assurance, were present in team meetings such as daily meetings and retrospectives. This helped building relationships across team borders [23] and could be seen as a way to improve the conditions for inter-team learning. In her experience report from Yahoo!, Benefield [33] describes how members of newly converted agile teams were encouraged to attend the meetings of more experienced teams to learn the Scrum method. It’s important however, to be careful with who’s allowed to participate in team meetings so that the team members doesn’t feel the need to report or defend their work [11].

### 2.4.3 Large-scale implementation and employee satisfaction

The Finnish telecom company Nokia has been successful in their implementation of large-scale Scrum, at least from an employee satisfaction perspective [34] [35]. In two survey studies conducted 2008 and 2009, which both received more than 500 respondents, a vast majority (75% and 76% respectively) of the employees showed a positive attitude towards agile software development. The positive attitude also didn’t change significantly over the year between the two surveys, indicating that agile software development isn’t a hype that loses its thrill once implemented [35].

### 2.4.4 Large-scale productivity, software quality and project success-rate

The decreased waste and rework showed in the case study of Petersen and Wohlin [21] could be seen as an indication that the productivity gain from agile methods is reachable also in large-scale organizations. The study was conducted at a different Ericsson unit than this thesis has chosen to study, but where the development process still is large-scale.

No specific studies on quality in large-scale organizations have been found. However [33] report how the members of agile teams in a large-scale implementation at Yahoo perceived an increase in overall quality.

The survey by Abbas et al [28] shows a tendency for agile development to be less successful in large organizations.

### 2.4.5 Architecture is important in scaling up agile methods

In a systematic literature review on agile and architecture the authors Breivold et al [12] state that, from their practical experience in scaling up agile methods architecture plays an important role. The main finding of their study is that few claims about the relationship between agile software development methods and architecture are based on empirical evidence. However, the authors highlight several interesting issues. For example they express the fear that the focus on current requirements, especially enforced by the XP practice simple design, jeopardizes the architectural
support for future customer needs. To combine agile methods with plan-driven characteristics is proposed as a feasible solution to this problem [12]. Another proposed solution to achieve a sustainable long-term perspective in an agile software development process is to combine agile methods with software product line engineering, according to Hanssen [36]. The idea of software product line engineering is the development of a set of artifacts such as software components, architecture design and data structures, which could be configured efficiently into applications [36].

Breivold et al [12] further raise the issue that the focus of early and constant results, that come with the practice _incremental and iterative development_, can be feared to decrease the scalability of the system. In the case study by Hanssen [36] empirical evidence of this phenomenon can be found; the priority to serving the market rapidly caused the system entropy to escalate, which led to a time-consuming necessity of refactoring the entire product line architecture. Also in the case reported by Stray et al [11] indication of the same effect can be seen. In this case the self-developed architecture framework in an agile software development project had to be exchanged three years into the project, in favor of a company-based framework. It's not possible to determine that the architectural problems were due to the agile software development process from the brief description of this event that Stray et al [11] gives. But what the authors do conclude is that the agile software development practices, especially _daily meetings and retrospectives_, failed to capture and stop the escalating problems in time [11]. Furthermore the developers in the case study of Li et al [4], reported to not have time for architectural work such as _refactoring_ due to the constant pressure to deliver, but the effects of this lack of architectural work wasn't discussed. On the contrary Pikkarainen et al [7] report that the continuous delivery of working software led to an earlier revelation of integration problems than before. Their project was however a small scale one. Also [33] report that the Scrum methodology used brought up architectural issues early enough for them to be corrected.

To be able to respond quickly to changing market needs and at the same time ensure the long-term survival of product assets Breivold et al [12] proposes that agile software development methods should be combined with architecture-centric ones, such as Architectural Lines. According to the authors this combination has several reported benefits, among others increased project visibility, early identification and solving of technical problems, efficient information sharing and constant development rhythm [12].

2.5 Synthesis of earlier research

The articles included in this chapter have presented empirical evidence on what impact agile principles and practices have had on software development in other organizations. We have synthesized the presented impacts into thirteen areas: Communication, Project Visibility, Internal software documentation, Knowledge sharing, Coordination, Job satisfaction, Pressure and stress, Architecture, Employee capabilities, Employee voluntary turnover, Productivity, Software quality and Project Success rate. We have also concluded the suggested impact of agile principles and practices in each area, based on the empirical evidence presented in the articles. The suggested impacts are shown in Figure 1.
The impacts are divided into effects and outcomes, where the distinction is that an effect refers to any impact that could be observed in the software development project while an outcome refers to an impact that is believed to be associated with business value for Ericsson. It is not excluded that some of these may lack business value and that others may have it.

The agile practices related to each area reported by the articles are presented. The agile practices are presented as either contributing or counteractive. For example: iteration planning is reported to improve formal and informal communication and is therefore presented as contributing, while whole team is presented as counteractive as it is reported to create communication problems [7].

![Figure 1. Synthesis of the empirical evidence on what impact agile principles and practices have on software development.](image-url)
The arrows in Figure 1 represent relationships between effects and outcomes.

A – Improved informal communication reduces the need for documentation [7] [21].

B – Improved formal communication increases the understanding of requirements and features [7]. Improved communication improves developer learning [24]. Open communication forums enable creation and sharing of knowledge [23].

C – Improved communication improves project visibility for team members, customers and managers [7].

D – Improved communication is an important contributor to improved coordination [7].

E – The reduced emphasis on documentation may lead to long term knowledge loss [10].

F – Higher job satisfaction leads to lower employee voluntary turnover [6].
3 Conceptual framework

In this chapter a conceptual framework is constructed from the theoretical basis presented in Chapter 2. The conceptual framework describes the believed impact of agile principles and practices on the large-scale software development of Project A and Project B. The conceptual framework is used as a guide for the data collection and data analysis.

3.1 From theory to conceptual framework

The purpose of this thesis is, as stated in section 1.4, to answer the question:

*What impact does the use of agile principles and practices have on the large-scale software development projects Project A and Project B?*

To answer this question, we chose to regard the evidence found in the literature review as suggestions on what impact the use of agile principles could have in Project A and Project B, as well as what differences may be found between the projects. However, it’s not considered feasible to investigate all suggested impacts inside this thesis. Consequently we reduced the theoretical background into a conceptual framework. The conceptual framework constitutes the theoretical point of departure for the empirical study and does as such describe what areas we will study in Project A and Project B. The reduction was made mainly for practical reasons, but also based on interest by the authors. For further discussion on this see chapter 5.2.3. The chosen areas are Internal software documentation, Knowledge sharing, Project visibility, Stress and pressure, Coordination effectiveness, Productivity, Software quality and Project success rate.

3.2 The conceptual framework

The conceptual framework is illustrated in Figure 2. It contains the five possible effects and three possible outcomes of agile principles and practices that will be investigated in Project A and Project B. In the following sections we describe each of the areas in the conceptual framework and how they will be approached in the empirical study.
The arrows in Figure 2 represent possible relationships between effects and outcomes proposed by the thesis authors. The existence of these relationships will be investigated in the empirical study.

A – Less internal software documentation is believed to increase productivity since less time spent on documentation means that more time can be spent on value creating activities.

B – A higher coordination effectiveness is believed to increase productivity thanks to decreased unnecessary overhead.
3.2.1 Description of conceptual framework - Effects

3.2.1.1 Less and insufficient documentation
Similar to [10] we will study software documentation in terms of the employee’s perception of documentation and investigate to what extent the members of Project A and Project B find the documentation in their projects sufficient. In the definition of software documentation we include all documents concerning the product and the development process, such as requirement specifications, design specifications, technical documentation of program code, trouble reports and change requests.

We argue that the reduced focus on internal software documentation could increase individual developer productivity - the less time spent on documentation work, the more time can be spent on value-creating activities – and will therefore investigate the effort spent on producing documentation in Project A and Project B.

3.2.1.2 Facilitated knowledge sharing
Knowledge sharing will be studied from the perspective to what extent team members can and do help each other. We will investigate if the project members perceive that the agile practices retrospectives, daily meetings and iteration planning facilitate their knowledge sharing, as suggested by [23]. We will further investigate if demos contribute, as [8] believes and also if open office space does, as [24] believes. As proposed by [23] and [33], inter-team knowledge sharing could be facilitated by letting external members attend agile meetings; we will investigate if the project members of Project A and Project B help each other across team borders. The possible problem of letting external members attend team meetings, highlighted by [11] won’t be investigated. Neither will trust or the effect iterative development may have on the awareness of capabilities [23], as these areas are considered out of scope of this thesis.

To investigate if the agile practice whole team has had any effect on the daily work of the employees, as proposed by [37], we will investigate what activities, such as coding, testing, and requirement specification, the employees perform frequently.

3.2.1.3 Increased project visibility
We define project visibility according to the visibility definition from Encyclopedia Britannica [26] as the capability for the project’s status to be readily noticed by both internal and external stakeholders. Since we lack the possibility to survey external stakeholders and don’t consider it feasible to survey the entire internal organization, the focus will be on project status as perceived by the project members of Project A and Project B. We will investigate if agile practices, namely iteration planning, open office space, demo, task boards or continuous integration increase project visibility in the studied projects, as suggested in [7]. We will further examine if agile practices improve visibility for both project members and project management in Project A and Project B.

3.2.1.4 Increased pressure and stress
The literature review indicated that the demos can increase stress [8], and that sign up can increase the pressure on individual team members to complete a task they have signed up for [23]. Daily meetings can increase both pressure and stress [11]. We also argue that iterative development can cause negative pressure and stress. It should be noted that stress and pressure under the same circumstances can vary greatly from person to person. Something that causes stress for one
person need not cause stress for someone else [38]. Stress is, in this way, subjective. Therefore, we will study the perceived stress and pressure in Project A and Project B on a personal level. However, we will also study the overall effects on a project level

3.2.1.5 Effective coordination
To study coordination in Project A and Project B we will use the theory on coordination in agile software development projects proposed by Strode et al [8] and the definition of coordination effectiveness given by the same authors in an earlier article [39]. In this definition the authors have identified eight components of coordination effectiveness – five implicit and three explicit. These are illustrated in Figure 3. We will investigate if the agile practices information radiators and daily meetings contribute to increasing the awareness on what other team members and other teams are doing and when (implicit coordination), as Strode et al [8] propose. We will also investigate if open office space and whole team are contributing to coordination effectiveness, as Strode et al also suggest.

![Figure 3. The definition of coordination effectiveness given by Strode et al. [39]](image)

### 3.2.2 Description of conceptual framework - Outcomes

#### 3.2.2.1 Increased productivity
We will investigate the productivity in Project A and Project B to see if the use of agile principles and practices has led to higher productivity, as seen in other organizations (for example [30] [31] [32] [33]). The productivity will be studied from two perspectives: perceived productivity and measured productivity. The intention is to investigate if the effect of productivity being perceived as higher, but not measured as higher reported by [4] is present here. Regarding individual agile practices we will investigate if iterative development contribute to an increase in productivity as proposed by [4], and also if whole team and open office space do, as proposed by [25].
Additionally we argue that productivity of a software development process can be investigated from three different perspectives: project, team and individual. On a project level the communication and coordination between teams are important factors for the productivity of the development process. On a team level the coordination between team members is of importance, especially the time left for value-creating activities when formal and informal communication and coordination is done. The possible impact of documentation on an individual developer’s productivity has been discussed earlier. We further argue that the ability to focus may possibly influence the individual developer's productivity. The emphasis on communication and collaboration in agile software development may disturb the individual work of a developer, especially the agile practice *open office space* [7] [24]. These three perspectives of productivity will be investigated by analyzing coordination both intra-team and inter-team with productivity in mind and by examining the developers’ ability to focus.

3.2.2.2 Higher quality
One interesting finding from the literature review is that software quality was perceived as higher, but not measured as higher in terms of defects introduced, in a project that transitioned from a plan-driven process to Scrum [4]. Another finding from the literature review was that there is a positive correlation between code quality and the amount of agile experience of the organization [28]. *Demos* has also been found to raise the software quality by Strode et al [8]. On the other hand, Lee and Xia proposed that *empowerment* could have an indirect negative effect on software quality [25]. This raises the question on what is true for Project A and Project B. Software quality will thus be studied both as perceived quality, and measured quality, in order to see if this phenomenon occurs in these projects as well.

3.2.2.3 Higher project success rate
Organizations with experience from agile software development have been found to have a higher project success rate [28]. Specifically, the practices of *iterative and incremental development* and *empowerment* seem to have a positive effect on a project’s on-time [4] and, in the latter case, on-budget completion [25]. However, *daily meetings* and *collective ownership* can be considered as contributing to maintaining escalating commitment [11] which can have a negative effect on project success rate, as discussed in 2.3.8. We will investigate the success rate, in terms of on-time and on-budget completion in Project A and Project B and analyze if agile principles and practices affect it.
4 Research questions

This chapter presents a list of questions that will be investigated in Project A and Project B through the empirical study. The questions are derived from the conceptual framework described in chapter 3, and a set of questions is used for every proposed effect and outcome.

Prerequisites
- What agile principles and practices are implemented in the development processes of Project A and Project B?
  - To what extent has the practice of whole team made the members of Project B perform activities from different functional areas?

Less and insufficient documentation
- Has the use of agile principles and practices had an impact on the use of internal software documentation in Project A and Project B?
  - How do the project members perceive the amount of internal software documentation in Project A and Project B? Is there a difference?
  - Do the project members believe that internal software documentation is important and/or that it can be replaced by face-to-face communication in Project A and Project B? Is there a difference?
  - How much effort do the members of Project A and Project B spend on producing software documentation?

Facilitated knowledge sharing
- Is knowledge more easily shared in Project B, compared to Project A, thanks to the use of agile principles and practices?
  - To what extent do the project members in Project A and Project B help each other to solve problems? Is there a difference?
  - To what extent do the project members in Project A and Project B feel that they can help each other to solve problems? Is there a difference?
  - Is there inter-team learning in Project A and Project B? Is there a difference?
  - Do \{daily meetings, iteration planning, retrospectives, demo, whole team\} have a positive effect on knowledge sharing in Project A and Project B? Is there a difference between the projects?
  - Do open office space have a positive effect on knowledge sharing in Project B?

Increased project visibility
- Has the use of agile principles and practices made Project B more visible to team members, other teams and project management, compared to Project A?
  - Do the team members, other teams and project management feel that they are aware of the teams’ status in Project A and Project B?
  - Do the team members and project management feel that they are aware of the project’s status in Project A and Project B?
  - Do \{demos, continuous integration, iteration planning, task boards, iterative development\} increase project visibility in Project A and Project B? Is there a difference between the projects?
  - Do open office space increase project visibility in Project B?
Increased pressure and stress
- Has the use of agile principles and practices increased the pressure and stress employees feel in Project B, compared to Project A?
  o What is the perceived level of pressure and stress in Project A and Project B? Is there a difference?
  o What is the measured level of pressure and stress in Project A and Project B? Is there a difference?
  o Do \{demos, iterative development, sign-up, daily meetings\} cause pressure and stress in Project A and Project B? Is there a difference?

Effective coordination
- Has the use of agile principles and practices increased coordination effectiveness in Project B, compared to Project A?
  o What is the perceived level of implicit coordination effectiveness in Project A and Project B? Is there a difference?
  o What is the perceived level of explicit coordination effectiveness in Project A and Project B? Is there a difference?
  o Do \{task boards, daily meetings, open office space, whole team\} increase coordination effectiveness in Project A and Project B? Is there a difference?

Increased productivity
- Has the use of agile principles and practices increased productivity in Project B, compared to Project A?
  o How do the project members perceive the productivity in Project A and Project B? Is there a difference between the projects?
  o Is there a difference in measured productivity between Project A and Project B?
  o Do the members of Project B feel that they lose focus because of the open office space?
  o Does effort spent on internal software documentation affect productivity in Project A and Project B?
  o Does coordination effectiveness affect productivity in Project A and Project B?

Higher quality
- Has the use of agile principles and practices increased the quality of the software produced in Project B, compared to Project A?
  o How do the project members perceive the software quality in Project A and Project B? Is there a difference between the projects?
  o Is there a difference in the measured software quality between Project A and Project B?

Higher project success rate
- Has the use of agile principles and practices made Project B more successful than Project A?
  o What is the on-time completion rate in Project B and Project A? Is there a difference between the projects?
  o What is the on-budget completion rate in Project B and Project A? Is there a difference between the projects?
5 Research design

The issue of knowing and understanding the effects of the agile principles and practices applied inside Project A and Project B was addressed with a qualitative, multiple-case study. In this chapter this research method is explained together with its eligibility. Each step of the research process is described in detail. The chapter concludes with a discussion on validity, reliability and ethic considerations.

5.1 Research method – a multiple-case study

A case study is a qualitative research method that “investigates a contemporary phenomenon in depth within its real life context” [40, p. 18]. It can include one or more cases and needs multiple sources of information, such as observations, interviews and documents [41]. The gathered data can be both quantitative, such as surveys, and qualitative, such as observations; this is not what distinguishes a case study from other methods [40].

Case study as a research methodology is more commonly used in social research fields, such as education, business or sociology [41] [40], but is also suitable for software engineering research [37]. Runesson and Höst [37] have, in their article on case studies in software engineering research, compiled definitions of a case study given by other authors from both the software engineering research and other research fields, among them Yin [40] and Creswell [41], and state: “[..] the key characteristics of a case study are that 1) it is of flexible type, coping with the complex and dynamic characteristics of real world phenomena, like software engineering, 2) its conclusions are based on a clear chain of evidence, whether qualitative or quantitative, collected from multiple sources in a planned and consistent manner, and 3) it adds to existing knowledge by being based on previously established theory [..]”

A multiple-case study includes more than one case. The different cases can either be selected because they are believed to produce the same results (literal replications) or contradictory results due to the change of one or more conditions (theoretical replications) [40].

5.1.1 Rationale for the chosen method

Creswell [41] describes that a qualitative research method is suitable when a complex, detailed understanding of the issue in its natural setting or context is needed. He further states that a qualitative research method can be chosen when it’s not possible to separate the studied phenomenon from its context. In this thesis, it’s considered that a deep understanding of the addressed issue will make the result more valuable to Ericsson, for whom the study is conducted. It’s also not considered possible to separate the people and the technology that constitute the development processes of interest from their context: the Ericsson organization, the physical workspace and the interactions among the people and between people and technology. Consequently it was decided to use a qualitative research method.

One qualitative research method is the case study [41]. Runesson and Höst [37] argue that the complex interactions between people and technology typical in a software development process is the common application of a case study and that case studies therefore are suitable for software engineering research. They have studied the case study methodology used inside other research fields and adapted them to the special needs of software engineering research [37]. Creswell [41] states that a case study is suitable when a case with clear boundaries is identified and an in-depth understanding and/or comparison of the cases are wanted. When the cases are the primary unit
of analysis and not subunits of a larger unit of analysis, such as an organizational unit, a holistic multiple-case study is suitable [40] [37].

In this thesis the impact of agile principles and practices on a project level is the primary unit of analysis. The two software projects identified are also considered to have clear boundaries in resources as well as time and a deep understanding of the differences and similarities between the projects is, as previously stated, considered beneficial. We therefore concluded that a multiple-case study is suitable for this thesis. Throughout the thesis the case study methodology synthesized and adapted for software engineering research proposed by Runesson and Höst was used, with additional help from the guidelines for general case studies proposed by Yin and Creswell.

5.2 Description of the research method

5.2.1 Step 1 - Planning and definition of scope
A case study is a flexible research methodology where iterations and change of plans are welcomed and encouraged, however careful planning has to be undertaken throughout the study to enable the establishment of the clear chain of evidence necessary for sound conclusions [37]. To achieve this, an initial plan of the thesis was formulated and continuously updated as the study advanced.

The general purpose that was given in the beginning was to study the effects of the agile transformation in PDU X and produce some evidence thereof. To make this practically possible with the present time limitations and to generate the concrete evidence requested by Ericsson, it was decided to study two different projects and make a comparison between them. For a case study in software engineering research a software development project is a suitable case [37]. The software development projects were chosen to show different perspectives of a development process: one plan-based and one agile approach with different amounts of implemented agile practices, as opined suitable by both Creswell [41] and Runesson and Höst [37]. Yin [40] opines that a case study of only two cases should strive for literal replications, i.e. cases that are expected to give the same results. However, the author also acknowledges the fact that issues of access and other arrangements can make the case study researchers choose theoretical replications also in case studies of two cases. Yin further states that, if the empirical evidence gained from such a study support the propositioned differences this is a strong start towards confirming a theory.

5.2.2 Step 2 – Literature Review
To collect knowledge and theories about the impact on agile principles and practices on software development and as such be able to build upon what others already had done, a literature review was conducted. The aim of the literature review was to establish a basis for the case study and generate propositions of possible effects the use of agile principles and practices could have in Project A and Project B. The importance of building a theoretical framework is also emphasized by Yin [40]. Yin states that the theoretical basis should be used throughout the case study for planning the empirical study, formulating a research strategy and generalizing the results.

To strengthen the conclusions of the case study it was chosen to conduct the literature review as a Systematic Literature Review. Systematic Literature Reviews are used for “identifying, evaluating
and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest” [42].

The systematic literature review was conducted in accordance with the guidelines proposed by Kitchenham and Charters for software development research [42]. The intention of the review was to answer the three questions:

1. What empirical evidence is there on the effects of using agile principles and practices in industrial software development?
2. Which of these effects apply to large-scale, industrial software development?
3. Which agile principles and practices are contributing to the evident effects?

After some initial trials with different search phrases a good balance between number of results and content was found with the search phrase “agile software development” AND (effect* OR impact*) in the Inspec database. Results were limited to articles written in English, Swedish and Spanish and published between 2005 and 2012. The language criteria was chosen because of practical matters and the publication dates because of Dybå and Dingsøyr’s [3] thorough literature review of empirical studies showing the limitations and benefits of agile software development methods up until 2005. The search was done for matches in keywords and resulted in an initial result set of 281 hits. After using the Inspec tool for removing duplicates, configured to favor articles with an abstract and from the Inspec database, 229 hits were left (out of which there were still some duplicate articles). We investigated the title and abstract of these 229 hits and selected 77 articles for further investigation. This selection was made on a beforehand decided selection criteria and the same criteria was used consistently by the two authors, any ambiguous articles were discussed. The selection criteria included requirements of empirical study, an industrial setting and/or formulation of model to measure the effects of agile software development methods. The selection criteria are presented in appendix 12.1.1.

The conclusions of the 77 articles were read and discussed. After a selection based on conclusions, 35 articles were selected for a full reading. This selection was made on the same criteria as before, with the deeper understanding of relevance gained from reading the conclusions. Finally after the full reading, 16 articles were selected for data extraction. This selection was based mainly on quality of the research method, but studies were also excluded when the findings of the study after the full reading no longer was considered relevant for the thesis. No standardized quality assurance method was used but the authors’ judgments; the research method of the included studies should be sufficiently described and discussed. For those of these 16 publications where only a summarized article was found by the Inspec database, data was extracted from the full article instead of the summary. For a list of the included studies see appendix 12.1.4. The data extraction protocol is presented in appendix 12.1.2.

To define the agile practices investigated throughout the thesis the list from Agile Alliance [1] was used, as this can be considered to be the industry standard. Also a literature review on agile practices was used as a complement to this list. The books on Extreme Programming [14] and Scrum [15] written by its founders were used to describe each methodology.
The earlier studies on the use of agile principles in Ericsson were found by a combination of database search and verbal communications with employees at Ericsson.

5.2.3 Step 3 – Definition of scope of empirical study

The literature review resulted in a list of suggestions to what impact the use of agile principles and practices could have on Project A and Project B. The effects were synthesized into thirteen different areas by us. Although all of the suggested impacts are of interest, limitations in time and data sources made it necessary to choose only a subset of these for the empirical study. The chosen areas were Internal software documentation, Knowledge sharing, Project visibility, Stress and pressure, Coordination Effectiveness, Productivity, Software quality and Project success rate. The study of the possible impacts on Communication, Job satisfaction, Architectural problems and Employee capabilities in Project A and Project B was left for future studies. A motivation for the excluded areas follows.

Communication is a central part of agile software development and explicitly mentioned in both the agile manifesto and the principles behind the manifesto [13] [43]. But at the same time it’s an abstract concept, which wasn’t considered meaningful to investigate properly without the possibility of observation or qualitative interviews and was therefore not included in the empirical study. Since Ericsson does a survey of the job satisfaction every year, although not it’s relationship with agile methods, this area was considered to be of less interest. Architectural problems were considered to be out of scope of this thesis. The creation of employee capabilities wasn’t considered a feasible area to study properly within the limited time frame, since not much change can be expected to be captured during a time frame of a couple of weeks.

For each of the studied areas a set of research questions were formulated. These research questions were then used to plan the data collection and formulate the survey questions.

5.2.4 Step 4 – Data collection

In a case study, data triangulation, i.e. using multiple data sources, is vital to reach sound conclusions [40]. We collected empirical data through a survey of project members, a study of internal documents, the gathering of metrics and personal contacts with key members of the two projects, as well as PDU X.

5.2.4.1 Survey of project members

Qualitative interviews may be more commonly associated with case studies than surveys. But surveys are stated to be a good choice when wanting to study a phenomenon in width and describe or explain it [44] and the purpose of the thesis is to describe and explain the impact of agile principles and practices on the large-scale software development projects Project A and Project B. It is also common to investigate the relationship between the survey units in some way [44], which made us believe a survey was suitable for investigating if there are any differences between Project A and Project B caused by their implementations of agile principles and practices. There is however a limit to the number of questions that could be asked and details that could be collected with a survey and therefore a limit to the achievable in-depth understanding of a phenomenon [44]. Care of this was taken in the formulation of the research questions.
The survey invitations were distributed via e-mail to a sample of 120 members of Project A and all members of Project B, on two separate occasions during fall 2012. The surveys were open for one week (Project A) and two weeks (Project B) and hosted on a paid version of SurveyMonkey. The week prior to the invitation a pre-notification e-mail was sent and three days (Project A) and one week (Project B) after the initial invitation a reminder was sent. The variables included in the questionnaire were measured using an interval scale, to facilitate analysis and interpretation of the results as suggested by [44].

Two different questionnaires were used in the survey, one for members of Project A, and one for members of Project B. Both questionnaires included an open-ended question that gave the respondents the possibility to comment on the ways of working in their project. The main difference between the two questionnaires was that some questions regarding agile practices that aren’t in use in Project A was removed from Questionnaire A. The survey of Project A was responded by 34 persons, this is a response rate of 28% and the number of respondents corresponds to 8.1% of the project members. The survey of Project B was responded by 52 project members, which corresponds to 43% of the total project members.

5.2.4.2 Internal documents
During the writing of the thesis we had access to several internal, Ericsson documentation systems. The information gained from these documents was incorporated in the analysis of the information gained from other sources. Data from internal documentation was primarily used for project background descriptions and general understanding of the software development in the two projects.

