Software Effort Estimation as Collaborative Planning Activity

Kristin Børte

Thesis submitted in partial fulfilment of the requirements for the degree of
Philosophiae Doctor

Department of Educational Research
Faculty of Education
University of Oslo
April 2011

Series of dissertations submitted to the
Faculty of Educational Sciences, University of Oslo
No. 144

ISSN 1501-8962

Cover: Inger Sandved Anfinsen.
Printed in Norway: AIT Oslo AS, 2011.

Produced in co-operation with Unipub AS.
The thesis is produced by Unipub AS merely in connection with the
thesis defence. Kindly direct all inquiries regarding the thesis to the copyright
holder or the unit which grants the doctorate.

Unipub AS is owned by
The University Foundation for Student Life (SiO)

Acknowledgements

Pursuing a PhD is a learning process that can be compared to taking a ride on a roller coaster. It can be slow, fun, exciting, demanding, and sometimes even scary. Luckily, like the roller coaster, it has a solid support structure to secure a safe completion of the ride, or in this case the writing of a thesis. I would therefore like to thank my main supervisor Professor Monika Nerland for your dedicated readings of numerous drafts, insightful and critical comments on my work and the interesting discussions that we have had during my PhD. I am grateful that you included me in the LiKE project at the University of Oslo. Being part of this research project has been very valuable to me as my work is interdisciplinary. I would also like to thank my two co-supervisors Professor Sten Ludvigsen and Professor Magne Jørgensen. Your expertise and experience within analysing talk and interactions and software effort estimation, respectively have been invaluable. Also your insightful and critical readings of my work have been much appreciated. Working together with the three of you has made it possible for me to finish this thesis. The philosopher Mikhail Bakhtin once wrote: "the word of language is half someone else's…" and I believe that you are the ones that should be credited in that respect.

I would also like to thank the managing director at Simula Research Laboratory, Aslak Tveito, and the former director of Simula School of Research and Innovation, Kristin Vinje, for believing in me and giving me the opportunity to pursue a PhD.

Thanks to my colleagues at Simula Research Laboratory and in particular the members of my research group. You have provided a productive, fun and inspiring working environment. I would also like to thank the IT Operations for the excellent help you provided when technical problems occurred. Also thanks to the members of the LiKE project at University of Oslo, your comments and discussions of various parts of my work have been much appreciated.

I would also like to thank Åsa Mäkitalo and Roger Säljö for your warm welcome in relation to my research stay at the University of Gothenburg. The two months that I spent working with your research group were of great value for me in terms of understanding the theoretical perspective that I have employed as well as improving my skills as an analyst of interactions.

When writing a PhD-thesis there are times when the writing and ideas seam brilliant while the next day they are just not working. A heartfelt thanks goes to my family and friends for following me throughout my PhD and most of all for still hanging around afterwards. I have truly appreciated our talks and your encouragement over the years.

April 2011

Kristin Børte

PART I: Extended Abstract

Table of Contents

PART II: The Articles

ARTICLE I: Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation work in a multi-specialist team. *Scandinavian Journal of Information Systems*, 22(2), p. 65–98

ARTICLE II: Børte, K. (submitted). Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation.

ARTICLE III: Børte, K., Ludvigsen, S., & Mørch, A. (submitted). The role of concepts in expert work: Unpacking 'the magic step' in software effort estimation.

ATTACHMENT: Co-author declaration

Part I

Extended Abstract

1 Introduction

In today's society, we have come to depend heavily on the use of computers and computer systems. From the moment we wake up, turn off the alarm, prepare our breakfast, commute to work, answer e-mails, shop groceries on our way home until we order a summer vacation on the Internet, we have interacted with a number of different software systems. We are even encouraged to use software systems when we travel by public transport or hand in yearly our tax forms. Not all of the software systems we use in our daily life are equally visible, but they are just as important. For example, computers, and their requisite software, keep our food cool in the fridge and make it possible for us to drive our car. It is not only we as individuals who depend on software systems. The society as such depends heavily on them. For instance, software systems control airplanes and air traffic, as well as monitor and control oil platforms and public subway systems. The paradox is that even though society depends on software systems for safety and utility, the information technology (IT) industry has limited control in terms of predicting the amounts of resources it takes to develop them. This lack of control increases costs and stresses the development process, thus jeopardising the safety and utility that the software system was meant to accommodate. However, total control is not possible to obtain. This is because software development is concerned not merely with technical aspects; it is also a social practice in which software professionals conduct the work. This opens up for a need to understand social aspects in software development. In research on computer-supported cooperative work, this has been an area of focus (Dittrich, Randall, & Singer, 2009). However, there are important aspects of software development, such as software effort estimation, in which the social aspects have yet to be attended to.

In both newspapers and scientific articles we can read about "software project failure". These failures often refer to large budget overruns experienced by software projects. In Norway, one of the recent software projects receiving this kind of publicity in the media was the new electronic ticket system for public transport in

Oslo. This system has exceeded the planned costs by millions of NOK and the schedule for release of the system by years (Krogstad, 2010). One of the reasons for large overruns in the software industry is believed to be inaccurate estimates of the work effort it takes to develop software systems. The estimates of work effort that are provided are often too optimistic when used as input for budgeting the costs of developing a software system, which results in the planned costs being too low compared to what is actually needed. Hence, large budget overruns are often a consequence. Moreover, inaccurate estimates can also have other severe consequences. Companies can lose contracts and new business opportunities because of financial trouble. People can get fired from their jobs because of projects overruns and the quality of the software can be compromised to meet the demand for delivery on time, which again might result in unexpectedly high costs for maintaining the system. Hence, it becomes clear that software development effort estimation has significant consequences for individuals, organisations and society at large (Grimstad, 2006). Understanding software effort estimation as a work practice will therefore provide important knowledge for project management. This thesis seeks to contribute in this respect by focusing on software effort estimation as collaborative work.

In software development, effort estimates are used among other things for purposes such as budgeting, planning and control (Boehm 2000). This implies that software effort estimation rests on the assumption that it is possible to plan for the future and that, if the plan is good and the actions are in accordance with the plan, software projects will not experience the excessively large overruns, as is often the case today. This assumption is in line with the planning model in cognitive science where a plan is treated as a series of actions that have been outlined to achieve a particular goal (Suchman, 2007). Plans are thereby perceived as determining action. Studies have shown that this is a somewhat problematic assumption, because gaining control of complex problem-solving processes such as planning is very difficult. Planning and also control are not phenomena that can be reduced to processes that are 100% controllable. Rather, Suchman (2007) has suggested that plans should be treated as resources that people can use when talking and discussing about future possible actions.

The problems of planning and the societal consequences of inaccurate software effort estimates have invoked research interest within the research field of software engineering, because finding a solution to these problems would "improve the use of scarce financial and human software development resources" (Jørgensen & Grimstad, 2009, p. 383). Researchers have addressed issues concerning inaccurate effort estimation of software development projects since the 1960s (Jørgensen & Grimstad, 2009)[1]. Most of that research has been concerned with the use and development of algorithmic and formal estimation models (Jørgensen & Shepperd, 2007). However, the most frequently used estimation method in the industry is called judgement-based estimation (Bratthall, Arisholm, & Jørgensen, 2001; Heemstra & Kusters, 1991; Hihn & Habib-Agahi, 1991; Jørgensen, 2004a). The term judgement-based means that the quantification step, i.e., "the step where an understanding of the software development estimation problem is translated into a quantitative measure of the required effort" (Jørgensen, 2007, p. 450), is based on a judgemental process rather than a deliberate mechanical calculation (Halkjelsvik & Jørgensen, submitted).