5.2.4.3 Personal communication
Throughout the entire thesis period several unstructured interviews over phone and video was conducted with different members of Project A, Project B and PDU X. Notes were taken during these interviews, but no material was transcribed. Contact has also been held via e-mails and instant messages. The data gained from this communication has been compiled and presented together with the data gained from other sources.

5.2.4.4 Metrics
To investigate not only the project members perception of the impact of agile principles and practices but also if the effects could be seen in the product and process performance we gathered different metrics. A metric is a combination of two or more measurements. The metrics are summarized in Table 2 on page 35. The following metrics were gathered:

**Pressure and stress**
We argue that reported overtime could be seen as an indicator of pressure and stress; more overtime indicates a more strained time plan and more pressure and stress. Reported overtime in Project A and Project B was therefore collected and compared. The overtime hours were put in relation to the total project hours for each project, to get a quota that allows for a comparison between Project A and Project B. The distribution of overtime hours was also studied, i.e. if

---

3 Based on a total number of project members of 420.
almost everyone in the project was working overtime, or if the majority of the overtime hours were put in by a small portion of project members.

**Productivity**

The common definition of productivity is the ratio between input and output of a process. The challenge is, though, how to choose the most suitable representatives for output and input [45]. In the articles included in the literature review productivity was measured in different ways. In some articles, for example [30] and [31], it was measured as the ratio between Lines of Code (output) and developer hours (input). Others, such as [4] measured the productivity of sub processes, for example testing, by measuring the cycle time of defect fixing. Yet others, such as [33], measured productivity only in terms of perceived productivity.

We wanted to measure productivity with a combination of metrics as well as the project members’ perception of productivity, to best investigate if agile principles and practices have had an impact on the productivity in Project A and Project B. However, due to practical reasons and differences in age, size and architecture of Product A and Product B it wasn’t considered feasible to use metrics based on lines of code. We therefore focused on functional productivity and calculated the average work item cycle time in the two projects. Additionally the productivity of the defect fixing process in Project A and Project B was investigated by calculating the cycle time of trouble reports (TR) and bugs. The cycle times in Project B were calculated for the entire project and for only the cross-functional teams. This separation was unfortunately only possible for trouble reports and bugs in Project A and it wasn’t possible to calculate the work items cycle times for the entire Project A as these calculations were based on data extracted from a system that only was used in parts of Project A. The system was considered the only possible source for this data.

**Software quality**

The most definite measure of software quality is, in our opinion customer feedback or defects found when the customer uses the software. But, since Product B isn’t released to customers yet this approach wasn’t possible and we had to instead study indications of software quality. Therefore we used defect density as a quality metric in the same way that Li et al. [4] did. Defect density is thus defined as adjusted found defects divided by project hours. Li et al [4] used implementation hours in the denominator, but this data wasn’t available for us why we used total project hours instead.

Test case pass rate was also used as an indicator of code quality, by dividing the number of passed tests with the number of executed tests. To be able to better analyze the test case pass rate, test coverage was also gathered, where it was available.
Project success rate

To measure project success rate, the same two metrics studied by Lee and Xia [25] was used: a project’s on-time, and on-budget completion. For on-time performance the measurement used was: How many of the features originally intended to be released at a specific date were actually released at that date?

For the on-budget aspect, a comparison was made between the original project budgets and the outcome this far.

Table 2. An overview of the metrics that were collected and analyzed

<table>
<thead>
<tr>
<th>Outcome/effect</th>
<th>Metric</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pressure and stress</strong></td>
<td>Overtime density</td>
<td>$\frac{Overtime\ hours}{Total\ project\ hours}$</td>
</tr>
<tr>
<td></td>
<td>Overtime concentration</td>
<td>$\frac{Persons\ reporting\ overtime}{Total\ project\ members}$</td>
</tr>
<tr>
<td><strong>Productivity</strong></td>
<td>Functional productivity</td>
<td>Average work item cycle time</td>
</tr>
<tr>
<td></td>
<td>Defect fixing efficiency</td>
<td>TR and bug cycle time</td>
</tr>
<tr>
<td><strong>Software quality</strong></td>
<td>Defect density</td>
<td>$\frac{Number\ of\ defects}{Total\ project\ hours}$</td>
</tr>
<tr>
<td></td>
<td>Test case pass rate</td>
<td>$\frac{Passed\ tests}{Total\ tests}$</td>
</tr>
<tr>
<td></td>
<td>Test coverage</td>
<td>$\frac{LOC\ executed\ by\ test\ suite}{LOC}$</td>
</tr>
<tr>
<td><strong>Project success rate</strong></td>
<td>On-time release of features</td>
<td>Fraction of features released when originally planned</td>
</tr>
<tr>
<td></td>
<td>Budget performance</td>
<td>$\frac{Budget\ outcome}{Planned\ budget}$</td>
</tr>
</tbody>
</table>

5.2.5 Step 5 – Data analysis

The point of departure for the analysis was that the two projects, Project A and Project B differ sufficiently in their approach to software development and are sufficiently similar in other aspects that the effects and outcomes of using agile principles and practices can be studied by comparing the two projects.

Regarding similarities between the two projects, they both operate in the same corporate context – the PDU X product development site in Europe. Further, we have no reason to believe that the people working in the two projects differ considerably in education level or experience, in fact many of the people currently working in Project B previously worked in Project A. Additionally do the two projects develop a product with the same functional requirement – Product A and Product B should be able to perform the same tasks and is to be sold to similar customers. Finally, do the two projects have the same technical preconditions: they use the same implementation languages, as well as similar test and development environments and support systems.
Regarding differences, we argue that the difference in project size between the two projects doesn’t affect our results appreciably. They are both large-scale projects with hundreds of software professionals and more than ten different teams. It can be assumed that for the primary unit of investigation in both the survey and the interviews, individual developers, it is hard to differ between 100 other project members or 400 – 100 is still a number large enough to not be able to keep track of everyone.

That Product A is a legacy product, which has been under development for many years and with different development methods, while Product B was designed from scratch with agile development in mind could however, potentially affect the results. This has been handled with research methodology, we have focused on effects closer related to the people than product age and architecture – such as knowledge sharing, pressure and stress – and also by taking this into consideration while deciding what metrics to use, e.g. no metrics on lines of code were used since this wasn’t considered feasible to compare between the two products.

In short, we focused our research approach on comparing areas between the two projects where project organization and development methodology are the primary differences. The proposition was consequently that the effects and outcomes of agile principles and practices would differ between the two projects. The intention was to analyze each project as a separate case and then do a cross-case analysis to further capture the impact of agile principles and practices on large-scale software development.

The data was analyzed using the pattern-matching logic, which “compares an empirically based pattern with a predicted one” [40, p. 136]. Our predicted patterns were derived from the theoretical framework and presented in the conceptual framework. In consequence we went through the effects in the conceptual framework one by one and combined the data from the different sources to determinate if the predicted effects are present in Project A and Project B. This approach, to base the data analysis on propositions generated from a theoretical framework is the preferred one, according to Yin [40].

We chose to first describe and analyze Project A and Project B separately and then do a cross-case analysis, as this approach is advocated by Yin [40]. The main purpose of the separate case descriptions and analyses were to provide insights to the project management of each project and the main purpose of the cross-case analysis was to respond to the thesis’ purpose: to investigate the effect of implementing agile principles and practices in large-scale software development projects. Our proposition was that the suggested effects of agile principles and practices would be evident in Project B and not in Project A – or at least more evident in Project B, and thus match the patterns in the theoretical framework.

5.2.5.1 Survey data analysis
According to Yin [40] a survey conducted within a case study should use the same instruments and analysis tools as a regular survey. The surveys were analyzed with the software IBM SPSS 20.

The first step in the survey data analysis was to prune the material from individual survey answer sets that were empty or only contained responses to the background questions; e.g. included less
than 10 answers. For Survey A, 27 of the initial 34 responses and for the Survey B, 48 of the initial 52 responses were included in the analysis. Additionally, for each question, cases with missing values were excluded, why the analysis of some questions includes less than 27 or 48 cases. To check that the samples of the two projects were representative for the entire projects, questions about background variables were included in the questionnaires. The distribution of the answers to these questions was used in a comparison with the known facts about the projects to evaluate how representative the samples were.

The primary goal with the survey was to see if the effects found in the literature review were perceived as present by the members of Project A and B. This was primarily done by investigating the mean and standard deviation of the questions that corresponded to an effect. The interval scale [44] used for these questions ranged between one and seven, where the endpoints represented ‘extreme values’. Accordingly a mean value above four was considered an indication that the investigated effect was present in the project.

According to [46] it’s important to evaluate the normality of the variables, since normality is an important assumption for many statistical tests, e.g. the t-test, and especially so for samples sizes smaller than 50. The normality of Survey A and Survey B was assessed with the Shapiro-Wilk test. A variable was considered normally distributed for a Shapiro-Wilk significance above $\alpha=0.05$. For Survey A, approximately half of the variables could be considered normally distributed, but for Survey B none of them could. In consequence the non-parametric method Mann-Whitney u-test was used instead of the t-test to investigate differences between the two projects. The Mann-Whitney U-test for independent samples was used on the null hypothesis that the means of the two projects are equal. The null hypothesis was rejected for an asymptotic significance (p) smaller than $\alpha=0.05$ and thus confirming a statistically significant difference in the answers between Project A and Project B.

It was possible to compare the survey answers in the two projects since the two questionnaires included the same questions and both surveys received enough respondents. Our opinion is that a significant difference in the answers provides evidence on the effect of agile principles and practices in large-scale software development.

Unfortunately the number of respondents in each functional role category was too small in both surveys to investigate differences between different roles in Project A and Project B, which was initially planned.

5.2.6 Step 6 - Conclusions and discussion
Conclusions on the impact of agile principles and practices were made individually for each case – Project A and Project B – as well as for the combination of the two projects. The purpose of the cross-case conclusions was to answer the purpose question presented in section 1.4.

Finally a discussion on the transferability of the results was also made, since this discussion was considered valuable to Ericsson.
5.3 Validity and reliability

Two important aspects to consider when doing a case study are the validity and the reliability of the research method. Validity concerns the ability of the research method to measure what it is intended to measure and reliability concerns the ability of the research method to resist the influence of external factors at the time of measurement, i.e. the ability of the method to give the same result on repeated measurements [44].

In the beginning of this chapter we motivated why a case study is a valid methodology to investigate the effect of using agile principles and practices in a large-scale software development project. But, according to Runesson and Höst [37] there are several ways to further improve the validity of a case study. One of them is to use data triangulation, which we have done. Another is to maintain a case study protocol. We haven’t maintained a case study protocol as defined by the authors, but we kept a project plan and project report (this report), which we continuously updated. A third way to improve validity according to both Runesson and Höst [37] and Lekvall and Wahlbin [44] is to have the research reviewed by peer researchers. Our supervisors and examiner has continuously reviewed our work, research methodology and survey questionnaires, as has members of Project A and Project B.

In surveys there is always a relatively high risk of non-respondents [44] and this can threaten the validity of the survey results [47]. In order to increase the response rate, we used some of the methods proposed by Manzo and Burke [47]. We sent out a pre-notification e-mail the week before the survey, carefully formulated the survey invitation e-mail and sent out a reminder the week after the invitation. The primary goal of these e-mails was to establish multiple contacts with the respondents as that is the best way to increase survey response rate [47].

To increase the survey response rate it is also important to pay careful attention to the header and content of an invitation e-mail to a web-based survey, while the sender address is of less importance for the survey response rate [47]. We sent out the survey invitations using the internal Ericsson e-mail client from one of the thesis authors e-mail address. The invitation e-mail header included a plea for help, but didn’t mention the survey or agile software development methods. The e-mail header was formulated this way to avoid creating response bias, as suggested by [47]. Also, the invitation email was carefully reviewed by both thesis authors. Additionally we tried to increase response rate by creating a buy in from project managers. However, we had no control of how, or even if, this support was executed and we clearly stated in our communication that participation in the survey was voluntary to avoid skewed responses.

Another threat to the validity of surveys is that the questions may not capture the real perceptions of the survey respondents [44]. We based our survey questions on our research questions. The research questions were in turn was derived from the conceptual framework, which was constructed from the theoretical basis, alternatively on the same theories the article authors had used. The articles had been chosen for their high quality. Nevertheless, it is practically impossible to determinate how to formulate a survey question objectively and a subjective judgment is therefore necessary [44]. One way to do this is to let insightful people review it [44]. We let several persons with domain knowledge, including the project management of Project B, our two supervisors and the examiner comment on the questionnaires. We also let a
test person with similar work assignments as the project members of Project A and Project B fill out the questionnaire under our observation and encouraged him to “think aloud”. That way we could confirm that the questions were interpreted as intended. To avoid misunderstandings due to language, as English is not our first language, the survey questions were also proof-read by a native English speaker.

Surveys, especially web-based ones, offer little control over the interview situation, why there might be low reliability in the survey results [37]. It is unknown, but possible that the respondents were tired, disturbed or not motivated when filling out the survey. This would have had a negative influence on the reliability of the survey results. The survey was designed to take 10-15 minutes to complete. We could not see any signs of respondents becoming tired after a certain amount of time (i.e. there was no emerging pattern that separated answers to questions towards the end of the survey). However, even though participation in the surveys was voluntary the surveys yielded a quite high response rate, especially in Project B. We can also note that many of the respondents filled out the last, open-ended question. We consider this to be an indication that the motivation of the project members to participate in the survey and express their opinion on the ways of working in their project was relatively high.

Regarding the personal contacts with project members via instant messages, e-mails and telephone we also had little control of the interview situation. It is certain that during the time of the study a lot of other things were going on in Ericsson and it has sometimes been hard to get answers and data. But, we made sure to formulate questions carefully and validate the citations and data used with the projects. The project descriptions presented in chapter 6 and 7 have also been validated by the projects.

The reliability of the survey was strengthened by the fact that the same exact questions were used in all questionnaires, with the exception from project specific names. Some of the questions in the survey was of an abstract nature, and might have represented concepts that were new to the respondents. However, there was always the option of answering “no opinion” or “don’t know” to these types of questions. We also believe that the questions, although abstract in some cases, were quite reasonable and of a kind that anyone working in software development should be able to answer. It is also our opinion that the questions in the survey questionnaires weren’t of a particularly sensitive nature, and we see no reason for the respondents to give incorrect answers due to the nature of the questions, especially when considering that the survey was taken anonymously.

The survey used a seven point scale, since such a scale has a good balance between resolution and reliability of the results. A higher resolution (e.g. a nine point scale) would have made smaller differences visible, but the reliability would have been lower due to the larger amount of randomness in the answers.

There are several considerations to take when studying metrics. Runesson and Höst [37] point out that the organization under study might have collected metrics with a different purpose than the researcher. For example, information needed for the study may be missing in the metrics, or the metrics may contain a lot of information that is not interesting for the researchers. This is
true for this study, since the metrics collected by Ericsson in most cases are not collected with the same purpose as this study. There are special “agile transformation and optimization” metrics in use within PDU X, but those were not used in this study. The reason for this is in part that those metrics aren’t final and haven’t been collected for very long, but also that the agile transformation and optimization metrics aren’t measuring what we are looking for. Also, the purpose of this thesis is in part to provide a perspective that is different from what Ericsson already knows about its organization.

The problem that the metrics are gathered with a different intent than the one in this thesis, and that the metrics might not contain sufficient data, is handled by not using the metrics on their own as a single source of information for analysis, but in combination with the information gathered from the survey and from personal contacts. This hopefully lessens the risk of misunderstanding and misusing the metrics and thereby strengthens the validity and reliability of our study.

5.4 Research ethics

Runeson and Höst state in their guidelines for case studies in software research that case studies in this field often include dealing with confidential information in organizations [37]. This was the case in this thesis and certain ethical considerations were taken.

Since the proposition for the thesis was given by Ericsson it is considered that the organization as a whole has given the informed consent of participation in the study. Nevertheless, the consent from Project A and Project B was also sought actively by contacting the project managers. Furthermore each individual participant was asked to consent their participation in the study before conducting any interviews and survey participation was voluntary. The purpose of the study was always communicated beforehand, as stated necessary in [44]. To encourage people to express their opinion freely, the survey responses were collected anonymously and the names of the interview objects were made anonymous in the report. No culturally sensitive questions were considered to be included in the survey or the interviews.

Feedback of results is important for the long-term trust, according to Runesson och Höst [37]. To achieve this we chose to present our results and analysis to the projects and give them the opportunity to ask questions and present their view not only on the data, but on our analysis, as proposed by the authors.

Confidential and sensitive information about Ericsson and its software processes was removed from the report publicized through Linköping University, to protect the company. The project and product names were replaced with fictive ones and some project and product details were removed. However, the details necessary to understand the processes and create a context are reported. We find the excluded names and details of little value to the external reader and believe that this doesn’t affect the quality of the report.
6  Case A: The plan-based project

The chapter begins with a detailed description of Project A. It then describes what agile principles and practices are implemented in Project A and presents the results from the collected metrics and the survey of project members. Additionally, the effect of the implemented agile practices is analyzed and case specific conclusions are made.

6.1  Empirical findings

6.1.1  Data sources

The data presented in this chapter comes from several different data sources. These are listed in Table 3. Data from all sources was combined and presented together to give a complete picture of the effect of the use of agile principles and practices in Project A. A summary of the survey questionnaire and the results of all questions are presented in appendix 12.2.

Table 3. Data sources used in the case description of Project A.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal documentation</td>
<td>ID</td>
</tr>
<tr>
<td>Personal communication with Agile Coach, PDU X</td>
<td>AC</td>
</tr>
<tr>
<td>Personal communication with Support function manager, PDU X</td>
<td>SFM</td>
</tr>
<tr>
<td>Personal communication with Quality Engineer, PDU X</td>
<td>QE</td>
</tr>
<tr>
<td>Project A specific</td>
<td>S-A</td>
</tr>
</tbody>
</table>

6.1.2  Case description

Project A is the project currently responsible for designing, developing and maintaining new versions of Product A - throughout the years several previous projects have had the same responsibility. Project A’s main development is situated at the PDU X head quarter and the project employs approximately 420 people, of which some are external consultants working on-site. There are also some 480 consultants working in a different part of the world, but they are not considered as part of the project in this thesis.

Product A supports radio networks and is actually an entire product family. The product bundle consists of a software package with additional hardware and back-up packages. The software is packaged in bundles with basic features for a certain network type, and a large set of optional features. The main supported functions are: Element Management, Fault Management, Software and Hardware Management, Performance Management Administration and Configuration Management. In its current form Product A has been under development since 2001 with over ten released versions so far and is still under development. For Ericsson it’s an important software product that is used by hundreds of network operators around the world.

Product A was originally created by combining three different software products, one for each of the three network types: GSM, 3G (WCDMA) and core. Since then support for more networks has been added along with a large range of features. The result is a large, complicated and tightly coupled software product made up of many software components. According to the Agile Coach [AC], the dependency management of these components is hard, with features spanning over a
large amount of components, and the integration of them is a complicated and error-prone process requiring extensive documentation.

### 6.1.3 The software development process in Project A – a waterfall approach

A new version of Product A is released two to four times per year and the development of each release follows a traditional waterfall model [17] with separate, consecutive steps for system analysis, design, implementation, integration and system test. However, the actual development done in the design and implementation phase is done in increments where features are individually advanced through the stages and delivered to integration, feature and early system test every month. Some design and implementation teams use Scrum.

Different subprojects within Project A are responsible for the different steps in the process. *Systems* is responsible for system analysis and requirement specification, *Design* for system design, *Features* for development and *Integration & Verification* is responsible for integration and system and feature testing. The basic unit testing is done in parallel to development, by *Features*. Between each step a handover between the different subprojects is necessary.

There are several challenges with this separation of responsibilities and phases, describes the Agile Coach [AC]. The handovers are handled through an extensive set of standardized documents and creates a risk for misunderstandings. It’s also a time-consuming activity to create the necessary documentation. To have the system analysis as a separate step in the beginning, separated from the development, makes it hard for the system analysts to determine the system impact of a requirement or change. It’s also difficult to capture and define an entire system at once. The handover misinterpretations and misunderstandings are intensified throughout the process and result in defects in the software. Since verification and testing is separate from development the defect reports are often delivered late – the Integration & Verification department is often several development iterations behind – and in general feedback cycles are long in the Product A development process. And again, these test results need to be delivered in the form of documents, creating more risk for misunderstandings and more time spent on documentation. The consequence is defects piling up and causing several iterations to be spent entirely on fixing them. All this has led to large volumes of change requests (CR) and trouble reports (TR) and lots of things to redo. At the end of every release cycle there has been a large trouble report backlog that has taken a lot of effort to take care of. This has in turn led to a lot of overtime, according to the Agile Coach [AC]. Throughout the years the development of Product A has also had limited interaction with customers and users, causing even more change requests.

Nevertheless, Project A has strived to become more agile for several years and has implemented some agile principles and practices. These are discussed in detail in section 6.1.5. There is however, no end-to-end implementation throughout the whole project. Inside the project there is an opinion that architectural and organizational characteristics limit the extent to which an agile software development process can be fully implemented.
6.1.4 Description of Survey A respondents

To investigate the effects of agile principles and practices in Project A a survey of the project members was conducted. The survey was sent to a random sample of 120 project members and 34 of them responded. 27 of these responses were complete enough to analyze (see sections 5.2.4.1 and 5.2.5.1 for more details on the survey analysis methodology). Four of the 27 respondents stated that they’re not longer working in Project A, but that they had earlier experiences from the project and was therefore included in the analysis anyway.

24 of the respondents stated that they are employed by Ericsson, two that they are working in Project A as external consultants and one didn’t respond to this question. On average, the survey respondents had been working for Ericsson during 8.36 years (standard deviation 6.95). The respondent with the longest experience had been working for Ericsson for 23 years and the one with the shortest for one year. The respondents had been working in Project A or any of Product A’s previous development projects between 1 and 13 years, with an average of 4.84 years (standard deviation 3.40).
Approximately half of the respondents – 13 persons – stated that they were part of a Scrum team and the other half – 14 persons – that they were not. The respondents’ experience with agile and non-agile software development is illustrated in Figure 5. In general the survey respondents had many years experience from non-agile software development but not as many from agile software development.

The survey respondents identified their primary work assignments with different functional roles. These are illustrated in Figure 6. Most of the respondents identified themselves either Designers or Developers (8 persons each) or Project Managers (7 persons). No one stated that they worked with Configuration Management. 63% (17 respondents) reported that one functional role corresponded best with their primary work assignments, 33% (9 respondents) reported two corresponding roles and one person reported that three functional roles corresponded with his or her primary work assignments.
6.1.5 What agile principles and practices are implemented in Project A?

Although Project A is largely a traditional, plan-based project we have identified the use of several agile practices in the whole project, or parts of the project.

Some teams in Project A have fully implemented the practices of *backlog, daily meetings, iteration planning, task board* and *sign up*. These practices are all, except for *sign up*, key elements in the Scrum methodology. Further, *Continuous integration* is used by some parts of Project A, but there is no end-to-end implementation for the whole project. The same is true for *collective ownership*; it is used only in parts of the project.

Project A has adopted an *iterative development* approach, but only within the traditional phases of its plan-based development model. During the design and implementation phase, work on individual features is done *incrementally*. A couple of teams have fully implemented an end-to-end use of *iterative and incremental development* according to the Scrum methodology and these are also fully cross-functional, but they constitute such a small part of the project that for the purpose of assessing the adoption of agile practices in Project A we decided not to consider them. Therefore it is considered that Project A has done a partial implementation of the *iterative and incremental development* practice and no implementation of the practice *whole team*.

There are two major releases every year, which can be considered as a partial implementation of frequent releases. *Demos* are scarce, but are held in the project to a certain extent. *Information radiators* are used in Project A, but not very widely. The main information radiators are the *task boards*. *Refactoring* is used occasionally, but there is no formalized plan for when it should be done, why it isn’t considered to be implemented in Project A.

---

**Figure 6. The functional role descriptions of the Survey A respondents.**

<table>
<thead>
<tr>
<th>Functional role</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Analyst/Architect</td>
<td>8</td>
</tr>
<tr>
<td>Product Owner</td>
<td>2</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>2</td>
</tr>
<tr>
<td>Developer</td>
<td>8</td>
</tr>
<tr>
<td>Configuration management</td>
<td>0</td>
</tr>
<tr>
<td>Administrative</td>
<td>1</td>
</tr>
<tr>
<td>Project manager</td>
<td>4</td>
</tr>
<tr>
<td>Tester/QA</td>
<td>5</td>
</tr>
<tr>
<td>Designer</td>
<td>8</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
</tr>
</tbody>
</table>
Project A teams sit together in an open office space. Retrospectives are done after every sprint in Project A, in those teams that work in sprints. The project teams are tightly connected to the line organization, and are not very autonomous. They cannot be said to be empowered. Further, neither pair programming, on-site customer, sustainable pace nor simple design is used in Project A.

### Table 4. Implementation of agile practices in Project A.

<table>
<thead>
<tr>
<th>Agile practice</th>
<th>Implementation status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backlog</td>
<td>Partial</td>
</tr>
<tr>
<td>Collective ownership</td>
<td>Partial</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>Partial</td>
</tr>
<tr>
<td>Daily meetings</td>
<td>Partial</td>
</tr>
<tr>
<td>Demos</td>
<td>Partial</td>
</tr>
<tr>
<td>Empowerment</td>
<td>Not in use</td>
</tr>
<tr>
<td>Frequent releases</td>
<td>Partial</td>
</tr>
<tr>
<td>Information radiators</td>
<td>Partial</td>
</tr>
<tr>
<td>Iteration planning</td>
<td>Partial</td>
</tr>
<tr>
<td>Iterative and incremental development</td>
<td>Partial</td>
</tr>
<tr>
<td>On-site customer</td>
<td>Not in use</td>
</tr>
<tr>
<td>Open office space</td>
<td>Full</td>
</tr>
<tr>
<td>Pair programming</td>
<td>Not in use</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Not in use</td>
</tr>
<tr>
<td>Retrospectives</td>
<td>Partial</td>
</tr>
<tr>
<td>Sign up</td>
<td>Partial</td>
</tr>
<tr>
<td>Simple design</td>
<td>Not in use</td>
</tr>
<tr>
<td>Sustainable pace</td>
<td>Not in use</td>
</tr>
<tr>
<td>Task board</td>
<td>Partial</td>
</tr>
<tr>
<td>Whole team</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

6.1.5.1 The project members’ perception of the agile practices in their project

In general, the respondents of Survey A [S-A] were satisfied with the implemented agile practices in their project. Almost all practices gained in the survey a mean value above four on a scale from 1=’Not at all’ to 7=’Completely’. The only exception was refactoring, which gained a mean value of 3.89 (standard deviation 1.94). The survey respondents somewhat agreed in their level of satisfaction of the implemented agile practices – the standard deviations were between 1.6 and 2.4. Open office space was the practice with the least agreement (standard deviation 2.35). For all practices except open office space, between 15% and 25% of the respondents stated that they had no experience of that practice. This agrees with the earlier description that agile practices are implemented in parts of Project A, but not in the entire project. In the open-ended survey one respondent wrote:
“Great fun, great satisfaction[SIC]^4 with achieved results.”
[System Analyst/Architect and Project Manager, Project A]

The most popular agile practice was the daily meetings (mean 5.18; standard deviation 1.65), followed by iterative development (mean 5.10; standard deviation 1.83) and iteration planning (mean 5.00; standard deviation 1.88). The overall ways of working in Project A received a mean value of 5.41 (standard deviation 2.31) on a ten point scale.

![Figure 7. The survey respondents' satisfaction of agile practices in Project A.](image)

Several survey respondents commented in the open-ended question on the fact that Project A has only implemented agile principles and practices to a limited extent and that the agile way of working may not be fully compatible with the Project A organization. For example, one survey respondent felt that the agile practices give results in proportion to how much effort and to what extent they are implemented:

“In our team we use a basic version of Agile/Scrum. Each and every piece of the puzzle yields proportionally to the investment, ie how seriously and with what quality we execute them (eg demos, velocity follow-up).”
[Project Manager, Project A]

On average the survey respondents had performed 1.67 of the six stated activities ‘Requirement specification’, ‘System design’, ‘Implementation’, ‘Writing tests’, ‘Running tests’ and ‘Refactoring’

---

^4 All quotations are presented as we received them in the survey answers. [SIC] is used to mark errors done by the survey respondents.
during the last seven days. All activities were more or less equally often performed; 10 persons had been running tests, nine persons had been implementing, eight persons had been specifying requirements, seven persons had been writing tests, six persons had been doing system design and five persons had performed refactoring. Four persons hadn’t been performing any activities or chose to not answer the question (there was no “none” option).

6.1.6 Internal software documentation in Project A

Since Project A follows a plan-based project model with up front system analysis and requirement specification, as well as handovers between different phases, there are a large number of documents to control the process and the produced software. The documentation needed is in itself specified in several different documents and specifications [ID].

On average the survey respondents [S-A] perceived the overall amount of internal software documentation in Project A as just right; the question received a mean of 3.95 (standard deviation 1.40) on a seven-point scale from 1=‘Way too little’ to 7=‘Way too much’. The median value was 4.00. However, 36.4% of the respondents gave the overall amount a score above four, which means that the amount is too high. No respondent perceived the overall amount of internal software documentation as ‘Way too little’, while one respondent thought it was ‘Way too much’. One respondent wrote in the open-ended question:

“Too much unnecessary documentation and no documents with required details. If no documents, where to keep information when people are moving and leaving the organization.”

[System Analyst/Architect, Project A]

The survey respondents [S-A] further stated that they think internal software documentation is important for Project A (mean 4.92; standard deviation 1.64) and that it can only be replaced with face-to-face communication to a limited extent (mean 4.04; standard deviation 1.68).
There were five kinds of documentation items that were rated as ‘Slightly too much’ by the respondents in Project A: Trouble reports, Process documentation and Requirement specifications, Change requests and Work packages. At the same time, they thought the amount of Technical documentation of code, Code comments and Design specifications was ‘Slightly too little’. Nine respondents had no opinion on the amount of code comments in Project A and seven respondents on the amount of technical documentation of code. The survey respondents’ opinion about the amount of the software documentation items is illustrated in Figure 8.

58.3% of the survey respondents [S-A] had spent 15 minutes or less daily on producing internal software documentation the last 30 days. 41.7% had spent five minutes or less daily. 8.3% had spent one hour or more. The time spent daily on producing internal software documentation is illustrated in Figure 9.

Figure 8. The survey respondents’ opinion about the amount of different software documentation items in Project A.
6.1.7 Knowledge sharing in Project A

In the survey [S-A] the members of Project A stated that they to a moderate extent get help with problems related to their project work. The respondents reported that they, compared to finding the solution themselves, get help from team members with an average frequency of 4.77 (standard deviation 1.45) and from members of other teams with an average frequency of 3.80 (standard deviation 1.63) on a scale from 1='Never' to 7='Always'. The survey respondents further felt that their team members have the necessary capabilities to help them with the project work (mean 5.38; standard deviation 1.42). They also definitely felt aware of what capabilities their team members have (mean 5.42; standard deviation 1.39).

According to the survey [S-A], the members of Project A feel that several agile practices contribute to knowledge sharing in their project. The respondents reported that daily meetings help their team to solve problems together (mean 4.67; standard deviation 1.68). They further reported that iteration planning (mean 4.33; standard deviation 1.65) and demos (mean 4.32; standard deviation 1.84) give them insights about the project they wouldn’t have otherwise. Furthermore, the respondents stated that they have gained insights from other functional areas than their own from their team members (mean 4.27; standard deviation 1.86). But they didn’t think that the retrospectives give new insights to the same extent (mean 3.70; standard deviation 1.69).

6.1.8 Project visibility in Project A

In Project A the members feel, according to the survey [S-A], that they are aware of their team’s status (mean 5.60; standard deviation 1.29) but not really of the other teams’ statuses (mean 3.32; standard deviation 1.70) and the entire project’s status (mean 3.64; standard deviation 1.87). The response scale had seven points from 1='Not at all' to 7='Completely'. In total six persons or 24% of the survey respondents stated that they weren’t aware at all of the other teams’ statuses, while only one respondent wasn’t aware at all of their own team’s status.
Nevertheless, the survey respondents felt that the agile practices that are implemented in the project are important for their visibility of the goals and plans of Project A. Most important was continuous integration (mean 5.82; standard deviation 1.40), closely followed by iteration planning (mean 5.60; standard deviation 1.31), demos (mean 5.59; standard deviation 1.84) and task boards (mean 5.55; standard deviation 1.73).

6.1.9 Pressure and stress in Project A
The survey respondents stated that they to a moderate extent feel negative pressure and stress by their work in Project A (mean 3.59; standard deviation 1.62). They also felt somewhat to moderately stressed by the iterative development (mean 3.28; standard deviation 1.57) and to finish what they have signed up for during a sprint, i.e. by the agile practice sign-up (mean 3.19; standard deviation 1.42). All questions had a seven point response scale from 1='Not at all' to 7='Extremely'.

Regarding the demos the survey respondents disagreed quite much – the standard deviation was 1.91 – but on average the respondents were moderately stressed by them (mean 3.82). The respondents also disagreed much whether they felt a negative pressure to report results on every daily meeting or not (standard deviation 1.95), but on average they didn’t feel that pressure to a large extent (mean 2.94).

6.1.9.1 Overtime density and concentration
For 2012 the overtime density in Project A was 3.40%, i.e. for every 100 project hours 3.40 overtime hours were reported. The overtime was reported by 30% of the project members.

6.1.10 The level of coordination effectiveness in Project A
There are approximately 420 members of Project A and as earlier described separate departments handling separate phases in the software development process. Consequently there is a need for effective coordination between the project members and teams in Project A. According to the Support Function Manager [SFM] a lot of overhead is spent on achieving effective coordination in Project A.

The support function manager [SFM] describes that Project A additionally has many external dependencies and work outsourced, which can lead to difficulties. Especially if the external organizations or units are in a different stage of adopting agile methods. The same issue is highlighted in the open-ended survey question by one respondent, who wrote that the fact that software code is spread among Ericsson and external consulting teams, sometimes located offshore, blocks the progress and causes problems:

“These [SIC] is absolutely no transparency at source code and requirements level between [Consulting company] and Ericsson. To many time [Consulting company] teams and Ericsson ones are working on the same piece of code with to coordination between them and because of these to many hours are spent fixing coding issues, fixing bugs in one sprint and in the next one the bugs are back because of some changes... and that is due to no communication/transparency. Is very hard to have 3+ agile teams working on the same base code adding that some of the teams are offshore (working hours do not fit, long communication, multiple persons touching in the same time code functionality)”

[Designer and Developer, Project A]
On a direct question the members of Project A stated that they feel that they can coordinate effectively within the project [S-A]. The question received a mean of 4.80 (standard deviation 1.73) on a seven point response scale from 1='Not at all' to 7='Completely'.

The model proposed by Strode et al [39] (see Figure 3, section 3.2.1.5) was also used to determine the coordination effectiveness in Project A, as coordination effectiveness is considered a fairly abstract concept. This model includes two aspects of coordination effectiveness: implicit coordination and explicit coordination, where each aspect consists of several components. Every component corresponds to a specific question in the survey, using the same seven point scale from 1='Not at all' to 7='Completely'. The aggregated mean of the components in the model was 4.64 (standard deviation 1.15), which can be seen as a quantification of the coordination effectiveness in Project A. The calculated level of implicit coordination effectiveness, i.e. the mean of the corresponding components, was 5.04 (standard deviation 1.11) and the calculated level of explicit coordination effectiveness was 4.15 (standard deviation 1.63).

Regarding the individual components of the coordination effectiveness model [39], the majority of the components received a mean value above four. Highest scores were received by the questions on work task coordination: the respondents felt that they definitely know what task they should be working on right now (mean 6.19; standard deviation 1.06) and when that task should be finished (mean 6.08; standard deviation 1.02). The components/questions concerning other project teams received a mean lower than four. The respondents didn't, for example feel that they know what the other teams are doing (mean 3.35; standard deviation 1.70) or that they are aware of the entire project’s status (mean 3.64; standard deviation 1.87). However the respondents did feel that they know what capabilities their project members possess (mean 4.22; standard deviation 1.73), but still not to the same extent as they know the capabilities of their team members (mean 5.42; standard deviation 1.39).

The survey respondents [S-A] felt that agile practices help them coordinate effectively. They definitely felt that the daily meetings help them know what their team members are doing (mean 6.17; standard deviation 1.10). Also scrum boards help the survey respondents know what their team members are doing, but not to the same extent (mean 5.33; standard deviation 1.80).

### 6.1.11 Productivity in Project A

The survey respondents would rate the productivity on all levels of Project A as high on a scale from 1='Extremely poor’ to 7='Extremely good’. They gave their own productivity the highest score (mean 5.63; standard deviation 1.08), followed by the productivity of their team (mean 5.22; standard deviation 1.22) and the overall project productivity (mean 4.88; standard deviation 1.18). 70.4% of the respondents gave their own productivity a rate of 6 or 7, while only 26.9% gave the overall project productivity the same score.

In the open-ended survey question one respondent, who is part of a Scrum team, commented on productive time lost on meetings in agile software development:

“I think agile [ways of working] taking lot of time on planning but actual deliverables (output) are less” [System Analyst/Architect, Project A]
Another respondent expressed an opinion that too much time is spent on keeping updated with development environment and test servers:

“In agile way of working the team members should take care of one issue at a time and finish it and deliver value. The reality is not quite like this: Many many hours are spend[SIC] upgrading and keeping up to date the local development environment and test servers. In agile way of working a developer/designer should not be de-focused from his development work with things like jumping/installing test servers, fixing crashes on test/dev environment which are independent of his or his team work etc."

[Designer and Developer, Project A]

A third respondent thought that the project members lack necessary competence and capabilities to perform the work efficiently:

“Many people in the project don’t have the needed competence and capability to do the job and this has negative effect on the productivity.”

[Designer, Project A]

6.1.11.1 Functional productivity

Several teams in Project A use an IT system for tracking issues, bugs, trouble reports and user stories. The system is also used for communicating support issues between teams, i.e. inquires and faults are reported trough the system instead of being reported by e-mail.

The IT system supports the work breakdown process encouraged by Scrum and several types of work items are present. Four common work item types are epics, stories, tasks and sub-tasks. Epic is the most high-level one followed by story, task and sub-task. An epic corresponds to a high-level product requirement and a sub-task corresponds to what the teams commit to do during a sprint. Table 5 shows the cycle times for the different work items. The cycle time is calculated as the number of calendar days between the date a work item is created in the system and its resolution date. The cycle times were calculated for all work items present in the system, i.e. from work items handled in the system from October 2010 up until January 2013. Please note that the system is not used by all Project A teams or even a majority. It’s neither used uniformly across the different teams and only used by agile teams in Project A, why this data merely represents the agile subset subset of Project A, not the whole project.

Table 5. Work item cycle times in Project A.

<table>
<thead>
<tr>
<th>Work item cycle time</th>
<th>Number of items</th>
<th>Average cycle time</th>
<th>Maximum cycle time</th>
<th>Minimum cycle time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epics</td>
<td>4</td>
<td>117 days 12 hours</td>
<td>154 days 4 hours</td>
<td>80 days 15 hours</td>
</tr>
<tr>
<td>Stories</td>
<td>486</td>
<td>59 days 19 hours</td>
<td>258 days 20 hours</td>
<td>0 days 0 hours</td>
</tr>
<tr>
<td>Tasks</td>
<td>255</td>
<td>23 days 3 hours</td>
<td>282 days 5 hours</td>
<td>0 days 0 hours</td>
</tr>
<tr>
<td>Sub-tasks</td>
<td>1713</td>
<td>12 days 7 hours</td>
<td>139 days 18 hours</td>
<td>0 days 0 hours</td>
</tr>
</tbody>
</table>

We see that the average cycle times decrease with each break-down level. For the most high-level work items, epics, it is 117 days and 12 hours, while it for sub-tasks is only 11 days and 18 hours. The longest cycle time is present though for a story: 282 days and 5 hours. This means that the story was open for more than three quarters of a year. The shortest cycle times of 0 days and 0
hours are most likely reflections of work items that where already done before being inserted into the system.

Since Project A and its predecessors has been going on for a very long time, some tasks that are very similar for every release has been highly automated and take very little time to complete, according to a Support Function Manager in PDU X [SFM].

6.1.11.2 Defect fixing efficiency

Project A handles defects in the software and the software documentation through documents called trouble reports. The defects are typically found by the Quality Assurance or Integration & Verification department and then reported to the developers. All trouble reports are reported and monitored using an internal Ericsson IT system. The report documents, handovers and standardized way of treating trouble reports are creating a lot of overhead, according to a quality engineer [QE].

The overall trouble report cycle time was calculated as a measure of defect fixing efficiency. During one completed release cycle of Product A, which is a time period of approximately one year and one month, 5418 trouble reports were closed. The average cycle time was 31 days from the original submission to the resolution date. The longest cycle time was three quarters of a year, 264 days, and the shortest was less than a day. Trouble reports that were cancelled or rejected were not included in the calculation, unless they were re-opened. Trouble reports that were rejected but re-opened later had their cycle time calculated from the original creation date to the final finish date, since that kind of delay in handling a trouble report should be considered part of the time it takes to actually take care of it.

A subset of the trouble reports is denominated bugs, which signifies that these reports concern defects that should be fixed in the current release [QE]. 4014 of the studied trouble reports counted as bugs, and their average cycle time was also 31 days.

Bugs and trouble reports are also, to some extent, handled through the same IT system as regular work items (see section 6.1.11.1). The IT system separates “Bug” and “Trouble report”. This system is primarily used by the teams in Project A that work in an agile way. The cycle times for the bugs and trouble reports handled through this system, i.e. for the agile teams, is shown together with the trouble report cycle time for the entire Project A in Table 6. The cycle times for the agile teams were calculated from all data present in the system, i.e. from October 2010 up until January 2013.

Table 6. Cycle times for bugs and trouble reports in Project A.

<table>
<thead>
<tr>
<th></th>
<th>Number of items</th>
<th>Average cycle time</th>
<th>Maximum cycle time</th>
<th>Minimum cycle time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Bugs</strong></td>
<td>4014</td>
<td>31 days</td>
<td>264 days</td>
<td>0 days</td>
</tr>
<tr>
<td><strong>Agile team Bugs</strong></td>
<td>477</td>
<td>16 days 4 hours</td>
<td>126 days 3 hours</td>
<td>0 days 0 hours</td>
</tr>
<tr>
<td><strong>Project TR's</strong></td>
<td>5418</td>
<td>31 days</td>
<td>264 days</td>
<td>0 days</td>
</tr>
<tr>
<td><strong>Agile team TR's</strong></td>
<td>100</td>
<td>21 days 18 hours</td>
<td>247 days 0 hours</td>
<td>0 days 0 hours</td>
</tr>
</tbody>
</table>
6.1.12 Software quality in Project A

Software quality in Project A was measured with quality metrics, as well as the project members perception of software quality.

On average, the survey respondents [S-A] rated the quality of the software produced in Project A as 4.72 (standard deviation 1.46) on a scale from 1='Extremely poor' to 7='Extremely good'.

6.1.12.1 Defect density

Defect density was collected from internal tools and documentation. During 2012, 18.95 defects were reported for every 1000 project hours in Project A.

6.1.12.2 Test case pass rate

One round of system stability testing in Project A was performed during 22 weeks in the beginning of 2012. During the period, the accumulated number of executed test cases rose from 36 in the first week to 1260 in the last one. The pass rate over the weeks is shown in Figure 10. The pass rate at the end of system testing was 88.5%.

According to a quality engineer at PDU X, system stability testing is a very important subset of the system tests done in Project A. The tests show how the system reacts under a heavy load [QE].

![Accumulated system stability test pass rate](chart)

**Figure 10. Accumulated system stability test pass rate**

6.1.13 Project A success rate

6.1.13.1 On-time release of features

Project A uses a policy of restricting customer use of features if the feature isn’t ready or hasn’t been verified in a live network. Restrictions can be set on part of a feature, the whole feature, a specific configuration, hardware or network element support. Restricted features are not
Features in Product A are divided into basic features and optional features. Basic features are always provided with an installation of Product A, while optional features are made available on customer demand.

### Table 7. Restricted features in one release of Product A.

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Reason for restriction</th>
<th>Not done</th>
<th>Not verified</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td></td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Optional</td>
<td></td>
<td>6</td>
<td>12</td>
<td>18</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>9</td>
<td>14</td>
<td>23</td>
</tr>
</tbody>
</table>

### Table 8. Number of features in one release of Product A.

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>28</td>
</tr>
<tr>
<td>Optional</td>
<td>81</td>
</tr>
<tr>
<td>Total</td>
<td>109</td>
</tr>
</tbody>
</table>

At the time of the release of one version of Product A in mid 2012, 23 features out of 109 had some kind of restriction. This corresponds to 21.1%. Most of the restrictions were due to insufficient verification. 9 features, or 8.3% of all features, were restricted due to not being completely developed.

6.1.13.2 Budget performance

In Project A, from July 2011 to May 2012, the project budget was exceeded by 2.2%. There was a very small deviance from budget until October 2011, but the budget size during those months was also quite small. From November 2011, the budget outcome differed from the target by 13% to 39% every month but March, but not always by exceeding it. Unfortunately, there was no numbers available for June 2012 when system release took place.

According to the support function manager, a budget deviation of a few percent in a project is considered OK [SFM].

### 6.2 Data analysis – what impact does the use of agile principles and practices have in Project A?

6.2.1 Representativeness of the survey respondents

The survey respondents had on average been working for Ericsson for a long time and in Project A on average almost five years. This should mean that they are well accustomed to the ways of working in the company and project and good representatives for these. The respondents had also considerably more experience with non-agile ways of working than agile ones, why they also...
are good representatives for the traditional approach to software development. However, 7 out of 27 survey respondents said that at least one of their primary work assignments were project management. This could be an indication that the survey respondents aren’t very representative for Project A. It could also mean that Project A has a lot of project managers, which also would confirm the picture of Project A as a complex organization (with many managers).

6.2.2 Experiences and perception of agile principles and practices

Half of the survey respondents [S-A] were part of a Scrum team, half were not. This indicates that far from all members of Project A are organized according to Scrum and – considering that at least five survey respondents reported to have ‘No experience’ of all agile practices, except from open office space – are not using much of any other agile practice either. This supports the description of which agile practices are implemented in Project A. Out of those working in Project A, hardly any have more than a couple of years of experience working with Agile software development.

The survey respondents from Project A were, with some exceptions, in general satisfied with the implemented agile practices. This indicates that the practices are working as they should, and that they are useful. The only agile practice that the respondents weren’t satisfied with was refactoring, this could be explained by the complex nature of Product A - since the product has been under development for many years it can be that refactoring is very hard - but it shouldn’t be excluded that the routines considering refactoring needs some looking over.

The daily meetings were the most popular practice. It can safely be assumed that the reason for this is that it is a quite simple practice that is easy to implement and gain results from, even though few other agile principles and practices are implemented.

Nevertheless the survey respondents were not very satisfied with the overall ways of working in their project. This indicates that there are other factors that affect the daily work of the project members to a larger extent than the agile ways of working. The survey was for example sent during a period of layoffs in the company, something that possibly has affected the overall atmosphere in a negative way.

Most survey respondents stated that their primary work assignment correspond with only one of the functional roles listed in the survey. Survey respondents in Project A also performed 1.67 different activities during a time period of seven days. The different activities were meant to represent things that would be done by the same person in a team using the whole team practice, while they would be done by different people in a team with less cross-functionality. The low number 1.67 indicates that members of Project A are working somewhat cross-functional, but not to a broad extent. It confirms that the practice of whole team is not widely used in Project A.

6.2.3 Impact on internal software documentation

The survey [S-A] shows that the overall amount of internal software documentation in Project A is perceived as just right. But if we look at individual software documentation items the picture is not as uniform. The open answer that there was too much unnecessary documentation, and too little necessary documentation is mirrored in the answers that the amount of some kinds of
documentation is perceived as slightly too little, while the amount of other kinds of documentation is slightly too large.

It is also possible to see a pattern for what kinds of documentation the amount is seen as too little. Technical documentation of code, code comments and design specifications are all quite technical and close to the actual software code, while trouble reports, process documentation, requirement specification, change requests and work packages are more created for the work process than for understanding the product. The amounts of the “technical” categories are generally rated as “slightly too little”, while the amounts of other categories are rated as “slightly too much”. A possible reason for this is the handovers present in Project A - the system is being handed over between different departments that all need to understand what the others have done. Another explanation could be that process documentation, which is primarily used by the project management is subject to detailed requirements, while technical specifications are primarily used by developers and aren’t subject to the same standards. The technically complex nature of the product is a third possible explanation. Pikkarainen et al [7] reported the same pattern; the separate department of testers didn’t think that the Scrum team of developers produced enough technical documentation. Our study thereby supports the findings of [7] that documentation is an important part of communicating technical specifications between departments. Pikkarainen et al [7] did however, attribute the reason for the lesser amount of documentation produced to the development team’s use of Scrum. Some teams in Project A use Scrum, but we can’t with the available data say whether it’s this fact, or some other that’s responsible for the insufficient amount of “technical” software documentation in Project A.

Regarding the “non-technical” categories of software documentation in Project A, we think that this result is not very surprising. Project A is a traditional project with a lot of policy and steering documents, and it is not surprising that these documents are perceived as too many by the respondents.

A majority of the respondents claimed to spend 15 minutes or less daily on producing internal software documentation. Out of those, eight respondents claimed to spend no time at all on producing internal software documentation. On the other end of the scale, two people spent more than two hours daily on the same task. It is evident that the production of internal software documentation is done to a large extent by a small proportion of the project members, while many project members produce no documentation at all. An interesting issue to investigate would be to which extent different groups of the project members are reading and using the documentation. Is it everyone or just a small subset of the project?

On average the survey respondents [S-A] had worked for both Ericsson and Project A during many years. From this one could expect that there is a lot of tacit knowledge about the project and product lying with the project members. In consequence documentation should be important, especially in times like these when Ericsson undergoes organizational changes and layoffs. This is confirmed by the survey respondents who rated internal software documentation as important, and think that it can only to a limited extent be replaced by face-to-face communication. The issue is also high-lighted by one respondent in the open-ended survey question. We do therefore conclude that the agile belief that face-to-face communication is the
most efficient way to convey information [13] is not true for Project A. However, as different kinds of documentation received different ratings as to whether the amount was right, the same would likely be true if the project members were asked on which kinds of documentation can be replaced. We think therefore that a plan for what documentation to keep and which to reduce should be made before adopting agile principles and practices to a larger extent in Project A to not lose important, tacit knowledge.

6.2.4 Impact on knowledge sharing

In the survey data we can see that knowledge is shared in Project A. The survey respondents reported that they both get help with their project work and have gained insights from their team members.

The survey respondents in Project A got help from members of their own team more often than they got help from members of other teams, and they also felt that members of their own team have the necessary capabilities to help them. This is no surprise – since teams in Project A are specialized in different areas, it’s no wonder the best help most often can be found within the team.

The project members feel that some of the agile practices in use are contributing to knowledge sharing within the project, namely daily meetings, iteration planning, and demos. This supports the claims that these practices contribute to knowledge sharing, as made by McHugh et al [23] (daily meetings, iteration planning) and Strode et al [8] (demos). It should be noted that demos, iteration planning and whole team is only used in some of the teams in Project A. However, respondents do not, on average, agree that retrospective meetings give them insights about the projects they wouldn’t otherwise have gotten. This doesn’t necessary mean that the retrospective meetings aren’t working, but it’s a sign that they aren’t useful for knowledge sharing in this project. It’s interesting to see that retrospective meetings stand out in this sense, since most interviewees in [23] also placed little value in it.

We conclude that knowledge is shared in Project A, nevertheless one survey respondent wrote in the open-ended survey question that he or she think that the project members lack necessary capabilities and that the project productivity is negatively affected by this. We therefore suspect that there is more to the issue of knowledge sharing in Project A than we found in our study. It’s possible that parts of the project need to attain more knowledge. If so, implementing for example whole team could be a possible way to do this.

6.2.5 Impact on project visibility

Project members in Project A clearly felt that they are aware of the status of their own team. Since the teams in Project A are quite large (up to 90 persons), this result is a bit surprising. Although we have no information of the size of the teams of the survey respondents, i.e. if they belong to the largest or smallest teams of Project A. The survey respondents weren’t really aware of the status of other teams, or of the status of the whole project, which was less surprising. It is not strange that the members are less aware of the other teams or don’t feel that they know the status of the entire project, considering its size.

Survey results indicate that the studied agile practices (continuous integration, iteration planning, demos and task boards) all are quite important for the project members’ awareness of the project’s status,
as proposed by Pikkarainen et al [7]. This doesn’t have to mean that project visibility would have been worse without the use of these practices, but it shows that the practices are a useful way to ensure project visibility. Continuous integration showed to be the most important practice for project visibility in Project A. While continuous integration isn’t implemented in the entire project, the survey respondents were very happy with the implementation of this practice, why it can be concluded that where implemented continuous integration has led to positive effects.