In the software industry, there is an increasing trend towards organising estimation work in teams of software professionals (Haugen, 2007). Moreover, studies of group estimation have also shown promising results regarding estimation accuracy when estimating in teams (Moløkken-Østvold & Jørgensen, 2004; Moløkken-Østvold, Haugen, & Benestad, 2008; Passing & Shepperd, 2003). Knowing that estimation work is often carried out as teamwork in the industry, with software professionals from different specialist areas in software development participating, there is a need to investigate teamwork processes. The research reported in this thesis examines the collaborative work in software effort estimation by analysing in depth the use of three different estimation approaches.

Most studies of judgement-based estimation have had as the aim to improve the accuracy of effort estimates in the industry. This is evident in the previous research on both teamwork in judgement-based estimation and in the studies of individual expert work in judgement-based estimation. After an inspection of the conducted studies of teamwork in judgement-based estimation it seems that the amount of

[1] See http://simula.no/BESTweb for an overview of all journal papers that have been published on software cost and effort estimation.

research that has been conducted is small in this area. Moreover, most of the studies of teams are based on methodological individualism in which the individual expert is taken as the unit of analysis. On the other hand, the research investigating individual expert work in judgement-based estimation is quite extensive. Several research methods have been used in this work, but the main method employed has been that of small-scale experimental studies. Through a series of experiments, researchers have revealed a number of different aspects that influence the decisions software professionals make on an estimate, such as misleading or irrelevant information, anchor information and wishful thinking (Halkjelsvik & Jørgensen, submitted). In addition, research has also revealed important findings regarding the difficulties software professionals have explaining the details as to how they arrive at an estimate by analysing interviews, think-aloud protocols and video recordings (Bratthall et al., 2001; Jørgensen, 2004a).

Previous research on judgement-based estimation has provided important contributions and useful insight regarding the human judgement and decision making that is involved in estimation work. Thus, the focus has therefore been on understanding the cognitive steps when deciding on an estimate. The research presented in this thesis focuses on collaborative work and not individuals. Studying teams allows for a focus on collaborative work in software effort estimation; however, the unit of analysis needs to be changed from the individual expert to the social interaction. To do this, a theoretical perspective that account for the social is needed.

1.1 Taking a sociocultural perspective on software effort estimation

The thesis employs a sociocultural perspective on software effort estimation work because, when studying collaborative activities, an understanding of human activity at different levels is needed (Valsiner & Van der Veer, 2000; Wells, 1999). The sociocultural perspective rests on a set of basic premises that provides such an understanding. It rests on an ontological assumption that humans are social in nature and that human personality is developed through social experiences, which takes place at the micro-genetic level, i.e., social interaction, through moment-to-moment interaction by engaging in different types of activities and action (Billett, 2003; De Graaf & Maier, 1994; Valsiner & Van der Veer, 2000). The epistemological assumptions rest on the idea of co-construction of knowledge, in which participants in a practice construct knowledge together as part of an intellectual interdependency (Valsiner, 1994; Valsiner & Van der Veer, 2000). This means that a sociocultural perspective sees humans as inherently social and that we interact with each other and with objects in our environment as ways of solving problems, sharing experiences, and learning and developing our knowledge.

There are some core premises that follow the sociocultural perspective. The first premise I will emphasise is the situated character of action. This implies that all human action takes place in a cultural, institutional and historical context. This means that human action is institutionally, historically and culturally situated. Human action is thereby shaped by the different situational factors that are available in a context. Moreover, the ways we solve problems are related to the context and the different tools that are available (Suchman, 2007). The second premise I will emphasise is that all human action is mediated (Wertsch, 1991). This implies that individuals use different types of tools, both physical and intellectual, as part of social practice. Artefacts and cultural tools have been developed to assist human cognition and thinking. These artefacts and tools inhabit human knowledge and experience, which has developed over time, thereby providing us with knowledge and information, and, in effect, expanding our predisposed biological and intellectual capabilities. The third core premise I would like to draw attention to is that knowledge and understanding is achieved in social interaction. It is in social interaction that knowledge is shared and constructed and where meaning making of objects, events and actions occurs. Hence, it is in social interaction that people use language and concepts, align their expertise and activate different types of resources that constitute what they achieve in interaction. The sociocultural perspective emphasise that we interact with, not only other people, but also cultural tools and objects in our environment and that this is how knowledge comes into use (Greeno, Collins, & Resnick, 1996).