6.2.6 Impact on pressure and stress
Survey respondents from Project A felt moderately stressed by their work in the project in general. There were however, not much overtime hours reported in the project. If the overtime had been evenly spread among all project members each project member would have needed to work one hour and 20 minutes overtime per 40-hour work week. This isn’t regarded as a very high amount. During the studied period the overtime was only reported by 30% of the project members, who then on average needed to work four and a half hours overtime each work week. This isn’t either considered a very high amount. This finding contradicts somewhat the description given by the Agile Coach [AC]. A possible reason for this could be that the majority of the overtime is executed by an even smaller amount than 30% of the project members and since these persons then are working a lot of overtime the general perception is that there is a large amount of overtime in the project. We have no further information about the distribution of overtime, if it is highly concentrated in certain time periods this could also affect the general perception of overtime in the project.

On average, the survey respondents were moderately stressed by the demos, but only a little stressed by iterative development, sign-up or daily meetings. Regarding sign-up and daily meetings, there was a wide variety in the answers, which means that the amount of negative stress caused by these practices varies much on a personal level. These findings somewhat agree with the previous works of Strode et al [8] (demos can increase stress), McHugh et al [23] (sign-up can increase pressure) and Moe et al [11] (daily meetings can increase both pressure and stress), although it’s also clear that these effects vary from person to person.

It’s clear from the answers that the implemented agile practices cause some negative stress and pressure, but it is not a very large amount. Further, it’s not known whether any alternate methods to accomplish the same task would cause more or less stress.

6.2.7 Impact on coordination effectiveness
On a direct question, members of Project A stated in the survey that they can coordinate effectively. The coordination effectiveness calculated with the model of Strode et al [39] also showed that the members of Project A can coordinate effectively. Since coordination and dependency challenges was mentioned by several persons we have spoken with, and Project A has only partially implemented the agile practices that Strode et al believe improve coordination effectiveness, this result was a bit unexpected. We would have expected to see the evidence of mentioned coordination challenges in the survey, but what we instead saw was that the coordination overhead is having the desired effect and the project is, according to the survey respondents effectively coordinated.
One possible explanation to the dubious results is that the coordination problems are mostly apparent on higher levels in the project organization, i.e. between teams, and thus not perceived by individual developers and testers to the same extent as to the managers we interviewed. However, there were a high number of project managers also among the survey respondents. Another possible explanation is the formulation of the questions and how each respondent interpreted the complex concept of coordination. In a software development project there are a lot of things that need to be coordinated – people, work items, product dependencies - and it can be that the survey respondents thought of work items when answering the questions, and the interviewed managers of people. It is also possible that the project members in general feel that they can coordinate effectively, but that they in fact can’t. If coordination effectiveness is split up into implicit and explicit measures, it is clear that the implicit coordination effectiveness is rated higher than the explicit effectiveness. The implicit coordination concerns whether the project members know the why, what, where, when and how of the project and the explicit coordination concerns whether the right things are at the right place, at the right time [39]. This result does then indicate that for Project A, the members are in general aware of the different components of the projects, “coordination in thought”, but that the actual work items aren’t coordinated as well, “coordination in practice”.

Although generally high, one question about implicit coordination effectiveness, “Do you feel that you know what the other teams are doing?”, received much lower scores than the rest of the questions. The low score here corresponds with the similarly low-scored project visibility question “To what extent do you consider yourself aware of the other teams’ status relative to their goals and plans?”. These answers show that the team members in general aren’t very aware of what other teams are doing, and what their status is. This could prove to be a challenge for inter-team coordination.

Another aspect that adds to the complexity of coordination in Project A is the coordination of external dependencies. This issue was brought up by both the Support Function Manager [SFM] and one survey respondent. Agile practices such as demos and having a Scrum Master, have been shown to help with the coordination of external dependencies [8], but these agile practices aren’t implemented in Project A.

In summary we conclude that the intra-team coordination effectiveness is satisfactory in Project A, but the project faces challenges in inter-team coordination. It also isn’t agile practices, but other coordination mechanisms that contribute to achieve the effective coordination in Project A.

### 6.2.8 Impact on productivity

The survey respondents from Project A ranked their own productivity as quite high, and gave a somewhat lower score to the productivity of their team. The productivity of the project overall receives an even lower score, but it’s still on the “good” part of the scale. The scores show that the productivity in Project A is perceived as good, but not perfect. The lower scores for other teams and the whole project could be caused, in part, by their low visibility. If you don’t know what the other teams and the rest of the project is doing, it’s possible to perceive their productivity as lower than your own.
The agile teams in Project A uses an IT system for tracking different work items such as epics, user stories, tasks and sub-tasks. The average cycle times are not surprisingly longer for higher level work items such as epics and user stories, and shorter for tasks and sub-tasks, but the individual items also vary greatly. For example, the longest cycle time of all items, 282, is held by a task, but the average cycle time of a user story is two months, while it is 23 days for a task. One reason for the large variation in cycle times is that work items may differ a lot in the required effort. Some tasks are done routinely and are highly automated, while other tasks require a lot of work to get done.

Project A also uses trouble reports, and the average trouble report cycle time in Project A was 31 days, about as long as a month. There is a vast spread in cycle time among trouble reports as well. Some trouble reports are handled on the same day, while others take the better part of a year to resolve. When singling out trouble reports that can be counted as bugs, the average cycle time is still 31 days.

It's also interesting to compare the cycle times of trouble reports for the agile teams compared to Project A as a whole. Cycle times are considerably shorter for the agile teams, regarding both general trouble reports and bugs. In this comparison, it’s clear that the productivity, measured in trouble report cycle times, is considerably higher in the agile teams. This gives an indication that productivity in agile teams are, in some cases higher than in traditional teams, as suggested by Ilieva et al [30], Moore et al [32], and even so in a large-scale context, as suggested by Benefield [33].

Some survey respondents from Project A brought up productivity in their answer to the open-ended survey question. While these answers aren’t necessarily representative for the project, or even the population of survey respondents, they still provide some interesting insights into the thoughts of some project members. One respondent wrote that the agile way of working uses a lot of time for planning but outputs less actual deliverables, which indicates that productivity in Project A is affected negatively by the adopted agile practices. This is however contradicted by what we found.

One respondent thought that his or her work is hindered by too many external dependencies and problems with the development environment. Using a more agile way of working wouldn’t remove the external dependencies, but it might provide a way to manage them. The task of the Scrum Master is for example to remove impediments of the developing team and facilitate a smooth progress of the work. The increased visibility gained with agile ways of working could also provide a way to find the problems earlier and alert the project and line management. Additionally, continuous integration and automated test and build systems could reduce the effort that developers need to put on performing such tasks and limit the extent of people that need to engage in them.

6.2.9 Impact on software quality
Software quality was rated fairly high in the Project A survey, but the score was far from perfect. This shows that the general picture among the survey respondents in Project A is that the quality of the software produced is good, albeit not perfect. The perceived good quality could be the
effect observed by Li et al [4], that quality is perceived as higher when adopting Scrum, although only a limited number of teams in Project A use Scrum.

Software quality in Project A was also studied with two quality metrics: defect density and test case pass rate. The defect density in Project A was 18.95 defects per 1000 hours. On its own, without a benchmark it is hard to judge this number as high or low.

The graph of system stability test pass rate shown in Figure 10 on page 55 shows that the pass rate declined to less than 50% during the first seven weeks of testing, but rose quite evenly up to 88.5% towards the end of the testing period. The low numbers during the period is to be expected, as tests are used to find and correct many kinds of errors, and there is some turnaround time between finding an error an fixing it in Project A. However, we don’t consider a pass rate of 88.5% at the end of testing to be good, especially when considering that the system stability tests is a very important system test subset. A possible explanation for the low final pass rate is that at least some of the failed tests are belonging to features that were restricted. The pass rate would then not really represent the product that was released.

Another possible reason for the low pass rate is the complex handling of defect reports, described by several project members. The long feedback cycles – many defects are found for the first time in the system testing phase and need to be reported back to the developers who then have come several steps further in the development – may cause that the defects aren’t found in time for them to be corrected before the release or before they multiply throughout the product. All things considered, it is our belief that a final pass rate below 90% is not good for any kind of testing, and indicates a software quality that is not too good.

6.2.10 Impact on success rate
At the time of release, less than 80% of the features of Product A were entirely ready. Most features were restricted due to not being sufficiently verified, but 8.3% of all features were restricted due to missing functionality. This is quite a large portion of the product, and it can’t be considered a very successful on-time release. Even though having features restricted at the time of release seems to be something usual in Project A and its previous instances, we still consider it remarkable to not release such a large portion of the product on time. To us, it’s a sign that there are challenges in the software development process in Project A.

From a budget perspective, Project A exceeded its budget by 2.2% from July 2011 to May 2012. 2.2% isn’t much to talk about, since it is within the range of a few percent that is considered OK (see 6.1.13.2). The monthly swings are more interesting. From November, the budget outcome differed from the target by 13% to 39% almost every month. The target was not always exceeded. This indicates that there is a difficulty to correctly estimate what the project costs will be, ahead of time in a monthly manner. It could potentially affect the future budget performance of the project, but it hasn’t done so during the studied period.

It is possible that these budget swings could be or could already have been made less severe, by the adoption of some agile principles and practices. The increased visibility brought by some practices could also have an effect on the possibility for management to grasp the whole project.
and counteract the swings, however we have not seen much implications in that direction. It should rather be said that the agile practices have not helped Project A to reach any high level of project success rate, showing no support for the effect suggested by Abbas et al [28]. It should however be noted that Project A has not adopted empowerment, which was suggested by Lee and Xia [25] to improve a project’s on-budget completion rate. It can neither be said that the practice of iterative and incremental development has brought Project A to a more on-time completion (as proposed by Li et al [4]), but then on the other hand, Project A is still in large parts a plan-based project.

### 6.3 Case A Conclusions

Project A has implemented agile principles and practices to a limited extent. We found that this has had an effect on some of the studied areas in the project, but that the expected effects have not happened in others. This shows that it’s possible to make a partial implementation of agile principles and practices and achieve a limited positive impact. We also found that the project members are satisfied with all studied practices, except refactoring.

Agile practices contribute to a moderate extent to knowledge sharing in Project A. Regarding project visibility; the agile ways of working raised the project members’ awareness of their own team’s status, but have failed to make them aware of the status of the entire project. The teams in Project A that have a higher level of adoption of agile principles and practices are more productive. Their trouble report and bug cycle times are shorter than the teams with a lower adoption level. Productivity in Project A is perceived as good by the project members.

The implemented agile practices in Project A have not caused negative pressure and stress to a large extent. We have neither seen any remarkable levels of overtime in Project A, as we had expected to be an effect of the plan-based approach.

Coordination within Project A is working well, but this has not been caused by the adaptation of agile principles and practices. Coordination is more effective within the development teams than between them. We didn’t see that the implemented agile practices have had an effect on the internal software documentation in Project A. The overall amount of internal software documentation in the project is rated as sufficient. However, there is a slight overabundance of documents concerning the development process, and slightly too few technical documents. It can be assumed that this is caused by the plan-based structure of Project A; that the handovers between departments and the complicated defect handling has led to a high dependency on technical documentation. Project A is believed to have a lot of tacit knowledge which leads us to believe that software documentation is important in the project to prevent loss of knowledge. We therefore have reason to doubt that the agile belief that face-to-face communication is the most efficient way to convey information [13], is true for Project A. We have neither seen any indications that the use of agile principles and practices has achieved a good software quality in Project A. Lastly, the limited extent with which Project A has implemented agile principles has had no impact on the overall project success rate, from what we can see.
The chapter begins with a detailed description of Project B. It then describes what agile principles and practices are implemented in Project B and presents the results from the collected metrics and the survey of projects members. Additionally, the effect of the implemented agile practices is analyzed and case specific conclusions are made.

### 7.1 Empirical findings

#### 7.1.1 Data sources

The data presented in this chapter comes from several different data sources. These are listed in Table 9. Data from all sources is compiled and presented together to give a complete picture of the effect of the use of agile principles and practices in Project B. A summary of the survey questionnaire and the results of all questions are presented in appendix 12.3.

<table>
<thead>
<tr>
<th>Data source</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal documentation</td>
<td>ID</td>
</tr>
<tr>
<td>Personal communication with Agile Coach, PDU X</td>
<td>AC</td>
</tr>
<tr>
<td>Personal communication with Support function manager, PDU X</td>
<td>SFM</td>
</tr>
<tr>
<td>Personal communication with Quality Engineer, PDU X</td>
<td>QE</td>
</tr>
<tr>
<td><strong>Project B specific</strong></td>
<td></td>
</tr>
<tr>
<td>Personal communication with Agile Manager, Project B</td>
<td>AM-B</td>
</tr>
<tr>
<td>Personal communication with Head Product Owner, Project B</td>
<td>HPO-B</td>
</tr>
<tr>
<td>Survey of project members</td>
<td>S-B</td>
</tr>
</tbody>
</table>

#### 7.1.2 Case description

The purpose of Project B is, according to the Agile Manager of Project B [AM-B] to design, develop and deploy a product, Product B, which in the long run can replace Product A. According to the Head Product Owner [HPO-B], Product B will gradually replace functionality in Product A, until Product A has actually become Product B. The hope is to be able to develop an equivalent product with fewer people in a more efficient way [AM-B].

Project B uses an agile project management method inspired by Scrum that consists of a flat organization where the entire development process is done in iterations and cross-functional teams are responsible for all phases in the process. There are also a number of supporting teams that handle the project management, continuous analysis, CI framework execution and so on.

The cross-functional teams include system analysts, designers and testers. Each development team also includes one Scrum Master and one Product Owner. The teams in Project B are responsible for features, compared to product/architecture layers in Project A. The Agile Coach [AC] describes that the stated goal is that every feature team in Project B should at the end of each sprint deliver business value. To enable this, each team in Project B requires vertical competence across all layers of the system architecture and also across the design, implementation and testing areas. Consistently all team members in Project B are now called developers, instead of system analysts, designers and testers. Another important organizational
change in Project B in comparison with Project A is that all team members report to the same line manager, called Agile Manager, instead of the several layers of hierarchy present in Project A. The Agile Coach [AC] explains that the rationale behind the organization change is the minimization of the error prone and time consuming coordination and handover processes present in Project A. The hope is also, according to [AC] to increase the team productivity and ensure a continuous growth of the people inside the project as they are encouraged and demanded to constantly learn new things.

In the internal documentation [ID], the project emphasizes close interaction with a reference customer. However, the Agile Manager [AM-B] explains that the interaction with the reference customer is limited to a demo for the customer once a month. The reference customer takes part in neither system analysis nor requirement specification. As of December 2012 no external releases of Product B have been made. However, continual work on the reference customers site is planned to start in week seven or eight 2013, according to the Support function manager [SFM].

As of December 2012 Project B consists of 14 Scrum teams – approximately 120 people. There are nine external consultants on-site in the project, the rest are employed by Ericsson. Project B is, similarly to Project A, mainly located at the PDU X head quarter. Many of the people that are now working in Project B have previously been working with Project A, according to the Head Project Owner [HPO-B].

7.1.3 The software development process in Project B - an agile approach

The software development in Project B follows an agile feature driven development model (often abbreviated FDD) [ID]. The project model incorporates continuous analysis of high-level architecture and system design. This continuous analysis work is called “Identification”, and is done by a central group for the whole project, some time ahead of the implementation of a feature. According to the Head Product Owner [HPO-B], lower level design decisions are made by members of the cross functional teams in preparation for developing each feature. Features are developed in a two-phase manner, where the preparation phase is called “Envisage” and the implementation phase is called “Realise”. These two phases are done by the cross-functional teams. The teams work in iterations (sprints) according to the Scrum methodology. Each sprint is three weeks long and after each sprint an internal demo is held for the other teams.

In Project B the full functional-testing responsibility on unit, integration and acceptance level lies on the cross-functional teams, whereas in Project A testing is the responsibility of the Integration & Verification unit, which is separated from the development teams. An Integration & Verification function is still present in Project B, but it’s involved later in the process, with focus on non-functional testing. This unit is called Release Verification.
Continuous integration is a key part of the software development in Project B. In parallel with the new product, Project B designs and implements an extensive end-to-end Continuous Integration (CI) framework. The CI framework is developed inside Project B and is still under development. By October 2012 the support for design, build and basic unit test was in place and automatic feature and system test was in the pipeline, explains the Head Product Owner in Project B (HPO-B). The framework has, according to internal documentation [ID], three main purposes: continuous testing, continuous integration, and continuous feedback. The idea is that the integration should be done automatically upon code commit, the tests should be automatically executed and feedback on build and test results should be given quickly to the developers. If the tests on all levels pass, the deployment should be made automatically.

The use of CI is, according to internal documentation [ID], expected to enable better and earlier testing, finding more faults earlier at a lower cost and providing feedback as soon as possible. The automated integration testing is expected to lead to faster integration since integration faults are found faster, and the higher level of automation is also expected to simplify the delivery
process. To enable continuous integration in an efficient manner, the Product B architecture has been designed with CI in mind and will thus hopefully overcome some of the integration problems present in Product A, explains the Agile Coach [AC]. The architecture of Product B is a service-oriented architecture developed by PDU X. Furthermore, the Product B architecture aims at being fully horizontally scalable, one of the challenges present in Product A according to the Agile Manager [AM-B].

7.1.5 Description of Survey B respondents

The survey data set included 48 responses complete enough to enable analysis (see chapter 5.2.4.1). 45 of these reported that they were Ericsson employees and 3 that they were working in Project B as external consultants. The respondents had worked for Ericsson between 0 and 20 years, with an average of 6.55 years. 34 of the respondents had been working in Project B for one year, seven for zero years and two for two years. Almost 90% of the respondents stated that they were part of a Scrum team, while 10% of them, five people, said that they were not.

The survey respondents [S-B] level of experience of agile and non-agile software development is illustrated in Figure 12.

![Survey B respondents experience of software development](image)

**Figure 12. The Survey B respondents experience of agile and non-agile software development.**

What role the respondents felt best corresponded with their primary work assignments were distributed among all the different roles. Two of the respondents didn’t respond to the question, while nine persons classified their primary work assignments with two different functional role descriptions and two persons with three different. The functional role descriptions of the respondents are illustrated in Figure 13.

---

5 A service-oriented architecture is “an application architecture within which all functions are defined as independent services with well-defined invokable interfaces which can be called in defined sequences to form business processes” [55]
Figure 13. The functional role descriptions of the Survey B respondents.

Approximately 60% (29 persons) of the respondents had previously worked with the development of Product A. In that project the largest number of persons had been working as a System Analyst/Architect (14 persons), followed by Designer (12 persons) and Tester/QA (11 persons). No one had been a Product Owner for Product A.

7.1.6 What agile practices are implemented in Project B?
In Project B the aim is to incorporate the principles of the Agile Manifesto into the organization, rather than just implementing a selection of practices, explains the Agile Coach [AC]. We have identified the implementation of several agile practices.

The earlier described continuous integration framework allows for end-to-end implementation of this practice in Project B. Project B has also fully implemented the practices of backlog, daily meetings, iteration planning, task board and sign up. These practices are all, except for sign up, key elements in the Scrum methodology. In addition to the daily meetings, Project B also has a “scrum of scrums” meeting three times per week, with representatives from all teams.

Teams in Project B are cross-functional and consist of a full scale of competence, from system analysis to testing. This is a full implementation of the whole team practice. Team members have been re-located, so that the whole team is working together in an open office space and can communicate face to face. Teams in Project B enjoy a high degree of autonomy and self-
organization, and can therefore be considered to have implemented the practice of empowerment. Collective ownership is used inside all teams in Project B, but not project-wide between the teams. For example, the developers are expected and encouraged to change code written by their team members, but members of one specific team aren’t expected to do changes to code primarily developed by another team.

Project B uses an iterative and incremental development method throughout the project. Releases could potentially be made at the end of every sprint, every three weeks. Internal demos and extensive retrospectives are also held at this time. An external demo is held every four weeks to showcase the latest development to the reference customer.

The use of information radiators has been formalized in Project B, and different types of information radiators are used. Refactoring is also used frequently; there are routines for when and how it should be done.

Pair programming is currently tried out in pilot teams. However, some resistance to this practice is found in the organization [AM-B] [AC], and we don’t consider it to be implemented.

Product B is based around a pre-defined, service-oriented architecture and work is done to support future functionality. Therefore, simple design could not be considered implemented. Evidence on the use of sustainable pace and on-site customer couldn’t either be found. Sometimes it’s considered that a Product Owner could be seen as an on-site customer, but this is not considered true in this thesis. The purpose of having an on-site customer is to get closer collaboration with the customer and better understand the customer’s real needs, and this cannot be fully achieved in this way if the “customer” on-site is not an actual customer. Project B has a close collaboration with one pilot customer, but that customer is not on-site.
Table 10. Agile practices implemented in Project B.

<table>
<thead>
<tr>
<th>Agile practice</th>
<th>Implementation status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backlog</td>
<td>Full</td>
</tr>
<tr>
<td>Collective ownership</td>
<td>Partial</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>Full</td>
</tr>
<tr>
<td>Daily meetings</td>
<td>Full</td>
</tr>
<tr>
<td>Demos</td>
<td>Full</td>
</tr>
<tr>
<td>Empowerment</td>
<td>Full</td>
</tr>
<tr>
<td>Frequent releases</td>
<td>Full</td>
</tr>
<tr>
<td>Information radiators</td>
<td>Full</td>
</tr>
<tr>
<td>Iteration planning</td>
<td>Full</td>
</tr>
<tr>
<td>Iterative and incremental development</td>
<td>Full</td>
</tr>
<tr>
<td>On-site customer</td>
<td>Not in use</td>
</tr>
<tr>
<td>Open office space</td>
<td>Full</td>
</tr>
<tr>
<td>Pair programming</td>
<td>Not in use</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Full</td>
</tr>
<tr>
<td>Retrospectives</td>
<td>Full</td>
</tr>
<tr>
<td>Sign up</td>
<td>Full</td>
</tr>
<tr>
<td>Simple design</td>
<td>Not in use</td>
</tr>
<tr>
<td>Sustainable pace</td>
<td>Not in use</td>
</tr>
<tr>
<td>Task board</td>
<td>Full</td>
</tr>
<tr>
<td>Whole team</td>
<td>Full</td>
</tr>
</tbody>
</table>

7.1.6.1 The project members’ perception of the agile practices implemented in Project B

The survey showed that the members of Project B are satisfied with the agile practices implemented in their project. All practices in the survey received an average score above five, with a median value of five or six, on a scale from 1='Not at all' to 7='Completely'. However, all practices received at least one individual score of three or less. Working in sprints – the agile practice of iterative development – received both the highest average score (5.74) and the lowest standard deviation (1.22), and can thus be considered to be the most popular. The product backlog received the lowest average score (5.00), with a standard deviation of 1.65. All questions had a standard deviation between 1.04 and 1.77. The question on iteration planning meetings had the highest standard deviation (1.77). The survey respondents’ satisfaction with the agile practices in Project B is illustrated in Figure 14.

The members of Project B were also satisfied with the overall ways of working in their project. 85.2% of the 47 project members that responded to this survey question gave the overall ways of working in Project B a score of 6 or more (on a scale from 1='Not at all' to 10='Completely'). On average the overall ways of working in Project B received a score of 7.51 with a standard deviation of 1.79. No respondent gave the overall ways of working a score of one or two.
The reactions were also positive in the open-ended last survey question. One Designer and Developer wrote:

“Overall it is a much improved way of working than traditional [Project A]. Sprint planning for small chunks of work i.e. 3 week sprints, is an excellent process that allows us to manage the software development process more easily. It allows changes to be incorporated more quickly and allows us to react to changes more quickly by the fact we only commit to 3 weeks of work.”

[Designer and Developer, Project B]

Another one agreed:

“Moving to [Project B] and an agile way or working has been very refreshing & a positive work experience thus far.”

[Tester/QA, Project B]

Yet another one was positive, but felt that Project B faces some challenges regarding project management:

“It’s a very collaborative [SIC], open create environment. This needs to be balanced with strong leadership team making clear, though, timely decisions and be more directive and dictating [SIC] to teams as and when is nesarcy [SIC]. The project is too loosely managed and I feel it’s going off track for a long time now. A clear correction path should be commnicated by the leadership/management team.”

[Tester/QA, Project B]

Figure 14. The survey respondents’ satisfaction with the agile practices in Project B.
On average the survey respondents had performed 2.44 of the six stated activities ‘Requirement specification’, ‘System design’, ‘Implementation’, ‘Writing tests’, ‘Running tests’ and ‘Refactoring’ during the last 7 days. The most frequently performed activities were Implementation (26 persons) and System design (25 persons). The least frequently performed were Requirement specification (13 persons) and Refactoring (15 persons). The activities Writing tests and Running tests had been performed by 19 persons each. Six respondents hadn’t performed any of the specified activities or chose to not respond to the questions (there was no ‘none’ option).

7.1.7 Internal software documentation in Project B
The teams in Project B use a set of supporting tools to handle their documentation needs. Each team has its own online wiki page that is used for documenting tools and how things are done. There is also a more static Project Site where steering and control documents, which are few, are posted by the project management. In addition, there is a document repository used for hosting the actual document files. Documentation of requirements and features is handled in the Product Backlog.

According to the survey [S-B], the project members perceive that the overall amount of internal software documentation in Project B is slightly too little. The average score of the respondents was 3.78 with a standard deviation of 1.10 on a scale from 1='Way too little' to 7='Way too much'. No respondent thought that the overall amount was way too much, while two persons thought it was way too little. The median score was 4.00. The project members also considered internal software documentation as important (mean 5.6; standard deviation 1.18) for Project B. They only perceived that internal software documentation can be replaced with face-to-face communication to a moderate extent (mean 3.55; standard deviation 1.49).

Regarding the different software documentation items the survey respondents perceived the amount of four of them as slightly too little: Requirement specifications, Process documentation, Work packages and Change requests. One of the respondents wrote in the open-ended survey question:

“More focus needs to go into documenting Non-Functional requirements and also an enormous amount of work needs to go into creating appropriately detailed User Stories with acceptance criteria attached.”

[System analyst/architect and Tester/QA, Project B]
The survey respondents further perceived the amount of Code comments and Trouble reports as just right and the Technical documentation of code as slightly too much. The median answer for all of the items was four. None of the respondents perceived the amount of any of the internal software documentation items as ‘Way too much’, while some respondents perceived each one of them as ‘Way too little’.

In the survey 43.5% of the respondents spent 15 minutes or less daily on producing internal software documentation and 37% spent 5 minutes or less. However, 19.6% spent an hour or more, every day. The time spent daily on producing internal software documentation by the survey respondents is shown in Figure 16.
The survey included no questions on the availability of internal software documentation in Project B, but one survey respondent wrote in the open-ended question that it is: “[..] hard to find documentation [..] documentation not updated / not maintained” [System Analyst/Arquitect, Project B]

7.1.8 Knowledge sharing in Project B

As described earlier, the teams in Project B are cross-functional, i.e. system analysts, designers and testers are all part of the same team. The idea is that all team members should be able to perform all tasks. To reach the full potential of the cross-functional teams a competence shift is necessary [ID] and the team members need to share their knowledge with each other. Team members are also undergoing training to broaden their competence, e.g. testers learn to do tasks that are traditionally done by developers, and developers are trained to do testing if needed.

Having experts within the teams is still a possible and natural occurrence, but the possibility for team members to help and support each other is a necessity to avoid the creation of bottlenecks, according to the Agile Coach [AC].

We investigated to what extent knowledge is shared in Project B through the survey and found that the project members get help from both members of their own team and other project teams to solve problems related to their project work. The survey respondents [S-B] reported that they, compared to finding the solution themselves get help from team members with an average frequency of 4.98 (standard deviation 1.27) and from members of other project teams with an average frequency of 4.09 (standard deviation 1.49) on a scale from 1='Never' to 7='Always'. They also stated that they had gained insights in other functional areas than their own from their team members (mean 5.46; standard deviation 1.68).

The survey respondents further reported that they feel that their team members have the necessary capabilities to help them with their project work (mean 5.53; standard deviation 1.47) and that they definitely feel aware of what capabilities their team members have (mean 5.54; standard deviation 1.31). This could be considered a precondition for knowledge to be shared.
According to the survey [S-B], the members of Project B felt that several agile practices help the knowledge sharing in their project. The respondents stated that they feel that the daily meetings do help their team to solve problems together (mean 5.15; standard deviation 1.73), as does definitely sitting together in an open office space (mean: 5.78; standard deviation 1.70). The survey respondents further felt that they get insights about the project they wouldn’t have gained otherwise from the iteration planning (mean 5.40; standard deviation 1.4), the retrospectives (mean 5.32; standard deviation 1.67) and the demos (mean: 5.22; standard deviation 1.64).

7.1.9 Project visibility in Project B

In general, the members of Project B feel aware of the project’s status, according to the survey results [S-B]. One Scrum Master wrote in the open-ended survey question:

“The team have [SIC] much more visibility on the product. We have clearer markers of progress that help people feel like they are achieving things”

[Scrum Master, Project B]

The survey respondents were most aware of their own team’s status (mean 6.04; standard deviation 1.21), while they were aware of the other team’s statuses to a lesser extent (mean 4.27; standard deviation 1.94). Quite a few of the survey respondents, 35.6%, didn’t feel aware at all of the other teams’ statuses (response score 1-3). The survey respondents further considered themselves aware of the entire Project B’s status relative its goals and plans (mean 4.98; standard deviation 1.64).

Out of the agile practices that the literature review suggested could have an impact on project visibility, continuous integration (mean 6.13; standard deviation 1.31) showed to be the most important in Project B [S-B]. Also the iteration planning meetings showed to be very important (mean 5.96; standard deviation 1.31). The survey respondents felt in fact that all the suggested practices were important for their awareness of the project’s status. In order of ascending importance for project visibility the other investigated agile practices were: open office space (mean 5.64; standard deviation 1.76), task boards (mean 5.51; standard deviation 1.75) and demos (mean 5.48; standard deviation 1.63).

7.1.10 Pressure and stress in Project B

Project B is an important, flagship project for Ericsson. According to one manager in PDU X the project’s time schedule is tight and challenging, regardless of the ways of working:

“It is a huge challenge whatever way we do it”

[Support Function Manager, PDU X]

The project members who answered the survey stated that they were moderately stressed by their work in Project B [S-B]. All questions on this theme gained a mean between three and four on a scale from 1 = ‘Not at all’ and 7 = ‘Extremely’. Worth to mention is also, that all questions were scored 7 – ‘Extremely stressed’ - by a couple of respondents.

One survey respondent commented on the stress of agile methods in the open ended question:

“Over all agile approach makes it interesting to work but has inherent stress factors”

[Designer and Developer, Project B]
Especially the *demos* are causing negative stress for the members of Project B. The level of negative stress caused by *demos* was on average perceived as 3.93 (standard deviation 1.84) and in total 10 persons - 22.2% - stated that they are very or extremely stressed by the *demos* in Project B. Also the expectation to deliver results at the end of each sprint (mean 3.72; standard deviation 1.81) and finish what one have *signed up for* (mean 3.89; standard deviation 1.79) were perceived as moderately stressful by the members of Project B. The respondents didn’t however feel stressed and pressured in a negative way to report progress on every *daily meeting* (mean 3.32; standard deviation 1.99).

7.1.10.1 Overtime density and concentration

In 2012 the overtime density in Project B was **2.37%**, i.e. for every 100 project hours 2.37 overtime hours were reported. The overtime hours were reported by **19%** of the project members.

7.1.11 The level of coordination effectiveness in Project B

Project B has approximately 120 project members and is expected to grow to a full staff of 300 within the next couple of years [*HPO-B*]. Although handovers are avoided in Project B since the feature teams include all functional roles and vertical competence across all architecture layers, the high-level system analysis is still done in a separate team and a separate release verification team handles the final testing. All teams are also to deliver to the same software system. There is consequently a need for the project members and teams in Project B to be able to coordinate effectively with each other.

On the direct question whether the project members feel that they can coordinate effectively within Project B the survey respondents [*S-B*] stated that they feel that they can (mean 5.14; standard deviation 1.71).

The level of coordination effectiveness in Project B was also calculated with help of the model proposed by Strode et al. [39] (see section 3.2.1.5 on page 25). In the survey, all the components in the model received a mean of four or higher (standard deviation 0.95-1.64) on a scale from one to seven. Lowest scores were received by the questions on whether the project members feel that they know what the other teams are doing (mean 4.24; standard deviation 1.64) and what capabilities they have (mean 4.86; standard deviation 1.61). Those two questions were the only to receive an average score below five.

The calculated the coordination effectiveness, made by calculating the average of the individual components, got a mean of 5.25 (standard deviation 1.03). The level of implicit coordination effectiveness, derived from the corresponding components, was 5.53 (standard deviation 0.92) and the level of explicit coordination effectiveness 4.72 (standard deviation 1.49).

The survey respondents [*S-B*] stated that they get help to know what their team members are doing by both *daily meetings* (mean 5.64; standard deviation 1.42) and *task boards* (mean 5.17; standard deviation 1.76). Other teams’ *task boards* also make project members more aware of what the other project teams are doing (mean 4.18; standard deviation 2.09).
7.1.12 Productivity in Project B

The survey respondents [S-B] rated all three levels of productivity – individual, team and project - as high; all questions got a mean value above five. On average the survey respondents rated the team’s productivity level as 5.75 (standard deviation 1.30), which interestingly enough is a higher average rate than what they gave themselves (mean 5.36; standard deviation 1.26). The overall productivity in Project B received an average score of 5.26 (standard deviation 1.16).

In the open-ended survey question several respondents commented on productive time lost on meetings. One Designer and Developer wrote:

“Sometimes I feel, that time is waisted [SIC] on some planning meeting. [...] Lots of time spent on meetings…”
[Designer and Developer, Project B]

A second respondent agreed:

“A 3 week sprint is short. Time is lost for demos, demo prep, meetings daily and otherwise in the end we are left with only 2 weeks and a bit to get our tasks to done. If we had four weeks we would spend a higher percentage of our time coding than in meetings etc. The team can take more stories on, its [SIC] not about the commitment or workload.”
[Tester/QA and Developer, Project B]

But a third survey respondent didn’t agree that the sprints are too short:

“[…] three weeks gives the coders the rigidity they need to get a decent bit of work done.”
[Scrum Master, Project B]

Regarding the open office space the respondents stated that they are slightly disturbed by sitting in an open office space (mean 3.04), but the disagreement was large (standard deviation 2.12) and some stated it to be very stressful and some not stressful at all. The size of the open office space was stated to be too big by two respondents in the open ended survey question. According to these respondents it would be better with separate areas for each team:

“Open plan sucks and is too disruptive. There should be sections for 8-10 people and those sections somewhat semi closed-off.”
[Tester, Project B]

“Open office is good but it only needs to be as large as the team.”
[Tester and Developer, Project B]

7.1.12.1 Functional productivity

Project B uses an IT system for tracking work items, defects and user stories. Every team has its own space in the system and manages its own work items. The system supports the work breakdown process encouraged by Scrum and several types of work items are present. The four most common work item types are epics, stories, tasks and sub-tasks. Epic is the most high-level one followed by story, task and sub-task. An epic corresponds to a high-level product requirement and a sub-task corresponds to what the teams commit to do during a sprint.

In Project B all teams – the cross-functional implementation teams as well as the supporting teams – follow the Scrum process and work in sprints. All teams also use the IT system for tracking work items. Table 11 presents the cycle time for each of the work items for the entire
Project B. Table 12 presents the cycle time for only the work items associated with the cross-functional, feature teams. The cycle times are based on data from February to December 2012 extracted from the IT system.

In total 5236 work items were present in the system, 3936 of them were closed or resolved and as such included in the calculation. 4398 work items were associated to the feature teams, 3295 of them were closed.

In Table 11 and Table 12 we see that for the entire Project B the high-level requirement (epic) cycle time is 87 days and for only the feature teams it is 89 days. The average project cycle time for the sub-tasks is just slightly shorter than one sprint (17 days). In the feature teams the sub-task cycle time is 18 days.

<table>
<thead>
<tr>
<th>Work item cycle time - Project summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of items</strong></td>
</tr>
<tr>
<td>Epics</td>
</tr>
<tr>
<td>Stories</td>
</tr>
<tr>
<td>Tasks</td>
</tr>
<tr>
<td>Sub-tasks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Work item cycle time - Feature teams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of items</strong></td>
</tr>
<tr>
<td>Epics</td>
</tr>
<tr>
<td>Stories</td>
</tr>
<tr>
<td>Tasks</td>
</tr>
<tr>
<td>Sub-tasks</td>
</tr>
</tbody>
</table>

7.1.12.2 Defect fixing efficiency

A “bug” in Project B is in many aspects the same thing as a trouble report in Project A – a report concerning a defect in the software or software documentation. Although used in the project, only two trouble reports (TR’s) were present. The bugs and trouble reports are tracked with the same IT-system as the other work items (see section 7.1.12.1).

Bug cycle time was calculated as a measure of defect fixing efficiency. Between February and December 2012, 203 bug reports were closed. The average cycle time was 10 calendar days and 13 hours from the original submission to the finish date. The longest cycle time was 130 days, and the shortest was not even a minute. This bug was most likely already resolved when inserted in the system and only inserted for documenting purposes. There were no rejected or cancelled bug reports in the data set. The average bug cycle time for the feature teams was 17 days and 4 hours.
### Table 13. Bug cycle time in Project B.

<table>
<thead>
<tr>
<th></th>
<th>Number of items</th>
<th>Average cycle time</th>
<th>Maximum cycle time</th>
<th>Minimum cycle time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project bugs</strong></td>
<td>203</td>
<td>10 days 13 hours</td>
<td>130 days 6 hours</td>
<td>0 days 0 hours</td>
</tr>
<tr>
<td><strong>Project TR’s</strong></td>
<td>2</td>
<td>2 days 1 hours</td>
<td>4 days 0 hours</td>
<td>0 days 3 hours</td>
</tr>
<tr>
<td><strong>Feature team bugs</strong></td>
<td>59</td>
<td>17 days 4 hours</td>
<td>130 days 6 hours</td>
<td>0 days 0 hours</td>
</tr>
<tr>
<td><strong>Feature team TR’s</strong></td>
<td>2</td>
<td>2 days 1 hours</td>
<td>4 days 0 hours</td>
<td>0 days 3 hours</td>
</tr>
</tbody>
</table>

#### 7.1.13 Software quality in Project B

In the survey the respondents rated the quality produced in Project B as good (mean 5.32; standard deviation 1.35 on a response scale from 1=’Extremely poor’ to 7=’Extremely good’). However, one survey respondent also commented on the fact that it’s hard to judge the “real” quality of Product B since the product hasn’t been released to and tested by external customers yet:

> “However it has to be said we haven’t delivered to the customer yet and that will be the real measure of our success. […] Indications are strong… but we have to realize the product to be 100% sure.”

[Scrum Master, Project B]

Another survey respondent points out that not all aspects of the quality of the software produced is under the control of Project B:

> “Certain support deliverables [Names of deliverables] are not 100% ready, this causes technical debt on other teams and puts support teams under pressure to finish, causing poor quality software”

[Designer and Developer, Project B]

#### 7.1.13.1 Test case pass rate

The test case pass rate of Project B was analyzed using data from the Continuous Integration framework. Unfortunately, the only available data was the results of the last 30 instances for every test set. The test sets are run with a frequency varying from a few per week to several times per hour, which resulted in a data set spanning 24 days with the majority of the test case data stemming from the end of the period. The number of test cases run on a single day varied from a handful to over a thousand. For test sets that ran more than once at a specific day, only the last run was counted. The gathered data was separated into non-acceptance tests and acceptance tests, and is presented in Figure 17 and Figure 18. The red dotted line marks the day for the demo at the end of a sprint. Some of the acceptance tests are using the same test sets as the non-acceptance tests. Test coverage for the acceptance tests was not available.
Figure 17. Non-acceptance test pass rate and line coverage

Figure 18. Acceptance test pass rate

7.1.13.2 Defect density
From the start of the project in February 2012 until the end of the year, 1.54 defects were reported for every 1000 project hours. However, a quality engineer at PDU X pointed out that not all defects found in Project B are present in the data, as very simple bugs might be fixed right away by the developers without being reported [QE].
7.1.14 Project B success rate
Between April and October 2012, Project B overshot its budget by 2.4%. However, the outcome numbers for April-August matched the budget exactly, which prompted concern that the provided numbers weren’t accurate. If only the numbers for September and October are considered, Project B overshot its budget by 7.0%. The September budget was overshot by 18%, while October was actually under budget by 3.5%.

When it comes to on-time release of features, Project B has unfortunately not yet reached the point where features are released to an external customer. Therefore, this metric cannot be used for Project B. The explicit hope in the project is that customer interaction will attain an on-time release of features of 100%.

“The hope is that in an agile world we wouldn’t have restrictions. Instead we would communicate with the customer”
[Support Function Manager, PDU X]

7.2 Data analysis – what impact does the use of agile principles and practices have in Project B?

7.2.1 Representativeness of the survey respondents
The survey respondents from Project B had on average worked for Ericsson a little longer than six and a half years. Many of the respondents had also worked in Project A before. Among the survey respondents there was a mix of people that have been working for Ericsson between 0 and 20 years and they had widely varying amount of experience in both agile and non-agile software development. This means that the people working in Project B have quite some experience from working for Ericsson in general, and in Projects A and B in particular.

In all of Project B, 9 of 121 people, 7.44%, are not employed by Ericsson but are external consultants. Of the survey respondents, 3 of 48 people, 6.25%, responded that they were external consultants. Those ratios are pretty close, and therefore we consider, in the matter of ratio of external consultants, that the survey respondents are representative of the whole project.

7.2.2 Experiences and perception of agile principles and practices
Most of the respondents stated that they were part of a Scrum team, which confirms the information presented in previous chapters – Project B has adopted Scrum and other agile principles and practices to a large extent.

Most of the studied agile practices have been fully implemented in Project B. In general, the practices are well thought of by the members of Project B. All practices received a score well towards “completely satisfied” in the survey. Project B members are also satisfied with the general ways of working in the project. Nevertheless, all practices also received bad scores by individual respondents, a sign that all project members aren’t satisfied with all practices. It wasn’t statistically possible for us to investigate if the functional roles differed in their perception of agile practices, but it could be interesting to investigate if some agile practices work better for some functional roles. The overall impression is that the people in the project are satisfied with the ways of working and the implemented agile practices. This is a sign that the practices are working as they should.
The open-ended survey response on lack of a dictating management (page 72) is interesting as empowerment of the teams is one of the key ideas in agile software development: the management should let the teams decide for themselves and not mandate and dictate. However, as earlier discussed, agile software development is more frequently implemented in small organizations, this response can be a sign that the empowerment isn’t working as good in larger organizations where hierarchical structures typically are present. But it’s also possible that the survey respondent simple isn’t accustomed to this way of working and will, once he or she is used to it, consider it beneficial.

Iterative development and daily meetings were the practices that received the highest average scores by the survey respondents, why it can be advisable to implement these practices first to receive maximum effect. They are also both central parts of the Scrum methodology.

When asked about what activities they perform on a weekly basis, the survey respondents in Project B stated that they, on average, performed 2.44 of the six suggested activities every week. This is a clear indication of the cross-functionality of the team members in Project B, and in such an effect of the practice whole team.

7.2.3 Impact on internal software documentation

The answers to the survey in Project B [S-B] indicate that the project members perceive the overall amount of internal software documentation in the project as slightly too small. When looking into different kinds of documentation the average answer for “Requirement specifications” stands out as “slightly too little”. Overall, the average answers tend to be on the “slightly too small” side of the scale, but not by much. We see this as a weak indication that some kinds of documentation aren’t produced in a sufficient amount in Project B, as proposed by Stettina et al [10]. We however don’t see that the effect found by Pikkarainen et al [7], that too little technical documentation is produced, is happening in Project B. Oppositely, we find that process documentation is somewhat lacking.

In Project B, it seems like a large portion of the project members spends little or no time on producing internal software documentation, while another portion of the project members spends hours every day. There is also a “middle section” of project members that spends 6-30 minutes every day on producing documentation. The majority of the survey respondents in Project B spend more than 15 minutes every day on producing internal documentation, which is more than what was found by Stettina et al [10]. This is an indication that the members of Project B produce more internal documentation than what is common in other agile software development projects, and may very well also explain why we didn’t find that too little documentation was available, as the same authors suggested. Project B is a large-scale project with many dependencies and external stakeholders, which increases the need for documentation. This might be one explanation as to why Project B is producing more documentation than other agile projects. Company policies regarding documentation might be another explanation.

Members of Project B think on average that internal software documentation can be replaced with face-to-face communication only to a moderate extent. In this way, they agree with what
was found by Stettina et al [10], that face-to-face communication is not enough to capture and sustain knowledge. This further shows that the project members put a high value in internal software documentation, and that documentation is a vital part of the communication, even in projects using agile principles and practices.

7.2.4 Impact on knowledge sharing

We expected members of Project B to perceive that knowledge is shared to a high extent within the project, since the project has implemented the agile practices with suggested effects on knowledge sharing. This is also true. From the survey data we can see that knowledge is shared in Project B, both inside teams and between teams.

According to the survey results, members of Project B get help from their team members more often than they find the solution themselves, and also get help from members of other teams. Another indication of this is the answer that the project members on average agreed that they had gained insights from other functional areas than their own from other team members. They also believe that their team members have the necessary capabilities to help them with their work. This indicates that the cross-functional teams used in Project B, which is an implementation of the practice whole team, has had a positive effect of the knowledge sharing in Project B, as proposed by Strode et al [8].

Survey respondents in Project B further agree that retrospectives, daily meetings and iteration planning has helped them in some way to solve problems together or to gain new insights. This verifies what was proposed by McHugh et al [23], that these practices can create forums for creating and sharing knowledge. The survey respondents also agree that they have gained insights from the demos, as suggested by Strode et al [8].

The impact of open office space on knowledge sharing gained the highest both mean and median values of all the investigated agile practices. This indicates that the use of an open office space is the most important agile practice for knowledge sharing in Project B. In this way, the theory put forward by Mishra and Mishra [24], that open office space improves developer learning by improving communication, is true for Project B.

The survey respondents further definitely felt aware of what capabilities their team members have. This supports the findings of Strode et al [8], who stated that open office space and whole team should increase this awareness. The effect could also have been helped by iterative and incremental development, as proposed by McHugh et al [23]. Our belief is that all the studied practices took part in creating the capability awareness.

However, while knowledge is definitely shared with ease in Project B, the knowledge sharing maybe hasn’t reached its full potential. The survey respondents report that they definitely feel that the team members do have the capabilities to help them, but at the same time that they don’t always ask for help. The question is then, why they don’t always ask for help? It is possible that it is more convenient and easier to find the solution oneself. But this mindset is also related to a traditional way of working - agile working methods encourage interactions [13] and one of the purposes of the open office space is that the teams should solve problems together. It could be
worthwhile to further investigate how knowledge sharing can be improved even more in Project B.

7.2.5 Impact on project visibility
Survey respondents are close to completely aware of their own team’s status, but less aware of the status of other teams. This was to be expected, as several agile practices that have an effect on project visibility are mainly aimed at the team level.

When the survey respondents were asked outright what practices they considered important for their awareness of the project’s status, continuous integration and iteration planning received the highest scores, while open office space, demos and task boards received lower scores, although still considered important. This confirms the findings of Pikkarainen et al [7], that project visibility is increased by these practices. It also confirms the findings of Laanti et al [34], that it is possible to achieve increased project visibility also in large-scale projects.

That continuous integration showed to be the most important contributor to project visibility is positive news, since faster and more feedback was one of the goals with the extensive continuous integration framework implemented in Project B.

7.2.6 Impact on pressure and stress
On average, the survey respondents in Project B were moderately stressed and pressured in a negative way by their work in the project. All specified agile practices were rated as extremely stress-inducing by a couple of respondents. Demos received the highest average rating, and 10 people responded that they are very or extremely stressed by them. One occurrence of a stressful period of overtime just before a demo for the reference customer was also found in the internal documentation [ID]. This is a clear indication that we see the same effect as Strode et all [8]; the demos are increasing stress. It could therefore be advised to review the routines concerning demos. But iterative development and sign-up received almost as high an average rating. This suggests that sign-up is increasing the pressure, as suggested by McHugh et al [23].

Interestingly, no single practice was perceived, on average, as more negatively stressing than the general work in Project B. This presumably means that there are other factors in Project B that cause negative stress and pressure, more so than the specified practices. One possible explanation is that the importance of the project, from an Ericsson perspective, is causing stress. Another explanation is an agile cocktail-effect, where the mix of different agile practices blended together cause negative pressure and stress, without it being possible to pinpoint the cause to a single practice. However we don’t consider the overtime reported in the project as very much and do therefore not believe the average level of pressure and stress is very high in Project B.

If the overtime had been evenly distributed among all project members it would have meant that every project member needed to work approximately one hour (0.948 hours) overtime per 40-hour work week. If we take into account that only 19% of the project members work overtime, these project members needed to work approximately five hours overtime each 40-hour work week. Neither of these numbers is considered very high. However, that only a fifth of the project members take on the overtime hours can’t be considered as sharing the responsibility, as advocated by the agile practice collective ownership. Unfortunately we had no further data on the
distribution of the overtime. It is possible that the majority of the overtime was reported by just a few of the 19% during just a few days or weeks, which would then have signified a very stressful period. But we have found no indications that this would be the case.

All in all, based on the numbers and the survey, the agile practices in Project B aren’t causing pressure and stress to a large extent. The agile practice sustainable pace isn’t explicitly stated to be implemented in Project B, but we think that it well could be.

### 7.2.7 Impact on coordination effectiveness

The questions about coordination effectiveness in Project B received high scores on average, and the standard deviation was quite low.

Although both the “other team” components still received a mean above four, the values are considerably lower than the others and it can therefore be seen as an indication that the coordination effectiveness is lower between teams than inside teams in Project B. This pattern matches the one regarding project visibility. Looking at actual agile practices, it seems like daily meetings help the team members to coordinate more than the task boards do, although both practices appear to make an important contribution. Coordination with other teams is also improved by their task boards, but the high standard deviation of these questions makes this observation a more careful one.

In summary, it is considered that Project B can coordinate effectively, and that agile practices contribute to this.

### 7.2.8 Impact on productivity

Productivity in Project B is on average rated as high by the project members. The project members gave their own productivity a lower score than the team productivity. This can be seen as an expression of a belief in Project B that the team is the most productive unit; although the difference in average value is not very ample.

Many survey respondents brought up issues related to productivity in their answers to the open question. Many of these answers speak of things that should lower the productivity, which is interesting since productivity is still rated as high.

Regarding work item cycle times we see that, as expected, the cycle time gets shorter with each item breakdown level. This is expected since the complexity and extent should decrease with each breakdown. There are also work items and bugs with a cycle time of 0 days and 0 hours. This was most likely caused by the work item or bug being resolved before it was put into the system.

The work item cycle time in Project B is longer in the feature teams compared to the project average, especially when looking at tasks. This could be an indication that the feature teams in Project B are less efficient than the supporting teams. Another reason could be that the tasks done by the feature teams are larger or more complex than those of the supporting teams. The breaking down of stories into tasks and subtasks is predominantly done on a team level, and what is deemed an appropriate task size may vary greatly between teams.
When studying only at the feature teams, it is surprising to see that the average cycle time for a user story is 45 days, definitely longer than a three week sprint. This would mean that the feature teams aren’t on average able to deliver business value at the end of each sprint, and as such not reaching the project goal. This is however not a definitive indication that the average user story, once committed to a sprint, takes longer than a sprint to finish. A story can exist in the product backlog for a while before being accepted into a sprint.

When considering bug cycle time, again the cycle times for the feature teams are longer than the project average. Bugs can of course also differ in complexity, but the subjective element of breaking down into tasks and subtasks is gone. The implication that the feature teams are slower than the supporting teams is therefore stronger in this case. However, the effect could still be entirely explained by more complex bugs being handled by the feature teams.

The average feature bug cycle time in Project B is a little bit shorter than the average feature sub-task, but the difference is only a day and a half. This means that it takes just a little longer to finish a sub-task in Project B, than it takes to fix the average bug. However, the cycle times doesn’t show anything about how much effort is put into resolving an item during its cycle, only how long time it takes.

It is possible that the high coordination in Project B, thanks to agile practices, has led to a high level of productivity. However, some of the open answers in the survey indicate that too much time is spent on coordination. A designer and developer says that time is wasted on planning meetings, and a system analyst/architect, designer and tester wishes that more time should be spent on work, and less on meetings and documentation. Therefore we cannot, at this stage, say that the agile practices have improved productivity by improving coordination.

It is also possible that a lower amount of documentation in Project B has led to more time being available for other activities, and therefore an increased productivity. However, we have seen that survey respondents from Project B spend more time on producing internal software documentation than members of other agile projects. Again, we can’t say at this stage if the agile practices in Project B have improved productivity by lowering the need for documentation.

The members of Project B didn’t seem to be more than slightly disturbed by the open office space, as suggested by [7] and [21]. We therefore don’t find any support for our initial assumption that the open office space could affect productivity negatively. The large dispersion in the results on this practice isn’t that surprising considering we’re all different and more or less sensitive to stress and pressure. However, since two respondents commented on the size of the open office space in the open-ended question it could be worth to investigate making separate open office-cubicles for each team, especially since this is also suggested by [24].

7.2.9 Impact on software quality
The project members rated the quality of the software produced in Project B as good. This is a good sign, but it is hard to truly evaluate the “real” quality of Product B, since it still hasn’t been released to a customer and tested in a live application. As one of the survey respondents pointed
out, the product has to be realized before any ultimate conclusions can be drawn. We can however look at indications of software quality.