When organised as teamwork, software effort estimation can be understood as a collaborative activity in which software professionals work together to achieve a predefined goal. It is also an institutional practice, which has developed over time as part of the increasing demand amongst individuals and societies for more and better software systems. Further, the work is mostly conducted by software professionals with specialist knowledge from the domain of software development. To grasp these dimensions, a focus on social interaction is needed. Taking a sociocultural perspective on this type of work opens up possibilities for investigating estimation work at the level of social interaction. This makes it possible to understand the collaborative activity of software effort estimation.

When following a sociocultural perspective, it is through social interaction that the task of estimation gets solved, thus it is an interactional accomplishment among the participants in the practice. A set of theoretical concepts is therefore needed to explicate and understand these interactional achievements. In this thesis, I use the concepts of distributed expertise, meaning making, meaning potential, recontextualisation, collective remembering and tool-mediated actions to grasp these interactional achievements that occur through the team's collaborative work. This makes it possible to attend to both the problem-solving aspect of estimation and the planning aspect, as estimation work revolves around reasoning about a future event.

As an activity, planning can be described as a practice that is both discursive and imaginative. This implies that it is through planning activities that reflections upon what might occur in the future take place. In this thesis planning activities are regarded as situated actions and as resources of action rather than controlling structures (Suchman, 1987, 2003, 2007). Moreover, planning is dependent on the moment-to-moment interactions that occur between different participants and between the participants and the context in which the planning activity is conducted (Suchman, 2007). Planning will therefore, in some sense, be vague because it is not possible to specify the social interaction up front. In relation to estimation work, it is thereby important to attend to both the actions that are projected in the future and also their dependability in the estimation activity at large (Suchman, 2003).

1.2 Aims and research questions

The research presented in this thesis seeks to 1) explicate the teamwork processes of software effort estimation and 2) to understand the estimation practice as more than merely a decision about numbers of work hours, but as a collaborative planning activity carried out through social interaction. Three different estimation approaches are investigated in depth, based on empirical data material gathered from two separate cases: one quasi-experimental estimation study in the software industry and one naturalistic case study of an ongoing large software development project. The thesis employs interaction analysis (Derry et al., 2010; Jordan & Henderson, 1995) as a means to investigate video recording of software professionals' collaborative work when estimating software development projects. In the analysis, both content and what the participants achieve in the interaction are attended to by using a set of analytical concepts as sensitising means.

The aim of this thesis is to make empirical, theoretical, and methodological contributions to the research on software effort estimation. The empirical contribution is twofold. Firstly, it contributes by providing an understanding of software effort estimation as collaborative planning activity. Secondly, it contributes to an explication of the complexity of software effort estimation work by proposing three process models that demonstrate the following: 1) the interactional accomplishments of software effort estimation; 2) how historical information becomes recontextualised to be useful in new contexts; and 3) the mediating role of cultural tools in software effort estimation. Thus, the models have an analytical purpose.

The methodological contribution in this thesis is related to the introduction of a new research method in the field of software effort estimation. By introducing interaction analysis as a way of investigating video recordings of the collaborative work in software effort estimation, it is possible to provide an understanding of the collaborative processes that are involved. With an analytical focus on the social interaction, the interactional achievements can thereby be explicated and thus an understanding of what goes on in the estimation practice can be provided. Prior to the work within this thesis, this approach to study judgement-based effort estimation as a collaborative and interactional achievement has not been employed in estimation research.

The main research question that the research presented in this thesis seeks to answer is:

– *How can we understand software effort estimation as collaborative planning activity?*

To answer this main research question three sub-research questions have been formed that have been addressed throughout three separate articles investigating three different approaches to estimation work. The following sub-questions have been formed:

i) What characterises software effort estimation as interactional accomplishments?

ii) What kinds of challenges do estimation teams face when utilising historical information in present working tasks?

iii) *In what ways are estimation work mediated by cultural tools?*

While the first sub-question runs through the three articles, it is especially dealt with in Article I, in which the kind of work conducted by software professionals when employing a bottom-up estimation approach is investigated. The next two sub-research questions are especially addressed in Articles II and III. Article II focuses on how teams of software professionals go about using historical information in present estimation tasks, while employing a top-down estimation approach. Article III focuses on how estimation work is mediated by different types of concepts and how the software professionals invoke different types of knowledge when reasoning about and reaching a decision on an effort estimate by investigating the planning poker estimation approach.

1.3 Outline of the thesis

The thesis consists of two parts. The aim of Part I, the extended abstract, is to account for the entirety of the research contribution. Part II consists of the three articles that have been written as part of this project. This means that the thesis is article based, and thus comprises three separate articles. Even though each represents a separate research contribution the three articles are related in the sense that the focus is on explicating the collaborative work of estimating a software development project in teams of software professionals. Moreover, the articles follow one another by describing different parts of estimation work in depth. Together, the three articles provide an understanding of estimation as collaborative planning activity taking place in teams of software professionals as interactional achievements.

The extended abstract in Part I is outlined as follows: in Chapter 2 following this introduction, a description is given of the phenomenon of software effort estimation and some of the different approaches to estimation work. Thereafter, a review of relevant research on judgement-based software effort estimation is presented. The focus in this review is on previous research where estimating in teams has been investigated. In addition, to provide an understanding of why software effort estimation is a difficult problem-solving task, a short overview of some of the relevant findings from studies of individual expert work in judgement-based effort estimation that are regarded as important is presented. The main purpose of this chapter is to provide an understanding of the phenomenon studied in this thesis as well as give a background for arguing the need of researching the collaborative work of software effort estimation.

In Chapter 3 the theoretical perspective employed in this thesis is presented. Here, the sociocultural perspective will be described and important theoretical concepts relevant for the study of estimation as collaborative planning activity will be explicated. The main aim of this chapter is to provide a conceptual framework for understanding estimation work as a social, collaborative planning activity that rests on interactional achievements to be completed. Based on this background, an analytical approach for studying the collaborative work in software effort estimation will be introduced.

In Chapter 4, the data material from the two cases that have been investigated and subject to analysis will be presented. This is followed by a description of interaction analysis as the method that has been used as the means to investigate the data. The analytical procedure of employing interaction analysis together with the intermediate concepts that are used as sensitising means in the analysis are then described for the three different articles that have been written as part of this thesis. Chapter 5 presents a summary of the three different articles that constitute the main contributions in this thesis, before the contributions of this work are discussed and concluded in Chapter 6.

Part II of this thesis consists of the three articles that have been published or submitted for publication, presented in the order in which they were written.

Article I:	Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation work in a multi-specialist team. *Scandinavian Journal of Information Systems*, 22(2), p. 65–98
Article II:	Børte, K. (submitted). Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation.
Article III:	Børte, K., Ludvigsen, S., & Mørch, A. (submitted). The role of concepts in expert work: Unpacking 'the magic step' in software effort estimation.

2 The phenomenon studied: Software effort estimation

In this chapter, I will first provide a description of the phenomenon of software effort estimation and discuss the importance of attending to this in research. As part of this description, some of the different approaches and techniques that can be employed when conducting this type of work will be outlined. Thereafter, a review of related research within judgement-based effort estimation will be presented and discussed. As will be shown, a large part of this research has focused on the individual expert as the unit of analysis and only a few studies have looked at teams. I will therefore include a few but important contributions that have focused on the individual expert, in addition to studies that have focused on teams. The aim of this review is to provide an understanding of the research field on judgement-based software effort estimation and argue the need for a focus on the social aspect in software effort estimation work. To obtain this I need to include research contributions from both strands of research.