It is evident in the charts in 7.1.13.1 on page 80 that the test case pass rate is consistently high during the entire duration of a sprint, but a bit lower a week after the start. The consistently high pass rate is an expected outcome of using continuous integration. We conclude that the faster feedback achieved by using continuous integration and an incremental and iterative development process has an important part in keeping the pass rate high.

The defect density of Project B is low, 1.54 defects per 1000 project hours. It’s however possible that some simple bugs are not included in the data, as they might not be reported at all. Nevertheless, since those bugs are fixed right away, they don’t really affect the quality of the software and it doesn’t really matter that they are not in the data set. The defect density is still an indication of high software quality.

When looking at the test case pass rate and defect density metrics for Project B, it’s almost surprising that the developers didn’t rate the quality of the software higher than they did. All numbers indicate that the quality of the software produced in Project B is really good. Therefore, we agree that agile software development can increase the quality of the software, as suggested by Abbas et al [28]. However, we have not found any signs that that the practice of empowerment has lowered the software quality, as suggested by Lee and Xia [25].

### 7.2.10 Impact on success rate

The budget numbers show that Project B overshot its budget by 7.0% during September and October. This is a bit too much, especially considering that the budget for September was overshot by 18%. The number is far above the OK level of a few percent (see 6.1.13.2). It should however be taken into consideration that Project B has grown a lot since the start, and that it can be hard to properly budget a growing organization. A project overshooting its budget can also have many other reasons, but what is clear is that empowerment didn’t ultimately help Project B to stay within budget, as was proposed by Lee and Xia [25].

It’s possible that the use of daily meetings has contributed to an escalation of commitment, which in turn has led to the budget being overrun (as suggested by Stray et al [11]). A possible way this could have happened in Project B would be that some of the teams had made a bad design decision and stuck with it too long, causing lots of hours to be spent in vain. An interesting single data point that actually indicates an escalation of commitment is the tester/QA that said the project has been going off track for a long time (see 7.1.6.1, page 71). However, when looking at the overtime metrics we don’t see an excessive amount of overtime in Project B, and we have not seen much of other indications towards an escalation of commitment in our empirical findings.

As the project still has a long time to go, it’s possible that the missed budget might be cancelled out by later savings, and that the project as a whole stays closer within budget. Also, as the project grows, the relative significance of the September and October budgets will be lower. Nevertheless, the indication right now is that Project B is overshooting its budget.
As project B has yet to reach a stage where features are released to external customers, no analysis of on-time release of features is possible. Only looking at two months budget is off course giving us a very narrow perspective on the success rate of Project B. However, we must use the data we have available to make our analysis, and that analysis must be that the indications we see, although they are not many, is that Project B has not gained a high rate of success thanks to agile practices.

7.3 Case B conclusions

Internal software documentation was rated as very important by the members of Project B. We saw indications that the amount of documentation concerning the development process is slightly too little. The amount of technical documentation was perceived as just right. It can be assumed that the slight lack of documentation has been an impact of agile principles and practices. The members of Project B put more effort into producing documentation than other agile projects.

Knowledge is shared in Project B to a high extent and many agile practices contribute to this. Members of Project B are also very aware of the status of their own team, thanks to the use of agile principles and practices, especially continuous integration. The agile way of working has further led to an awareness of the status of other teams and the entire project. Project B can also coordinate effectively, and the agile principles and practices that are used in the project contribute to this.

Members of Project B are under negative pressure and stress to a moderate extent. Some agile practices have contributed to this, especially demos. However, we also believe that there are other factors that have impacted the stress level. Members of Project B don’t work a lot of overtime.

The team is perceived as the most productive unit in Project B. We have seen no signs that the use of open office space impacts the developers’ ability to focus and therefore don’t believe that it affects productivity negatively. Quality-wise, there are indications that the software quality in Project B is high, and that is has been helped by the use of agile principles and practices, especially continuous integration. These indications are matched by the project members’ perception of the quality. Regarding success rate, we have had access to a very limited amount of data. The indication is that the agile ways of working in Project B have not led to an increased success rate.
8 Cross-case analysis

This chapter analyzes each of the areas in the conceptual framework from a cross-case perspective to see whether the suggested effects of agile principles and practices are more present in Project B, thanks to its higher adoption of agile principles and practices.

8.1 Difference in implementation of agile practices

As described in section 5.2.5, the point of departure for the analysis is the differences in ways of working between Project A and Project B, more precisely to what extent the two projects have adopted agile principles and practices.

Project B on the one hand has an explicit goal to work in an agile way, and has adopted agile principles and practices uniformly across the whole project. Project A on the other hand is overall a plan-driven, traditional project, although some parts of the project has adopted agile principles and practices to a moderate extent. Some agile practices are in use in both Project A and Project B, while others are in use only in Project B. There are no practices that have been implemented in Project A but not in Project B. A couple of agile practices that appear in the theoretical background in chapter 2; on-site customer, pair programming, simple design and sustainable pace, are not in use in either project. The use of agile practices in the two projects is compared in Table 14. A definition of the agile practices can be found in chapter 2.1.

Table 14. Agile practices in projects A and B.

<table>
<thead>
<tr>
<th>Agile practice</th>
<th>Implementation status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Project A</td>
</tr>
<tr>
<td>Backlog</td>
<td>Partial</td>
</tr>
<tr>
<td>Collective ownership</td>
<td>Partial</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>Partial</td>
</tr>
<tr>
<td>Daily meetings</td>
<td>Partial</td>
</tr>
<tr>
<td>Demos</td>
<td>Partial</td>
</tr>
<tr>
<td>Empowerment</td>
<td>Not in use</td>
</tr>
<tr>
<td>Frequent releases</td>
<td>Partial</td>
</tr>
<tr>
<td>Information radiators</td>
<td>Partial</td>
</tr>
<tr>
<td>Iteration planning</td>
<td>Partial</td>
</tr>
<tr>
<td>Iterative and incremental development</td>
<td>Partial</td>
</tr>
<tr>
<td>On-site customer</td>
<td>Not in use</td>
</tr>
<tr>
<td>Open office space</td>
<td>Full</td>
</tr>
<tr>
<td>Pair programming</td>
<td>Not in use</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Not in use</td>
</tr>
<tr>
<td>Retrospectives</td>
<td>Partial</td>
</tr>
<tr>
<td>Sign up</td>
<td>Partial</td>
</tr>
<tr>
<td>Simple design</td>
<td>Not in use</td>
</tr>
<tr>
<td>Sustainable pace</td>
<td>Not in use</td>
</tr>
<tr>
<td>Task board</td>
<td>Partial</td>
</tr>
<tr>
<td>Whole team</td>
<td>Not in use</td>
</tr>
</tbody>
</table>
8.1.1 Perception of agile practices

It is evident from the survey that the members of Project B are considerably more satisfied with the ways of working in their project than the members of Project A. The survey respondents from Project B rated their satisfaction of the overall ways of working in their project on average two points higher than the respondents from Project A. A generally higher level of satisfaction could also be seen in the open-ended survey answers: the open-ended survey question in Project B received more positive comments than the one in Project A.

The respondents from Project B also rated their experiences of several agile practices significantly higher than the respondents from Project A. These practices were: whole team, refactoring, continuous integration and retrospectives. The largest difference in mean was achieved for the practice of refactoring, which had a difference in mean value of 1.7. It is safe to assume that this is caused by the different product architectures; it is almost certainly harder to refactor an old product with complex internal dependencies than a new service-oriented one.

That the members of Project B are more satisfied with many agile practices and also with the overall ways of working is in line with what we expected since Project B has implemented these agile practices from the start. It is also in line with what was found by Laanti et al [34]. Project B has adapted the project organization and product architecture to agile software development, which is expected to facilitate the use of the agile practices. This result could therefore be seen as an indication that it actually is necessary to adapt the product and project design to agile software development to receive the full benefits of it. We think that the high level of satisfaction with agile ways of working should be a nice confirmation for the management of Project B and PDU X that agile ways of working is appreciated by the employees of Ericsson.

No significant difference could be supported in the respondents’ experiences of product backlog, daily meetings, task boards, iteration planning, demos, open office space and iterative development; respondents from both teams are equally satisfied with them. Worth to note is that these are all practices, except open office, key elements of the Scrum methodology, why this could be an indication that where implemented in Project A, Scrum works equally good as in Project B. For Ericsson this could be seen as an indication that it is feasible to introduce Scrum into legacy projects.

8.2 Has the use of agile principles and practices had an impact on the use of internal software documentation in Project A and Project B?

The approach to internal software documentation in agile projects is driven by the Agile Manifesto that values face-to-face conversation and “working software over extensive documentation” [13]. While this seems like a healthy approach, the theoretical background presented in section 2.10 also showed evidence that the internal software documentation produced in agile projects may not be sufficient for knowledge sharing between teams [7] and to preserve knowledge in the long term [10]. Since Project B has implemented agile principles and practices to a larger extent than Project A, we expected – based on our theoretical background – that the members of Project B would perceive the internal software documentation in their project as less sufficient than the members of Project A. This was not the case.
The survey respondents in Project B gave the overall amount of internal software documentation a slightly lower score (mean 3.78) than the survey respondents in Project A (mean 3.95). There was also a larger portion of the survey respondents in Project A that perceived the amount as too much, compared to Project B. However, the difference in mean values between the two projects was not statistically significant. Therefore we must say that, based on the survey data, the overall amount of documentation in both Project A and Project B is just about right – neither too much nor insufficient. This contradicts the findings of Stettina et al [10], where the members of agile software development projects considered the software documentation generated in their projects as insufficient. There could be several explanations why we didn’t find this pattern in Project A and Project B. A first one is that the findings of [10] simply aren’t valid for all agile projects, or for large-scale agile projects. A second reason could be that the number of survey respondents in Project A was too small to accurately reflect project attitudes at large. A third reason could be that also Project B isn’t fully following agile principles and practices in this regard. It is possible that general Ericsson policies and directives affect the software documentation produced in Project B more than the Agile Manifesto.

The survey respondents from Project A perceived that internal documentation is more important, nor believe that it can be replaced with face-to-face conversation to a larger extent than the survey respondents from Project A. We therefore conclude that internal software documentation is important, no matter the ways of working.

There was however a statistically significant difference between the respondents from Project A and Project B regarding several software documentation items. In Project A the survey respondents perceived the amount of Requirement specifications as slightly too much, while the respondents in Project B perceived the amount of them as slightly too little. The same was true for Process documentation; the respondents from Project A perceived the amount of Process documentation as slightly too much, while the respondents from Project B perceived it as just right or slightly too little. Also regarding Trouble Reports and Change Requests the survey respondents from Project A considered them slightly too many, while the survey respondents from Project B considered them just enough or slightly too few, with a statistically significant difference.

The average perceptions of the amount of internal software documentation items in Project B are all very close to four, which would be just right, while they in Project A are perceived as slightly too much. If we combine this with our finding that no significant difference could be found in the overall amount of internal software documentation between the two projects, it seems like the use of agile principles and practices in Project B has led to a better, more balanced use of internal software documentation. While the agile software development method appears to have reduced the amount of internal software documentation produced, it is not to a level of insufficiency, as [10] concluded. Using agile principles and practices has rather reduced the unnecessary internal software documentation.

The opposite relationship compared to the other documentation items was however true for Technical documentation of program code in the two projects. The survey respondents from Project A perceived the technical documentation of program code as slightly too little, while the
survey respondents from Project B perceived it as just right. This was expected as it is in line with the findings of Pikkarainen et al [7]. In the article, a separate team of testers felt that the documentation produced by the Scrum team, who did the implementation was insufficient. In our study it is Project A that has separate functional teams, while Project B has not. It can also be assumed that the need for documentation of program code overall is smaller in Project B, since handovers between departments are avoided thanks to having the teams vertically responsible for features. Another possible explanation is the wider use of modern software such as work item and issue tracking tools in Project B. The use of such systems is not a must in agile software development, but they support the agile process. A third possible explanation is that documentation of program code is automatically generated to a larger extent in Project B due to the higher degree of automation in the build and integration process in Project B.

Although happy with the overall amount, it seems from the open-ended survey answers that some members of both projects are not fully satisfied with the details in the internal software documentation produced in their projects. One survey respondent in Project A states that there is too much unnecessary documentation and still not enough details in it and one respondent from Project B writes that the non-functional requirements and user stories aren’t documented in enough detail. While it’s nearly impossible to satisfy everyone, we think that this is an indication that it would be worthwhile to review the use of some software documentation.

One survey respondent in Project A does also highlight the risk of undocumented knowledge getting lost when people leave the organization. Although Project A in this study is representing a less agile project, parts of Project A have implemented agile ways of working and this particular respondent has stated that he or she belongs to a Scrum team and solely has got experience from agile software development. We therefore consider that this supports the proposition by [10] of a higher risk of losing undocumented knowledge when working in an agile way. As Ericsson, like many other companies, undergoes re-organizations now and then, and the workforce will come and go over time, this issue should be worth further consideration.

Surprisingly enough the survey respondents from Project A in general spent less time on documentation than the respondents from Project B. In none of the projects did the majority of the respondents spend less than 15 minutes daily on producing documentation, as reported true for agile projects by [10]. Again, this may be explained by the large scale of both projects and general company documentation policies.

8.3 Is knowledge more easily shared in Project B, compared to Project A thanks to the use of agile principles and practices?

Knowledge sharing in agile software development projects is a two-sided coin. Agile software development requires a more extensive knowledge sharing. For example, an implementation of whole team requires the team members to acquire competence from all functional areas. At the same time are agile principles and practices shown to facilitate knowledge sharing [8] [21] [23] [24]. Project B has implemented agile principles and practices to a larger extent than Project A. Knowledge sharing is also emphasized by the project management in Project B, which believes that a competence shift is necessary for the new ways of working to reach its potential. We therefore expected to find a higher level of knowledge sharing in Project B than in Project A.
To be able to measure the abstract concept of knowledge sharing we investigated it from three aspects: to what extent the project members do help each other, to what extent they feel that their project members can help them and how much they have learned about other functional areas from team members. The latter would be an indication whether the competence shift necessary for *whole team* is taking place. We also investigated the level of ‘cross-functionality’ in the projects, i.e. to what extent the project members performed tasks from more than one functional area.

Between the two projects we could not find a statistically significant difference in how much the project members helped each other or to what extent they felt they could get help. The respondents from both projects also thought that their team members have the necessary capabilities to help them with their project work. We therefore conclude that the use of agile practices has brought a potential for more intra-team knowledge sharing in Project B, but that this potential is not fully taken advantage of by the project members.

We could however find a statistically significant difference in how much the project members believed they had learned about other functional areas from their team members. The members from Project B stated in the survey that they have gained more insights from other functional areas than their own, compared to the members from Project A. The members from Project B also stated that they had performed a greater number of activities from different functional areas during the last seven days. This is in line with the findings of [21] and [8], who reported that the practice of *whole team* widens the developer’s knowledge and make team members able to perform each other’s activities. We can therefore confirm that the practice of *whole team* is more widely spread in Project B, compared to Project A, and that this has had positive effects on the knowledge sharing.

There was also a statistically significant difference between the two projects in the extent agile practices contribute to knowledge sharing. The members of Project B felt that the *iteration planning* meetings give insights about the project to a higher extent than the members of Project A. The members of Project B also felt that they get insights from the *retrospective meetings* and the *demos* to a much higher extent. This difference could be an indication of a – from a knowledge sharing perspective – more suitable implementation of these agile practices in Project B, compared to Project A. Project B has adapted its project organization and the product architecture to better suit the agile ways of working. The difference could also be an indication that a project-wide implementation is necessary to gain full effect. Another possible explanation is communication, it may be that the purpose and structure of *iteration planning* meetings, *retrospectives* and *demos* are communicated differently in Project B, compared to Project A, or that the intention with them is different.

It seems then that the conditions for knowledge sharing and managing the necessary competence of the agile software development are present in Project B, but that the full potential of this isn’t taken advantage of. Knowledge is shared with the same ease in Project A and Project B. While this finding somewhat contradicts the findings in the theoretical background, it is positive that knowledge sharing does happen in both agile and non-agile projects in Ericsson. Nonetheless it’s
always worth aiming higher, and since improved knowledge sharing is a claimed benefit with agile software development Ericsson should try to get the most out of it. But, since building competencies isn’t something that happens overnight it could also be that this is something evolving right now and that we in a future study would find another result.

8.4 Has the use of agile principles and practices made Project B more visible to team members, other teams and project management, compared to Project A?

Increased communication and project visibility are shown benefits of agile software development [4] [7]. Since Project B has adopted agile principles and practices to a larger extent we – based on the conceptual framework presented in Chapter 3 – expected Project B to be more visible than Project A. This was also the case.

Based on the survey data, the agile ways of working in Project B has considerably increased the project visibility on a project level compared to Project A. The members of Project A don’t consider themselves very aware of the statuses of the other project teams and the entire project’s status, while the members of Project B do. Project B is in this sense more visible than Project A. This supports the findings of [4], [7] and [34]. It can be assumed that the iterative and incremental development process in Project B has contributed to the higher level of project visibility. The shorter development cycles and the constant delivery of value should have made it easier for project members to grasp the status of the project. We also see that the larger organization with different sub-projects in Project A has made it more difficult for the project members to embrace the whole project.

Nonetheless, there was no statistically significant difference between the two projects in how aware the project members considered themselves of their own team’s status. Teams have been used as an organizational unit in Project A for a long time and this indicates that the teams are transparent, whether the project is agile or not. Neither could a statistically significant difference between Project A and Project B be found in how important the project members thought that agile practices are for their project visibility. Demos, iteration planning, task board and continuous integration seem to increase project visibility equally much in Project A and Project B. This indicates that the benefit of increased project visibility could be used by implementing these practices also in legacy project, where the organization and product may not be fully adapted to agile software development.

The number of respondents with each functional role was unfortunately too small to statistically analyze whether project visibility is increased for both team members and project management.

Increased project visibility is usually mentioned in positive terms as it increases the accuracy of forecasting, project planning and estimates of remaining work. It can safely be assumed that Ericsson benefits from these aspects of it and increased visibility is therefore considered a positive effect of agile software development. But, the increased visibility to stakeholders external to the company also raises questions about the protection of intellectual property and company secrets. The success of a close collaboration relationship with a reference customer, as in the case of Project B, relies on trust and full openness is important. But, we think that the possible risk of
opening up should also be considered. It is for example possible that an on-site customer gains additional information about the company, products and ways of working from merely being at the company site. This information could be crucial to the company’s competitive advantage. What happens then if the reference customer choses to collaborate with the competition the next time?

8.5 Has the use of agile principles and practices increased the pressure and stress employees feel in Project B, compared to Project A?

The theoretical background in Chapter 2 presented evidence that several agile practices increase the negative stress and pressure in software development projects, namely demos, daily meetings and sign up. As Project B has implemented these practices to a larger extent than Project A we expected to find a higher level of perceived negative stress and pressure in Project B.

The average levels of negative stress and pressure stated by the survey respondents from Project B are all slightly higher than the average levels stated by members from Project A. But the difference was not statistically significant. The survey respondents from both projects felt moderately stressed and pressured in a negative way by their project work. Further, the negative stress caused by iterative development, demos and sign-up, as reported by [8] and [23], appeared to be present to a moderate extent inside the projects. But neither in this case could a statistically significant difference between the two projects be found. Worth to notice is though that an increased level of negative stress caused by agile software development was mentioned in the open-ended question by one survey respondent from Project B. No extensive conclusions can be made from this fact as the respondents were free to write what they wanted, but it may be an indication that the inherent stress factors of agile principles and practices are worth further investigation.

The negative stress caused by a pressure to report progress on the daily meetings, reported by [11], does not seem to be present to a large extent in nor Project A nor Project B. This is positive.

Studying the overtime metrics, one can see that overtime density is higher in Project A than in Project B. In fact the overtime reported in Project B is almost 50% lower than in Project A. This is an indication that the iterative and incremental way of working has led to less overtime, as hoped for by project management. On the other hand the overtime is more concentrated in Project B, compared to Project A. This indicates that there are parts of Project B that from time to time need to put in extra hours to get the work finished in time. The time before demos, especially demos to the customer, is believed to be such an occasion. This would then explain the relatively high level of stress caused by the demos reported by the survey respondents in Project B. Overall, little overtime is reported in both projects.

In general, it seems like the use of agile principles and practices doesn’t increase stress and pressure to a great extent in Ericsson. However, stress is something very individual and the low number of respondents in Project A may have affected our findings in this area more than others. The overtime metrics were presented for the entire projects.
Not to forget is that the survey also showed that the level of negative pressure in the two projects is moderate. A moderate level of stress is maybe expected in a software development project as it is a business with tight deadlines and high uncertainty, but as the survey questions explicitly asked for negative pressure and stress in, we believe that this issue is something to investigate further. Especially the demos seem to cause negative stress and we therefore believe that a discussion on the pressure and stress created by demos could be a good starting point for improving the situation. The people are a software development project’s most important asset and to take care of them is therefore very important.

8.6 Has the use of agile principles and practices increased coordination effectiveness in Project B, compared to Project A?

From the survey data we can say that Project B has a higher level of coordination effectiveness calculated with the model of Strode et al [8] than Project A at a significance level of \( \alpha=0.10 \) \((p=0.052)\). There was also a significant difference between Project B and Project A in the extent the members felt that they knew what the other teams were doing. This makes us conclude that Project B has achieved a higher both inter-team and intra-team coordination effectiveness than Project A.

The background to the higher coordination effectiveness in Project B is believed to be related to the use of agile principles and practices. The most important explanation is most likely the use of whole teams in Project B and the fact that these teams are responsible for features instead of architecture layers, as this has considerably reduced the number of internal dependencies that needs to be coordinated. Project B has reduced both the handovers between functional departments and within them, compared to Project A. We further believe that the design of the open office space in Project B could explain part of the increased coordination effectiveness. While the members in both projects sit together in an open office space, the cross-functional teams sit together in Project B, and as such every team member is in the proximity of most persons they need to coordinate with. This is in line with the concept of proximity that Strode et al [8] described as an important coordination mechanism.

Additionally it can be seen that the daily meetings form an important coordination mechanism, as proposed by [8] and [24]. Stand-up meetings (daily scrums) are used in both projects and the members from both projects thought they got much help from these in knowing what their team members are doing. Project B also uses the Scrum of Scrum, with the purpose of coordination, which could be assumed to contribute to the higher coordination effectiveness in Project B. Regarding information radiators, or more exactly task boards, they seem to be increasing coordination effectiveness inside the team, but not between teams. No significant difference in this was found between the projects and neither in Project B, where information radiators are more consistently used by all teams, did the project members feel that it helped them to know what the other teams are doing to a large extent.

In summary, our results support the findings of Strode et al [8] and we can say that agile principles and practices improve coordination effectiveness. One shouldn’t forget that Project A is still believed to be able to coordinate effectively but it is believed that this coordination is achieved with a large overhead (see section 6.2.7). That the same – or even a higher – level of
coordination effectiveness can be achieved without the overhead, by using agile ways of working, is therefore be a positive finding for Ericsson.

While the difference in size between the two projects may also affect the coordination effectiveness – Project A is four times as large as Project B – they are both large-scale projects with hundreds of project members and tens of teams that all need to coordinate and deliver into the same software system. We therefore believe that the scale is not the important differentiator in this matter.

8.7 Has the use of agile principles and practices increased productivity in Project B, compared to Project A?

One of the goals with implementing agile principles and practices in Project B was, according to project management, to produce an equivalent product as Project A in a more efficient way. The data indicates that this is achieved, but similarly to when we conducted the literature review it has been hard to find definitive evidence that agile ways of working increase productivity.

In both projects the survey respondents rated the productivity as quite high. No significant difference could be found in how the members of Project A and Project B rated their own productivity or the productivity of the whole project. There was a difference though, in how they rated the productivity of their team: the members of Project B rated their team’s productivity significantly higher than the members of Project A did. This strengthens the impression that the team is a more important unit in Project B than in Project A, which supports the agile way of thinking.

When looking at the work item cycle times, it is evident that the cycle times for higher level work items is shorter in Project B. However, for tasks and sub-tasks, the cycle times are shorter in Project A. It should be noted that the work item cycle times for Project A is only computed for the agile teams, as they are the only teams using the system from where the data is collected. In this way, both data sets are stemming from teams with a large adoption of agile practices, and the difference can in consequence not really be traced to a difference in ways of working. That the cycle times for lower level work items is even shorter in Project A than in Project B is most likely caused by the higher level of automation and standardization in Product A.

The bug cycle times tell another story. The average bug cycle time in Project A is thrice as long as the average bug cycle time in Project B, or almost twice as long as the average for the feature teams in Projects B. The trouble report cycle time is also much shorter in Project B, but since there are only two trouble reports in the Project B data, their significance is very low. It is also evident that the bug cycle time of the agile teams in Project A is about as long as the bug cycle time for the feature teams in Project B, which further strengthens the impression that, when it comes to bug cycle times, the agile practices implemented in the teams are making the process more efficient. For example, continuous integration, facilitates for detecting bugs and integration errors faster, which contributes to shorter bug cycle times.

In both projects the survey respondents rated their individual productivity or the productivity of their team higher than the productivity of the entire project. This could be seen as an indication
of productivity loss due to coordination needs. Since the agile ways of working is believed to improve coordination effectiveness in Project B, compared to Project A, as described in section 8.6, this further strengthens the impression that the agile ways of working has increased productivity in Project B. The less overhead spent on coordination, the more effort can be spent on value creating activities.

Our initial proposition was that the use of internal software documentation would affect the productivity of the project – the more time spent on documentation the less time spent on value-adding activities (see section 3.2.2). One survey respondent also wrote in the open-ended question that he or she thinks that too much time is spent on documentation. But as the members from both projects spend relatively little time daily on producing documentation, we can’t support this proposition and don’t believe that internal software documentation affects productivity to a great extent in either of the projects.

Interestingly enough, several respondents from both projects mentioned productivity in the open-ended survey question. Several of them thought that “time is lost on planning” when working in an agile way. It is true that the recurrent meetings associated with agile ways of working, e.g daily meetings, retrospectives and iteration planning meetings, take quite a lot of time. But we wouldn’t say that this time is actually “lost” as we have found both in earlier research and in our own study that these meetings also give a lot of positive effects, among them improved project visibility and improved knowledge sharing. As always, balance is best.

To summarize, we have found a difference in the productivity of agile teams compared to teams with a lower adoption of agile practices both inside Project A and when comparing Project A and Project B. We therefore believe that the agile teams (characterized by the use of whole team and incremental and iterative development) have a higher productivity. In this way, we agree that agile practices increase productivity, as found by Ilieva et al [30] and Layman et al [31].

8.8 Has the use of agile principles and practices increased the quality of the software produced in Project B, compared to Project A?

When looking at the quality metrics, they are in favor of Project B. The defect density is much lower, and the test case pass rate is consistently higher. Although the test case data reflects different testing stages and different testing approaches, the system stability tests from Project A are somewhat comparable to the acceptance tests done in Project B. Even if we disregard the comparability of the different kinds of testing, the pass rates in themselves speak a clear language. The consistently higher test case pass rates we see in Project B shows how the use of continuous integration, as it allows for a much faster cycle of receiving feedback on tests and fixing found defects, contributes to a higher software quality.

The higher defect density in Project A can in part be explained by the different ways of handling defects in the two projects. In Project B, to a higher extent than in Project A, defects are fixed on the spot by the same person that found them and not reported in the system. However it’s not likely that the entire difference could be explained by this and it’s therefore concluded that the metrics are revealing a higher actual defect density in Project A. A possible reason for the higher defect density in Project A is the complex product architecture, with features spanning over a
large amount of components. It’s not impossible that it simply is harder to develop Product A without causing defects. In Project B, the cross-functional teams are developing one feature each, which should make it a lot easier to develop a feature without introducing defects into other parts of the product. It should also be easier to fix any defects introduced, for the same reasons.

The members of Project B didn’t rate the quality of the software produced in their project as significantly higher than the members of Project A. There could be several possible reasons for this. First of all, the quality of the software produced in Project B could actually be the same as in Project A, but the metrics tell another story. Second of all, the modesty of the survey respondents could have made them answer a lower number than they actually think. A third possible reason is the size of the projects, which makes it hard for individual project members to assess and thereby report the whole picture of the software quality. It could also be related to the different product ages of Product A and Product B. Product A on the one hand, has been released in many versions to many customers and has been tested many times. Product B on the other hand, hasn’t been released and tested by customers yet. It could therefore be that people feel more certain of the software quality in Project A than in Project B.

All in all, we see clear indications that the software quality is higher in Project B than in Project A, and that the agile practices, especially continuous integration, has a part in explaining this difference. However we must also say that the different architecture of the products and the way trouble reports are handled also had an impact on the quality of the software. Then again, it has been said that Project B tries to incorporate the principles from the Agile Manifesto rather than individual practices. In this case, as the lower quality in Project A has been caused partly by slower, document-based communication and a more plan-based approach to testing, it’s concluded that the agile principles employed in Project B has led to a higher software quality.

8.9 Has the use of agile principles and practices made Project B more successful than Project A?

It is the hope of the project management that Project B will be able to accomplish the same thing as Project A, while using fewer resources.

Looking at budget reports, Project A overshot its budget by 2.2%, while Project B overshot its budget by 7%. Although the numbers for Project B are based only on two months, it’s clear that the project has had problems staying within budget. However, both projects have had ample swings from month to month, and it’s quite possible that the overall outcome of Project B would be similar to Project A, had the available data spanned a longer period. Nevertheless, Project B overshot its budget by a bit too much, while Project A stayed within the acceptable range.

The reasons for overshooting a budget can be many, and as we have said earlier, it’s not easy to budget a growing organization. Nevertheless, we were looking for ways in which agile principles and practices have helped Project B to stay within budget, and we haven’t found any.

Project A had restrictions on a large portion of its features at the time of release, and was far from entirely done when it was supposed to be. Most of the restricted features had not been verified enough, but some of them simply weren’t done. This shows that Project A had problems
getting the product ready on time. The use of restrictions in this way is nothing new in Project A and its previous instances, but it’s still not a good sign that a project isn’t done when it’s supposed to be. In this matter, no comparison can be made with Project B, since it has yet to come to a stage were this metric can be used.

Project A stayed reasonably within budget, but didn’t deliver on time what it was supposed to. Project B showed a tendency to overshoot its budget. Project A might be considered very successful by using other measures such as customer satisfaction or return on investment, but that was not possible for us to investigate. Project B has yet to release anything to its customers. Therefore we must say that we have not found any indications that the agile practices and principles used in the projects has helped bring them towards a successful outcome, when using our success rate metrics. However, we haven’t found anything indicating the contrary either.
9 Conclusions

This chapter presents what conclusions can be made regarding the impact of using agile principles and practices in the large-scale software development projects, Project A and Project B.

This study investigated the impact of using agile principles and practices in two large-scale software development projects – the largely plan-based project Project A and the entirely agile project Project B – within eight different areas: internal software documentation, knowledge sharing, project visibility, pressure and stress, productivity, software quality and project success rate. We can conclude that using agile principles and practices has a positive impact in several of these:

*Agile principles and practices lead to a more balanced use of internal software documentation*

We saw a reduction of the number of individual documentation items where the amount was perceived as too large in Project B, compared to Project A, and can derive this to the use of agile principles and practices, particularly *whole team*. The overall amount was also perceived as just right, most likely thanks to general company documentation policies. In consequence we conclude that the use of agile ways of working has led to a more balanced use of internal software documentation when supported by sound documentation policies.

*Agile principles and practices contribute to knowledge sharing*

We found that agile principles and practices contribute to knowledge sharing, but no difference could be found between the projects – knowledge was shared with the same ease – why other, to us unknown factors are believed to also be important when it comes to knowledge sharing.

*Agile principles and practices increase awareness of the big picture*

Agile practices, particularly *continuous integration*, were clearly shown to increase the project members’ awareness of the big picture. The members of Project B perceived a significantly higher visibility of the status of the other teams and the entire project, compared to the members of Project A. We also saw higher coordination effectiveness in Project B.

*Agile principles and practices achieve effective coordination, with less overhead*

The implementation of agile practices in Project B, especially its use of *open office space, whole team* and having the teams responsible for features was shown to substitute the coordination overhead necessary in Project A. We also saw higher coordination effectiveness in Project B.

*Agile principles and practices increase productivity*

The agile ways of working has, where implemented, improved productivity in both projects, especially so the defect fixing efficiency. In Project B the use of *iterative and incremental development*, the less complicated defect handling process and the use of *whole teams* was shown to be particularly important for the productivity increase. Therefore we conclude that using agile principles and practices increase productivity.
**Agile principles and practices possibly increase software quality**

Indications are that the software quality is higher in Project B than Project A. Some of this can be derived to the use of agile ways of working, and we do therefore conclude that agile principles and practices have possibly increased software quality. However, no final conclusions can be made before Product B has been released to customers.

Within some of the investigated areas no impacts could be found. The members of Project A and Project B are moderately stressed by their project work. But it doesn’t seem like agile principles and practices increase the level of pressure and stress. Finally, we couldn’t find any indication that agile ways of working has improved project success rate, concerning on-time release of features and budget performance. That these effects of agile principles and practices weren’t found doesn’t necessarily mean they don’t exist, only that we couldn’t see them.

We were from our study also able to draw some additional conclusions:

**Internal software documentation is important and cannot fully be replaced with face-to-face communication**

Internal software documentation was found to be very important in both projects, especially to communicate technical details. Therefore we conclude that the agile belief that face-to-face communication is the most efficient way to convey information is not true for the large-scale software development projects in Ericsson.

**Agile principles and practices give positive impacts also when partially implemented**

A positive impact of several agile principles and practices was visible in Project A. This indicates that a partial implementation also can give positive results. However, we see that the full project implementation of the practices in Project B give stronger results.

**It is feasible to implement agile principles and practices in large-scale software development**

In summary, the way agile principles and practices are implemented in Project A, but especially in Project B, have been found to have many positive effects. We believe this provides evidence that agile ways of working can be beneficial in large-scale implementations.
10 Discussion

The report is closed with a discussion of the found results and the research method. A revised conceptual framework, which illustrates what impact of agile principles and practices we could find is presented and the possibility of transferring the results to other software development projects within Ericsson and to other software development organizations is discussed. Finally some ideas for future work are presented.

10.1 Results and research method

We concluded that the agile principles and practices have had an impact on several areas in both studied projects, and especially in the more agile one - Project B. The found effects were all positive, which shows that it is beneficial to use agile principles and practices in (large-scale) software development projects. In fact we didn’t see any of the suggested negative effects. This is a pleasant result, since it is our impression that software development projects throughout the years have experienced many challenges and it’s positive if agile principles and practices can bring improvements. We also think it is positive to see that large organizations such as Ericsson dare to make major changes in their ways of working and we are satisfied to have been able to show that this change was for the better.

However, we also think that the lack of indications on some of the expected effects is interesting, as we have the impression that agile software development many times promises heaven and earth. A common saying is that there are no quick fixes and that seems indeed to be true for software development. Project B had to change its project organization and product architecture to gain the full benefits of using agile principles and practices and this has required a lot of time, effort and money.

We think that another interesting finding is the lack of empirical evidence to the claimed effects of agile principles and practices in earlier work and we also experienced this challenge in our own study. It wasn’t hard to find people who believed that agile principles and practices give a certain effect, but it was hard to back that claim with evidence.

Another observation, which would explain that many of the effects of agile principles and practices were present also in Project A, is that many things now collected under the name “agile principles and practices”, are in fact not that new. The daily scrum is just a standardized form of the morning meeting held in many companies for decades. The product backlog is just a formalized to-do-list.

We chose to use a qualitative research method and do a multiple-case study and think that this was a very good choice. We believe that we have been able to gain a deep understanding of the ways of working in the two projects and that this was necessary to be able to draw sound conclusions. We don’t think that a quantitative study would have been able to gain the same results.

Our initial plan was to conduct in-depth interviews with selected project members at the project site. However, because of budget constraints within Ericsson we didn’t have the possibility to visit the project and therefore decided to conduct a web-based survey instead. Afterwards, we’re
happy with this turn of events and believe that a survey after all was a better way to capture the broad picture of the two projects. On its own a survey probably wouldn’t have provided a deep understanding, but in combination with the personal communication we had with several project members, the internal documents and the calculated metrics, we believe that we were able to get a good picture of the impact of agile principles and practices in the two projects.

We couldn’t follow our initial plan regarding the metrics to a hundred percent, but we put in a lot of effort to collect what we could and we consider that the metrics we finally managed to gather were enough to support the other data sources. To have full control over the measurement situation we would have wanted to gather the information the metrics were calculated from ourselves, but this wasn’t practically possible.

Several of the areas we studied are abstract concepts, e.g. knowledge sharing or coordination effectiveness and it was challenging to measure them. After evaluating different ways we decided to measure them through the survey, since they are all concepts perceived by people and for example not characteristics of a product. It is possible that the abstractness has affected the survey results, but we strived to concretize the concepts and investigate them from different angles. For coordination effectiveness we used an existing theory to do this. Noteworthy is that there was a significant (at $\alpha=0.01$) correlation between the level of coordination effectiveness derived from the variables in the model and the one extracted from the direct question in the survey.

We believe that the survey in Project B is very representative for the whole project as it was responded by almost half of the project members and the dispersion in age, experience and functional roles was satisfactory. In Project A the survey did also gain a satisfactory dispersion, but the sample size and the number of respondents represent a much smaller portion of the entire project. However, as the sample was made randomly and the respondents more or less agreed with each other (the standard deviation of most questions was below two) we still believe that the survey results can be used to represent the entire project. We have no reason to believe that the result from a larger sample would have deviated much from this one. Nevertheless a larger sample and a higher response rate would of course have strengthened the validity of the results.

The metrics were collected for the entire projects. We have further seen no indications that the internal documents we studied weren’t representative for the whole projects and when communicating with the key project members we were careful to use the full project names and not, for example, the sub-project names in Project A. Therefore the results should be considered representative for the entire Project A and Project B.

All in all, we believe that we have used a rigorous research method and hope that we have managed to confirm some benefits and highlight some challenges of using agile principles and practices in large-scale software development projects, as Ericsson requested.
10.2 Revised conceptual framework

The conceptual framework presented in chapter 3 presented what effects and outcomes we, based on the theoretical framework believed to find in the studied projects. In this section we present a revised conceptual framework of what effects and outcomes we found and what agile practices contributed to or counteracted these.

<table>
<thead>
<tr>
<th>Found effects</th>
<th>Found outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balanced use of documentation</td>
<td>Increased productivity</td>
</tr>
<tr>
<td>Facilitated knowledge sharing</td>
<td>Contributing practices: iterative and incremental development, whole team, continuous integration</td>
</tr>
<tr>
<td>Increased project visibility</td>
<td>Higher quality</td>
</tr>
<tr>
<td>Contributing practices: open office space, daily meetings, whole team, demo, iteration planning.</td>
<td>Contributing practices: continuous integration</td>
</tr>
<tr>
<td>Pressure and stress</td>
<td>Contributing practices: iterative development, sign-up, daily meetings, demo.</td>
</tr>
<tr>
<td>More effective coordination</td>
<td>Contributing practices: iterative and incremental development, whole team, continuous integration</td>
</tr>
</tbody>
</table>

Figure 19. Revised conceptual framework.

In contrast to what the earlier work suggested, we saw that agile principles and practices had led to a balanced use of internal software documentation, not an insufficient amount.

We didn’t find that agile principles and practices increased the overall presence of knowledge sharing in a project, as proposed by the theoretical framework. However we found support for our theoretical findings that several agile practices contribute to knowledge sharing, specifically
open office space, daily meetings, whole team, demos and iteration planning. The results were dubious regarding retrospectives, why we can't say whether this practice contributes to knowledge sharing or not.

We concluded, in line with our theoretical framework, that agile principles and practices, increase project visibility and that iteration planning, open office space, demos, task boards and continuous integration contribute to this. We found no support for iterative development.

No increase in the overall levels of pressure and stress was found in the projects we studied. Nevertheless, some agile practices, namely iterative development, sign-up, daily meetings and demo, were perceived as stressful by the project members, as suggested by earlier work.

Agile principles and practices were found to increase coordination effectiveness in line with what was proposed in the theoretical framework. Daily meetings, open office space and whole team were also found to contribute to coordination effectiveness. Additionally we found that task boards contributed, which more or less corresponds with the information radiators proposed in the theoretical framework.

No support for the assumed relationships between the effort spent on producing internal software documentation, coordination effectiveness and productivity could be found.

We did find support that agile teams, using incremental and iterative development, whole team, and continuous integration had a higher productivity than less agile teams, but we didn’t find any indications that empowerment increases productivity.

We found no support for the theoretical finding that demos contribute to software quality. However, we found indications that continuous integration can improve software quality.

No indications that agile principles and practices increase project success rate could be found. However, the data available to study this area was very limited.

10.3 Transferability
The results of a case study should be considered in terms of analytical generalization and the cases to which the results are generalizable should be identified with help of the theoretical framework [40]. We have chosen to base our transferability discussion on the areas in our revised conceptual framework.

Balanced use of documentation
It is our conclusion that the use of agile principles and practices leads to a more balanced amount of internal software documentation being produced, but we also believe that this balance is caused in part by the documentation policies used by Ericsson in general. Therefore we believe that other software development projects in Ericsson would find similar results should they adopt the same agile principles and practices. However, we also consider it possible that other software development organizations, without a similar documentation policy, may find that the adoption of agile principles and practices can lead to a lack of available internal software
documentation. Hence, we find it advisable for other software development organizations that wish to apply agile principles and practices to carefully review their documentation policies and the need for internal software documentation, especially from a large-scale point of view.

Increased project visibility, more effective coordination and facilitated knowledge sharing
The application of agile practices has a positive effect on project visibility in both projects, but especially in Project B. Further, we saw that the agile practices contribute to a high level of coordination effectiveness in Project B, without the coordination overhead that is necessary to maintain a lower level of coordination effectiveness in Project A. While we didn’t find a general increase in knowledge sharing caused by the application of the agile principles and practices, we saw that the studied practices contribute to knowledge sharing in both projects. We therefore believe that agile principles and practices is a way to obtain a sharing of knowledge in a project, but far from the only way to accomplish it.

We did not find that increased visibility or knowledge sharing through the application of agile principles and practices was hindered, or helped, by specific aspects in Project A or Project B. The level of coordination effectiveness is however believed to be impacted by the co-location of all studied teams. Project A has got 480 off-shore consultants, but they weren’t included in our study. Consequently, we believe that these effects can be obtained in other co-located software development settings in Ericsson and other organizations by application of the same principles and practices. Nevertheless, as project visibility and coordination is of a special interest in large-scale software development, it might very well be true that the positive effects we found are not perceived as important to the same extent in smaller settings.

Pressure and stress
We saw that some agile practices, especially demos, were perceived as stressful by some project members. We have no reason to believe that similar results would not be found in other software development projects in Ericsson. However, stress is a very subjective area, and it is fully possible that other people in other settings would find very different results.

Increased productivity and higher quality
We found that the agile practices, especially continuous integration, contribute to both an increased productivity in the agile teams of both projects, and a higher quality in Project B. Neither in this case did we find any special circumstances that make us believe our findings to be uniquely applicable to the projects we studied, but that the results could be similar in other Ericsson projects. However, there are many different aspects of and approaches to software quality, and there are certainly other software development projects and organizations that have found other ways to produce high quality software.

Further considerations
The people working in both the studied projects represent a wide variety of software development professionals, with varying amounts of experience from both agile and non-agile software development. They also vary widely in the amount of time they have been working for Ericsson. Since we saw that both projects benefited from the use of agile principles and practices,
we don’t believe that a certain amount of or time of employment is needed for being able to find value in the application of the same agile software development methods.

We saw that the practice of *continuous integration* was more useful in Project B, where the product architecture had been designed with this practice in mind, which leads us to believe that it can be made more useful for the development of a software product that has a suitting architecture. However, we also saw that *continuous integration* was useful in the parts of Project A where it was used. This means that *continuous integration* is still useful even if it is not implemented end to end, and even for a product with a less suitable architecture.

Further, we found that agile principles can have positive impacts even when they are not fully implemented. This means that software development organizations that in some aspect are deemed not suitable for a full implementation of agile principles and practices still can benefit from a partial implementation, where it is suitable. However, it’s also important to note that what constitutes a full implementation of agile principles and practices, and what makes an implementation partial, is a highly subjective topic. Further, there are of course several different ways to implement a single practice. We have not described the way every single practice has been implemented in the studied projects. It’s possible that a different implementation of a practice or principle would lead to different results – it’s even likely. What we believe is important is that the practices are implemented in such a way that they are compatible with the rest of the project, and with the desired effect in mind.

From a technical viewpoint there are some important similarities between the two projects. The developed products’ functionality is essentially the same, and the technical development environment is also similar. There are also differences; especially in the software architecture and dependencies between components. However, it is our impression that the similarities are more important than the differences. Further, some of the impacts we’ve found, such as increased project visibility, aren’t dependent on the technical characteristics of a software development project. We therefore believe that the found effects would be very similar if the same principles and practices where adopted in other projects in PDU X. We also believe that the projects share technical attributes with many other projects in Ericsson in general, and Development Unit Radio in particular. However, when considering software development projects outside Ericsson, as the technical differences grow, the possible impact of said differences becomes hard to assess. It’s possible that a different software development project in another company would get other effects from the same practice, caused by technical differences. There might also be aspects, such as external requirements of a specific quality control or documentation process, which impedes or makes it impossible to implement certain practices. In such cases, the same effects can not be expected since the same application of the practices is impossible.

We have not studied any cultural aspects of the members of the two projects, other than that they are mostly coming from the same country and speaking the same language. Therefore, we cannot say that our findings are universally applicable to software development projects in another country. However, the studies found in the literature review has drawn conclusions from a wide variety of software development projects around the world, and we have not seen any
pattern in which theoretical findings from a certain culture have been harder to verify in the projects we have studied.

10.4 Ideas for future work

There were several areas in our theoretical framework that we didn’t have the possibility to study and we think that all of them are of interest. We also believe that several of the areas we did study need further investigation to enable deeper conclusions on the impact of agile principles and practices in large-scale software development projects, in Ericsson and in other software development organizations.

We have touched the area of communication in our analysis and also by studying internal software documentation, knowledge sharing, project visibility and coordination. But we didn’t do a thorough study of the area. Most of the evidence on improved communication in the theoretical framework came from small projects with few project members; it would be interesting to see if the same effect is present in projects with hundreds of members.

We didn’t explicitly study job satisfaction, but found that the employees of Project A and Project B were very satisfied with the agile practices implemented in their project. In Project B they were also very satisfied with the overall ways of working in their project. It would be interesting to investigate if a causal relationship between these two aspects exists also in Ericsson, as there were strong theoretical indications of this.

The possible existence of architectural problems, especially in the long term, due to the use of agile principles and practices didn’t fit the scope of this thesis. But we believe that this aspect is very important and interesting for Project B in particular, and large-scale software projects in general. Product B is supposed to replace Product A, which has been in use for over a decade. It is also a very important software product for Ericsson. There is consequently a large need for the architecture of Product B to be robust and sustainable. If it were to erode due to the use of agile principles and practices, this could cause problems for Ericsson in the future.

Our findings have indicated a certain level of stress among the members of both projects, but to us it is not entirely clear what is causing it. It is possible that the stress level of employees is already investigated in the yearly employee surveys that are done globally at Ericsson, but if it is not we think it might be a good idea to further investigate what is causing this stress and how it is affecting the employees.

We weren’t able to draw any conclusions regarding project success rate, partly because of lack of data. It would be interesting to study the success rate of Project A and Project B in a longer time frame, to get a better picture if agile principles and practices do improve it. We also believe it would be interesting to study knowledge sharing during a longer time. The theoretical framework proposed that the use of agile principles and practices may lead to loss of employee capabilities in the long term, and if this is true it could cause competitive problems for Ericsson in the future.

Initially, we had an idea that we could classify and count the different kinds of documentation used in the projects, to get a better picture of how the agile practices are impacting the use of
documentation. We did not have time to do this, but it could be worth the time in a future study, as a complement to what the survey respondents said. Possibly it would give new insights to what types of documentation is reduced by the use of agile principles and practices. 

We were also planning to study productivity and its underlying components more thoroughly, but there wasn’t enough time. Our proposition is that productivity depends on two factors: the amount of time the developers can spend on value-creating activities and their ability to produce value during this time. We propose that each component is further determined by different factors. The amount of value-adding time is for example likely affected by the time spent on coordination, internal software documentation, recurrent meetings and solving problems with development environments, as well as motivation. The value-adding capacity is possibly determined by the developer’s individual capacities and capabilities for a certain work task, his or her level of experience, and also by the effectiveness of the ways of working used, e.g. their ability to reduce waste and rework. In our study we could see indications that coordination overhead and the developers capabilities affect productivity, why the issue becomes even more appealing as a topic for future work.

Product B hasn’t been released to customers yet. We’re very curious to see how the project manages this challenge and if the indicated higher software quality is also perceived by the customers.

Finally we have just studied two software development projects in one company, in one country. To better understand the effects of agile principles and practices in different contexts, more studies are certainly needed.
11 Bibliography


12 Appendices

12.1 The systematic literature review

12.1.1 Study selection criteria

• Language: English, Swedish, Spanish
• Date of publication: 2005 and forward
  AND
• (Setting: industrial (not student)
• Research design: case, experimental, survey or action
• Presentation of empirical evidence )
  OR
• Presentation of theory or model for studying effect of an agile software development method

12.1.2 Data extraction protocol

<table>
<thead>
<tr>
<th>Study characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>Publication year</td>
</tr>
<tr>
<td>Study year</td>
</tr>
<tr>
<td>Type of study</td>
</tr>
<tr>
<td>Research method</td>
</tr>
<tr>
<td>Research questions</td>
</tr>
<tr>
<td>Number of cases/respondents</td>
</tr>
<tr>
<td>Description case/respondents</td>
</tr>
<tr>
<td>Agile practices studied</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Study findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reported positive effects</td>
</tr>
<tr>
<td>Reported negative effects</td>
</tr>
<tr>
<td>Outcome (observed or referenced)</td>
</tr>
<tr>
<td>Comments/Other findings</td>
</tr>
</tbody>
</table>

12.1.3 Studies found in the literature review

<table>
<thead>
<tr>
<th>No.</th>
<th>Research method</th>
<th>Publication year</th>
<th>Number of cases/respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Survey</td>
<td>2011</td>
<td>&gt;1000</td>
</tr>
<tr>
<td>S2</td>
<td>Multiple case study</td>
<td>2012</td>
<td>3</td>
</tr>
<tr>
<td>S3</td>
<td>Single case study</td>
<td>2011</td>
<td>1</td>
</tr>
<tr>
<td>S4</td>
<td>Survey</td>
<td>2010</td>
<td>576</td>
</tr>
<tr>
<td>S5</td>
<td>Survey</td>
<td>2006</td>
<td>459/936</td>
</tr>
<tr>
<td>S6</td>
<td>Multiple case study</td>
<td>2012</td>
<td>3</td>
</tr>
<tr>
<td>S7</td>
<td>Single case study</td>
<td>2012</td>
<td>1</td>
</tr>
<tr>
<td>S8</td>
<td>Survey + interviews</td>
<td>2011</td>
<td>79</td>
</tr>
<tr>
<td>Study</td>
<td>Type</td>
<td>Year</td>
<td>Authors</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>S9</td>
<td>Single case study</td>
<td>2009</td>
<td>1</td>
</tr>
<tr>
<td>S10</td>
<td>Single case study</td>
<td>2010</td>
<td>1</td>
</tr>
<tr>
<td>S11</td>
<td>Multiple case study</td>
<td>2008</td>
<td>2</td>
</tr>
<tr>
<td>S12</td>
<td>Survey</td>
<td>2010</td>
<td>129</td>
</tr>
<tr>
<td>S13</td>
<td>Survey + multiple case study</td>
<td>2010</td>
<td>505/10</td>
</tr>
<tr>
<td>S14</td>
<td>Single case study</td>
<td>2010</td>
<td>1</td>
</tr>
<tr>
<td>S15</td>
<td>Survey</td>
<td>2010</td>
<td>781</td>
</tr>
<tr>
<td>S16</td>
<td>Multiple case study</td>
<td>2008</td>
<td>2</td>
</tr>
</tbody>
</table>

### 12.1.4 List of studies


## 12.2 Project A survey questions and result summary

Table 15. The number of respondents, standard deviation, mean, median, minimum and maximum value for all questions in the Project A survey.

<table>
<thead>
<tr>
<th>Question</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Are you employed by Ericsson or working as an external consultant?</td>
<td>24</td>
<td>8.36</td>
<td>8</td>
<td>6.95</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>- Ericsson employee</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- External consultant</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Other</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Including consultant work, for how long have you been working at</td>
<td>25</td>
<td>8.36</td>
<td>8</td>
<td>6.95</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>Ericsson?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. How much experience do you have including previous employers, from</td>
<td>23</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>working with ... agile software development?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... NON-agile experience?</td>
<td>25</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Are you currently working in Project A? (or any of its subprojects)</td>
<td>23</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Yes</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- No</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Including work as a consultant, for how long have you been working</td>
<td>25</td>
<td>4.84</td>
<td>3</td>
<td>3.40</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>in a Product A development project? (including your time in Project A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Which functional role description(s) correspond(s) with your primary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>work assignments?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Analyst/Architect</td>
<td>5</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designer</td>
<td>8</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tester/QA</td>
<td>4</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project manager</td>
<td>7</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrative</td>
<td>1</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration management</td>
<td>0</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer</td>
<td>8</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum Master</td>
<td>2</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Owner</td>
<td>2</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of roles in Project A*</td>
<td>27</td>
<td>0</td>
<td>1.41</td>
<td>0.57</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7. Are you part of a Scrum team?</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Yes</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- No</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Based on your experiences in Project A, how satisfied are you with</td>
<td>20</td>
<td>4.75</td>
<td>5</td>
<td>1.74</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>the following software development practices?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product backlog</td>
<td>20</td>
<td>4.75</td>
<td>5</td>
<td>1.74</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Scrum boards</td>
<td>20</td>
<td>4.95</td>
<td>5</td>
<td>1.61</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Activity</td>
<td>Number of respondents</td>
<td>Mean</td>
<td>Median</td>
<td>Standard Deviation</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-----------------------</td>
<td>-------</td>
<td>--------</td>
<td>--------------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Daily scrums</td>
<td>Valid: 22, Missing: 5</td>
<td>5.18</td>
<td>5</td>
<td>1.65</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Sprint planning meetings</td>
<td>Valid: 18, Missing: 9</td>
<td>5.00</td>
<td>5</td>
<td>1.88</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Retrospective meetings</td>
<td>Valid: 22, Missing: 5</td>
<td>4.64</td>
<td>4</td>
<td>1.79</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Demos</td>
<td>Valid: 22, Missing: 5</td>
<td>4.64</td>
<td>5</td>
<td>2.01</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Working in sprints</td>
<td>Valid: 20, Missing: 7</td>
<td>5.10</td>
<td>6</td>
<td>1.83</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Self sign up for tasks</td>
<td>Valid: 20, Missing: 7</td>
<td>4.95</td>
<td>5</td>
<td>1.73</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Open office space</td>
<td>Valid: 25, Missing: 2</td>
<td>4.44</td>
<td>5</td>
<td>2.35</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Cross-functional teams</td>
<td>Valid: 23, Missing: 4</td>
<td>4.43</td>
<td>4</td>
<td>1.73</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Valid: 18, Missing: 9</td>
<td>3.89</td>
<td>4</td>
<td>1.94</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>Valid: 22, Missing: 5</td>
<td>4.55</td>
<td>5</td>
<td>1.85</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

9. How satisfied are you overall with the ways of working in Project A?
   Valid: 27, Missing: 0
   | 5.41 | 6 | 2.31 | 1 | 9 |

10. How would you rate...
   ... your own productivity in Project A?
      Valid: 27, Missing: 0
      | 5.63 | 6 | 1.08 | 3 | 7 |
   ... the productivity of your team in Project A?
      Valid: 27, Missing: 0
      | 5.22 | 5 | 1.22 | 2 | 7 |
   ... the overall productivity in Project A?
      Valid: 26, Missing: 1
      | 4.88 | 5 | 1.18 | 2 | 7 |
   ... the quality of the software produced in Project A?
      Valid: 25, Missing: 2
      | 4.72 | 5 | 1.46 | 1 | 7 |

11. If you compare to finding the solution yourself, how often do you...
   ... get help from team members to solve problems related to your project work?
      Valid: 26, Missing: 1
      | 4.77 | 5 | 1.45 | 2 | 7 |
   ... get help from members of other teams in Project A to solve problems related to your project work?
      Valid: 25, Missing: 2
      | 3.80 | 4 | 1.63 | 1 | 7 |
   Level of knowledge sharing in Project A*
      Valid: 25, Missing: 2
      | 8.60 | 9 | 2.63 | 4 | 14|

12. To what extent do you feel that the members of your team have the necessary capabilities to help you with your project work?
   Valid: 26, Missing: 1
   | 5.38 | 6 | 1.42 | 1 | 7 |

13. Considering the last 7 days, which of the following have you performed?
   Requirement specification
      Valid: 8, Missing: 19
      Number of performed activities the last 7 days*
      Valid: 27, Missing: 0
<pre><code>  | 1.67 | 1 | 1.30 | 0 | 5 |
</code></pre>
<table>
<thead>
<tr>
<th>Proposition</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>The daily scrums help my team to solve problems together</td>
<td>21</td>
<td>6</td>
<td>4.67</td>
<td>1.68</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>The sprint planning meetings give me insights about the project I wouldn't get otherwise</td>
<td>18</td>
<td>9</td>
<td>4.33</td>
<td>1.65</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>The retrospective meetings give me insights about the project I wouldn't get otherwise</td>
<td>20</td>
<td>7</td>
<td>3.70</td>
<td>1.69</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>The demos give me insights about the project I wouldn't get otherwise</td>
<td>22</td>
<td>5</td>
<td>4.32</td>
<td>1.84</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>I have gained insights from other functional areas (design, testing, system analysis etc.) than my own from my team members</td>
<td>22</td>
<td>5</td>
<td>4.27</td>
<td>1.86</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

15. What is your opinion about the amount of the following software documentation items in Project A?

<table>
<thead>
<tr>
<th>Specification</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement specifications</td>
<td>22</td>
<td>5</td>
<td>4.59</td>
<td>1.47</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Work packages (user stories, work items, etc.)</td>
<td>23</td>
<td>4</td>
<td>4.30</td>
<td>1.52</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Design specifications (implementation plans, one-pagers, etc.)</td>
<td>21</td>
<td>6</td>
<td>3.67</td>
<td>1.68</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Technical documentation of code (in e.g. Confluence)</td>
<td>19</td>
<td>8</td>
<td>3.16</td>
<td>1.57</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Process documentation (policies, project documents etc.)</td>
<td>22</td>
<td>5</td>
<td>4.64</td>
<td>1.71</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Trouble reports</td>
<td>23</td>
<td>4</td>
<td>4.87</td>
<td>1.49</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Change requests</td>
<td>24</td>
<td>3</td>
<td>4.46</td>
<td>1.29</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

16. What is your opinion about the overall amount of internal software documentation in Project A?

17. How important do you consider internal software documentation for Project A?

18. Based on your experiences from Project A, to what extent do you feel that internal software documentation can be replaced by face-to-face communication?

19. Considering the last 30 days, how much time have you on average spent each day on producing internal software documentation?
<table>
<thead>
<tr>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>20.</strong> To what extent do you consider yourself aware of …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... your team's status relative to its goals and plans?</td>
<td>25</td>
<td>2</td>
<td>5.60</td>
<td>6</td>
<td>1.29</td>
</tr>
<tr>
<td>... the other teams' status relative to their goals and plans?</td>
<td>25</td>
<td>2</td>
<td>3.32</td>
<td>4</td>
<td>1.70</td>
</tr>
<tr>
<td>... the status of the entire Project A relative to its goals and plans?</td>
<td>25</td>
<td>2</td>
<td>3.64</td>
<td>4</td>
<td>1.87</td>
</tr>
<tr>
<td><strong>21.</strong> How important do you consider the following software development practices for your awareness of the project's status?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demos</td>
<td>22</td>
<td>5</td>
<td>5.59</td>
<td>7</td>
<td>1.84</td>
</tr>
<tr>
<td>Sprint planning meetings</td>
<td>20</td>
<td>7</td>
<td>5.60</td>
<td>6</td>
<td>1.31</td>
</tr>
<tr>
<td>Scrum boards</td>
<td>20</td>
<td>7</td>
<td>5.55</td>
<td>6</td>
<td>1.73</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>22</td>
<td>5</td>
<td>5.82</td>
<td>7</td>
<td>1.40</td>
</tr>
<tr>
<td><strong>22.</strong> To what extent do you feel that you are pressured and stressed in a negative way …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... by the fact that your team should deliver results at the end of each sprint?</td>
<td>18</td>
<td>9</td>
<td>3.28</td>
<td>3</td>
<td>1.57</td>
</tr>
<tr>
<td>... to finish what you signed up for during a sprint?</td>
<td>16</td>
<td>11</td>
<td>3.19</td>
<td>3</td>
<td>1.42</td>
</tr>
<tr>
<td>... by the demos?</td>
<td>17</td>
<td>10</td>
<td>3.82</td>
<td>4</td>
<td>1.91</td>
</tr>
<tr>
<td>... by your work in Project A in general?</td>
<td>22</td>
<td>5</td>
<td>3.59</td>
<td>4</td>
<td>1.62</td>
</tr>
<tr>
<td><strong>23.</strong> Do you feel pressured in a negative way to report results of progress on every daily scrum?</td>
<td>17</td>
<td>10</td>
<td>2.94</td>
<td>3</td>
<td>1.95</td>
</tr>
<tr>
<td><strong>24.</strong> Do you feel that you …</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... understand the overall goal of Project A?</td>
<td>26</td>
<td>1</td>
<td>4.88</td>
<td>5</td>
<td>1.84</td>
</tr>
<tr>
<td>... understand how the task you are currently working with fits with the overall project goal?</td>
<td>25</td>
<td>2</td>
<td>5.12</td>
<td>6</td>
<td>1.64</td>
</tr>
<tr>
<td>... know what task you should be working on right now?</td>
<td>26</td>
<td>1</td>
<td>6.19</td>
<td>7</td>
<td>1.06</td>
</tr>
<tr>
<td>... know when that task is required to be finished?</td>
<td>26</td>
<td>1</td>
<td>6.08</td>
<td>6</td>
<td>1.02</td>
</tr>
<tr>
<td>... know what your team members are doing?</td>
<td>26</td>
<td>1</td>
<td>5.50</td>
<td>6</td>
<td>1.24</td>
</tr>
<tr>
<td>... know what the other teams are doing?</td>
<td>26</td>
<td>1</td>
<td>3.35</td>
<td>3</td>
<td>1.70</td>
</tr>
<tr>
<td>... know what capabilities your team members have?</td>
<td>26</td>
<td>1</td>
<td>5.42</td>
<td>6</td>
<td>1.39</td>
</tr>
<tr>
<td>... know what capabilities your project members have?</td>
<td>23</td>
<td>4</td>
<td>4.22</td>
<td>4</td>
<td>1.73</td>
</tr>
<tr>
<td>... can coordinate effectively within</td>
<td>25</td>
<td>2</td>
<td>4.80</td>
<td>5</td>
<td>1.73</td>
</tr>
<tr>
<td>Question</td>
<td>Number of respondents</td>
<td>Mean</td>
<td>Median</td>
<td>Standard Deviation</td>
<td>Min</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>------</td>
<td>--------</td>
<td>--------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Do you feel that the work of others you depend on is ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... available at the right time?</td>
<td>24</td>
<td>3</td>
<td>3.96</td>
<td>1.71</td>
<td>1</td>
</tr>
<tr>
<td>... is available at the correct location?</td>
<td>25</td>
<td>2</td>
<td>4.40</td>
<td>1.85</td>
<td>1</td>
</tr>
<tr>
<td>... the way you need it to be?</td>
<td>25</td>
<td>2</td>
<td>3.92</td>
<td>1.63</td>
<td>1</td>
</tr>
<tr>
<td>To what extent do you feel that the following software development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>practices help you know what your team members are doing?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum boards</td>
<td>15</td>
<td>12</td>
<td>5.33</td>
<td>6</td>
<td>1.80</td>
</tr>
<tr>
<td>Daily scrums</td>
<td>18</td>
<td>9</td>
<td>6.17</td>
<td>7</td>
<td>1.10</td>
</tr>
<tr>
<td>To what extent do you feel that the other teams' scrum boards help</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>you know what they are doing?</td>
<td>15</td>
<td>12</td>
<td>3.93</td>
<td>4</td>
<td>1.79</td>
</tr>
<tr>
<td>Do you want to add something about the ways of working in Project A?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level of implicit coordination effectiveness in Project A*</td>
<td>23</td>
<td>4</td>
<td>5.04</td>
<td>5</td>
<td>1.11</td>
</tr>
<tr>
<td>Level of explicit coordination effectiveness in Project A*</td>
<td>24</td>
<td>3</td>
<td>4.15</td>
<td>4</td>
<td>1.63</td>
</tr>
<tr>
<td>Level of coordination effectiveness in Project A*</td>
<td>21</td>
<td>6</td>
<td>4.65</td>
<td>5</td>
<td>1.15</td>
</tr>
</tbody>
</table>

*Derived variable
12.3 Project B survey questions and result summary

Table 16. The number of respondents, standard deviation, mean, median, minimum and maximum value for all questions in the Project B survey.

<table>
<thead>
<tr>
<th>Question</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Are you employed by Ericsson or working as an external consultant?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Ericsson employee</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- External consultant</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Other</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Including consultant work. for how long have you been working at Ericsson?</td>
<td>47</td>
<td>1</td>
<td>6.55</td>
<td>5</td>
<td>6.16</td>
<td>0</td>
</tr>
<tr>
<td>3. How much experience do you have including previous employers. from working with ... agile software development?</td>
<td>48</td>
<td>0</td>
<td>4</td>
<td>6.16</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>... NON-agile experience?</td>
<td>43</td>
<td>5</td>
<td>0.88</td>
<td>1</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>4. Have you previously worked in any Product A development project?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Yes</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- No</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Which functional role description(s) corresponded with your work assignments in the Product A development project?</td>
<td>14</td>
<td>34</td>
<td>1</td>
<td>1.33</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>System Analyst/Architect</td>
<td>12</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designer</td>
<td>11</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project manager</td>
<td>3</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrative</td>
<td>2</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer</td>
<td>9</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum Master</td>
<td>7</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Owner</td>
<td>0</td>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration management</td>
<td>4</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of roles in Project A*</td>
<td>48</td>
<td>0</td>
<td>1.13</td>
<td>1</td>
<td>1.33</td>
<td>0</td>
</tr>
<tr>
<td>6. Including work as a consultant. for how long have you been working in Project B?</td>
<td>43</td>
<td>5</td>
<td>0.88</td>
<td>1</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>System Analyst/Architect</td>
<td>7</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designer</td>
<td>13</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tester/QA</td>
<td>11</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>Number of respondents</td>
<td>Mean</td>
<td>Median</td>
<td>Standard Deviation</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------</td>
<td>------</td>
<td>--------</td>
<td>--------------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Project manager</td>
<td>1</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administrative</td>
<td>1</td>
<td>47</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration management</td>
<td>4</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer</td>
<td>13</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum Master</td>
<td>7</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Owner</td>
<td>2</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>3</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of roles in Project B*</td>
<td>48</td>
<td>0</td>
<td>1.23</td>
<td>1</td>
<td>0.59</td>
<td>3</td>
</tr>
</tbody>
</table>

**8. Are you part of a Scrum team?**
- Yes: 43
- No: 5

**9. Based on your experiences in Project B, how satisfied are you with the following software development practices?**

<table>
<thead>
<tr>
<th>Practice</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product backlog</td>
<td>45</td>
<td>3</td>
<td>5.00</td>
<td>5</td>
<td>1.65</td>
<td>1</td>
</tr>
<tr>
<td>Scrum boards</td>
<td>42</td>
<td>6</td>
<td>5.57</td>
<td>6</td>
<td>1.43</td>
<td>2</td>
</tr>
<tr>
<td>Daily scrums</td>
<td>45</td>
<td>3</td>
<td>5.60</td>
<td>6</td>
<td>1.57</td>
<td>1</td>
</tr>
<tr>
<td>Sprint planning meetings</td>
<td>43</td>
<td>5</td>
<td>5.16</td>
<td>6</td>
<td>1.77</td>
<td>1</td>
</tr>
<tr>
<td>Retrospective meetings</td>
<td>45</td>
<td>3</td>
<td>5.53</td>
<td>6</td>
<td>1.60</td>
<td>1</td>
</tr>
<tr>
<td>Demos</td>
<td>47</td>
<td>1</td>
<td>5.19</td>
<td>5</td>
<td>1.70</td>
<td>1</td>
</tr>
<tr>
<td>Working in sprints</td>
<td>46</td>
<td>2</td>
<td>5.74</td>
<td>6</td>
<td>1.22</td>
<td>2</td>
</tr>
<tr>
<td>Self sign up for tasks</td>
<td>42</td>
<td>6</td>
<td>5.48</td>
<td>6</td>
<td>1.53</td>
<td>1</td>
</tr>
<tr>
<td>Open office space</td>
<td>48</td>
<td>0</td>
<td>5.33</td>
<td>6</td>
<td>1.68</td>
<td>1</td>
</tr>
<tr>
<td>Cross-functional teams</td>
<td>46</td>
<td>2</td>
<td>5.41</td>
<td>6</td>
<td>1.48</td>
<td>2</td>
</tr>
<tr>
<td>Refactoring</td>
<td>37</td>
<td>11</td>
<td>5.59</td>
<td>6</td>
<td>1.04</td>
<td>3</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>39</td>
<td>9</td>
<td>5.62</td>
<td>6</td>
<td>1.48</td>
<td>1</td>
</tr>
</tbody>
</table>

**10. How satisifed are you overall with the ways of working in Project B?**

<table>
<thead>
<tr>
<th>Satisfaction</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.51</td>
<td>47</td>
<td>1</td>
<td>7.51</td>
<td>8</td>
<td>1.79</td>
<td>3</td>
</tr>
</tbody>
</table>

**11. How would you rate ...**
- your own productivity in Project B? 47
- the productivity of your team in Project B? 48
- the overall productivity in Project B? 43
- the quality of the software produced in Project B? 41

**12. If you compare to finding the solution yourself, how often do you ...**
- get help from team members to solve problems related to your project work? 45

<table>
<thead>
<tr>
<th>Help</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.98</td>
<td>3</td>
<td>5</td>
<td>1.27</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Number of respondents</td>
<td>Mean</td>
<td>Median</td>
<td>Standard Deviation</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>------</td>
<td>--------</td>
<td>-------------------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>Number of respondents</td>
<td>Valid: 133, Missing: 7</td>
<td>44</td>
<td>4</td>
<td>4.09</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Mean</td>
<td>4.09</td>
<td>4</td>
<td>1.49</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>5.53</td>
<td>6</td>
<td>1.47</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>2.44</td>
<td>2</td>
<td>1.62</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>1</td>
<td>1</td>
<td>1.68</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>7</td>
<td>7</td>
<td>1.47</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

13. To what extent do you feel that the members of your team have the necessary capabilities to help you with your project work?

14. Considering the last 7 days, which of the following have you performed?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement specification</td>
<td>13</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System design</td>
<td>25</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>26</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Writing tests</td>
<td>19</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Running tests</td>
<td>19</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refactoring</td>
<td>15</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of performed activities</td>
<td>48</td>
<td>0</td>
<td>2.44</td>
<td>1.62</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

15. To what extent do you agree with the following propositions?

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>The daily scrums help my team to solve problems together</td>
<td>41</td>
<td>5.15</td>
<td>6</td>
<td>1.73</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Sitting together in an open office space helps my team to solve problems together</td>
<td>45</td>
<td>5.78</td>
<td>7</td>
<td>1.70</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>The sprint planning meetings give me insights about the project I wouldn't get otherwise</td>
<td>43</td>
<td>5.40</td>
<td>6</td>
<td>1.40</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>The retrospective meetings give me insights about the project I wouldn't get otherwise</td>
<td>44</td>
<td>5.22</td>
<td>6</td>
<td>1.64</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>The demos give me insights about the project I wouldn't get otherwise</td>
<td>45</td>
<td>5.46</td>
<td>6</td>
<td>1.68</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

16. What is your opinion about the amount of the following software documentation items in Project B?

<table>
<thead>
<tr>
<th>Documentation Items</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement specifications</td>
<td>38</td>
<td>10</td>
<td>3.37</td>
<td>1.30</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Work packages (user stories, work items, etc.)</td>
<td>37</td>
<td>11</td>
<td>3.86</td>
<td>0.89</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Design specifications (implementation plans, one-pagers, etc.)</td>
<td>Number of respondents</td>
<td>Mean</td>
<td>Median</td>
<td>Standard Deviation</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>-----------------------</td>
<td>------</td>
<td>--------</td>
<td>--------------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td>Valid</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design specifications (implementation plans, one-pagers, etc.)</td>
<td>38</td>
<td>10</td>
<td>3.66</td>
<td>4</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>Code comments</td>
<td>35</td>
<td>13</td>
<td>4.00</td>
<td>4</td>
<td>1.03</td>
<td>2</td>
</tr>
<tr>
<td>Technical documentation of code (in e.g Confluence)</td>
<td>37</td>
<td>11</td>
<td>4.22</td>
<td>4</td>
<td>0.89</td>
<td>2</td>
</tr>
<tr>
<td>Process documentation (policies, project documents etc.)</td>
<td>39</td>
<td>9</td>
<td>3.79</td>
<td>4</td>
<td>1.08</td>
<td>1</td>
</tr>
<tr>
<td>Trouble reports</td>
<td>25</td>
<td>23</td>
<td>4.00</td>
<td>4</td>
<td>0.65</td>
<td>3</td>
</tr>
<tr>
<td>Change requests</td>
<td>24</td>
<td>24</td>
<td>3.88</td>
<td>4</td>
<td>0.90</td>
<td>1</td>
</tr>
</tbody>
</table>

17. What is your opinion about the overall amount of internal software documentation in Project B?

18. How important do you consider internal software documentation for Project B?

19. Based on your experiences from Project B, to what extent do you feel that internal software documentation can be replaced by face-to-face communication?

20. Considering the last 30 days, how much time have you on average spent each day on producing internal software documentation?

21. To what extent do you consider yourself aware of...

22. How important do you consider the following software development practices for your awareness of the project's status?
<table>
<thead>
<tr>
<th>Continuous integration</th>
<th>Number of respondents</th>
<th>Mean</th>
<th>Median</th>
<th>Standard Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>8</td>
<td>6.13</td>
<td>7</td>
<td>1.31</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>23. To what extent do you feel that you are pressured and stressed in a negative way...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... by the fact that your team should deliver results at the end of each sprint?</td>
<td>46</td>
<td>2</td>
<td>3.72</td>
<td>4</td>
<td>1.81</td>
<td>1</td>
</tr>
<tr>
<td>... to finish what you signed up for during a sprint?</td>
<td>44</td>
<td>4</td>
<td>3.89</td>
<td>4</td>
<td>1.79</td>
<td>1</td>
</tr>
<tr>
<td>... by the demos?</td>
<td>45</td>
<td>3</td>
<td>3.93</td>
<td>4</td>
<td>1.84</td>
<td>1</td>
</tr>
<tr>
<td>... by your work in Project B in general?</td>
<td>46</td>
<td>2</td>
<td>3.93</td>
<td>4</td>
<td>1.72</td>
<td>1</td>
</tr>
<tr>
<td>24. Do you feel pressured in a negative way to report results of progress on every day scrum?</td>
<td>44</td>
<td>4</td>
<td>3.32</td>
<td>4</td>
<td>1.99</td>
<td>1</td>
</tr>
<tr>
<td>25. Do you feel that sitting together with your team in an open office space disturbs your work?</td>
<td>45</td>
<td>3</td>
<td>3.04</td>
<td>2</td>
<td>2.12</td>
<td>1</td>
</tr>
<tr>
<td>26. Do you feel that you ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... understand the overall goal of Project B?</td>
<td>46</td>
<td>2</td>
<td>6.20</td>
<td>6</td>
<td>1.11</td>
<td>1</td>
</tr>
<tr>
<td>... understand how the task you are currently working with fits with the overall project goal?</td>
<td>46</td>
<td>2</td>
<td>6.33</td>
<td>7</td>
<td>1.09</td>
<td>1</td>
</tr>
<tr>
<td>... know what task you should be working on right now?</td>
<td>46</td>
<td>2</td>
<td>6.26</td>
<td>7</td>
<td>0.95</td>
<td>4</td>
</tr>
<tr>
<td>... know when that task is required to be finished?</td>
<td>46</td>
<td>2</td>
<td>5.98</td>
<td>7</td>
<td>1.44</td>
<td>2</td>
</tr>
<tr>
<td>... know what your team members are doing?</td>
<td>46</td>
<td>2</td>
<td>5.57</td>
<td>6</td>
<td>1.44</td>
<td>2</td>
</tr>
<tr>
<td>... know what the other teams are doing?</td>
<td>46</td>
<td>2</td>
<td>4.24</td>
<td>7</td>
<td>1.56</td>
<td>2</td>
</tr>
<tr>
<td>... know what capabilities your team members have?</td>
<td>46</td>
<td>2</td>
<td>5.54</td>
<td>6</td>
<td>1.31</td>
<td>2</td>
</tr>
<tr>
<td>... know what capabilities your project members have?</td>
<td>44</td>
<td>4</td>
<td>4.86</td>
<td>5</td>
<td>1.61</td>
<td>1</td>
</tr>
<tr>
<td>... can coordinate effectively within Project B?</td>
<td>44</td>
<td>4</td>
<td>5.14</td>
<td>6</td>
<td>1.71</td>
<td>1</td>
</tr>
<tr>
<td>27. Do you feel that the work of others you depend on is ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... available at the right time?</td>
<td>43</td>
<td>5</td>
<td>4.70</td>
<td>5</td>
<td>1.49</td>
<td>1</td>
</tr>
<tr>
<td>... is available at the correct location?</td>
<td>41</td>
<td>7</td>
<td>5.02</td>
<td>5</td>
<td>1.44</td>
<td>1</td>
</tr>
<tr>
<td>... the way you need it to be?</td>
<td>41</td>
<td>7</td>
<td>4.90</td>
<td>5</td>
<td>1.39</td>
<td>1</td>
</tr>
<tr>
<td>Number of respondents</td>
<td>Mean</td>
<td>Median</td>
<td>Standard Deviation</td>
<td>Min</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
<td>--------</td>
<td>-------------------</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scrum boards</td>
<td>41</td>
<td>7</td>
<td>5.17</td>
<td>6</td>
<td>1.76</td>
<td>1</td>
</tr>
<tr>
<td>Daily scrums</td>
<td>44</td>
<td>4</td>
<td>5.64</td>
<td>6</td>
<td>1.42</td>
<td>1</td>
</tr>
<tr>
<td>28. To what extent do you feel that the following software development practices help you know what your team members are doing?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29. To what extent do you feel that the other teams' scrum boards help you know what they are doing?</td>
<td>40</td>
<td>8</td>
<td>4.18</td>
<td>5</td>
<td>2086.00</td>
<td>1</td>
</tr>
<tr>
<td>30. Do you want to add something about the ways of working in Project B?</td>
<td>16</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level of implicit coordination effectiveness in Project B*</td>
<td>46</td>
<td>2</td>
<td>5.53</td>
<td>6</td>
<td>0.92</td>
<td>4</td>
</tr>
<tr>
<td>Level of explicit coordination effectiveness in Project B*</td>
<td>43</td>
<td>5</td>
<td>4.72</td>
<td>5</td>
<td>1.49</td>
<td>1</td>
</tr>
<tr>
<td>Level of coordination effectiveness in Project B*</td>
<td>46</td>
<td>2</td>
<td>5.25</td>
<td>6</td>
<td>1.03</td>
<td>3</td>
</tr>
</tbody>
</table>

*Derived variable