2.1 What is software effort estimation and why is its study important?

In the introduction, I stated that software systems permeate our daily lives and that society as such depends on software systems for safety and utility. For instance, for safe and efficient operation, airports, oil platforms and collective transportation all rely on software systems. This means that the consequences of failure in one of these systems can be severe. In 2008, the Norwegian information and communications technology (ICT) sector as a whole had annual revenues of 241 billion NOK (SSB, 2008). A large part of this amount is connected to software development. Further, at the time of writing, there are several huge software development projects in progress in Norway. FLEXUS, which is a project for developing an electronic ticket system for public transportation in Oslo, has, over the last 20 years, cost more than 600 MNOK (Krogstad, 2010). The Norwegian Public Service Pension Fund (SPK) has started a project called PERFORM, the aim of which is to adapt to the new pension regulations and to replace existing technologies with a new solution. This project has a cost frame of 888 MNOK (St.prp. nr. 1 [2008-2009]). The Norwegian Labour and Welfare Service (NAV) is going to develop a new software system, which is the largest public software project in Norway. At the time of writing, 1200 MNOK have been budgeted for the first three years and it has a cost frame of 1000 MNOK[2] for the ICT parts. These examples show that the amount of money that is spent or planned to be spent on software development is large. Also, large cost overruns have been reported in relation to software projects. For example, the Tress-90 project, which is known as the largest IT-project failure in Norway, was cancelled after three years due to substantial delay and extreme cost overruns. A project revision showed that while the initial estimate of the project was 383 MNOK, the cost that had been spent after three years was 463,1 MNOK and to complete the project, it was estimated that an additional 430 MNOK was needed[3]. In more recent times, the above-mentioned FLEXUS project has received publicity due to delays and cost overruns. This project has exceeded the planned costs by 110 MNOK and the schedule for release by more than four years (Krogstad, 2010). These two examples illustrate the limited control the IT-industry has when it comes to predicting the planned costs for developing software systems, as the reported budget overruns are large.

Developing a software system is a complex task, which comprises different software processes such as specification, design, implementation, validation and evolution. It is therefore challenging to create predictable software processes so that the software project can meet the delivery on time, on budget and in a cost-effective manner (Sommerville, 2007). As software development is dealing with future-oriented aspects of a system that is yet to be developed, planning is essential to control and monitor these software processes. In this respect, the estimation of work effort is a core task, because it is used for purposes such as budgeting, trade-off and risk analysis, project planning, control and software improvement investment analysis (Boehm, Abts, & Chulani, 2000). In software development projects, the most important cost driver is work effort. The reason for this is because, while the material

[2] http://www.steria.no/id/11003784.0 retrieved 11.01.2011
[3] http://www.stortinget.no/no/Saker-og-publikasjoner/Publikasjoner/Innstillinger/Stortinget/1994-1995/inns-199495-210/7/ retrieved 14.04.2011

costs of software development are low, the number of work hours software professionals spend on developing a software system is large.

When carrying out the work of estimating a software development project, software professionals engage in a particular institutionalised practice with certain rules and constraints. In the industry, project managers seldom carry out estimation work alone. Instead, they depend on input from specialists in different areas of software development such as programmers, technical architects and database specialists in order to provide an effort estimate. This implies that there is a need for different kinds of specialist knowledge to conduct the estimation work. Further, it is an increasing trend in the industry to organise estimation work in teams of software professionals (Haugen, 2007). The estimation work is then organised in meetings where a particular estimation method to use has been agreed upon. When working in teams, the interactional and communicative processes become significant because of the collaborative dimension that comes into play.

An estimate can be defined as a "prediction of how long a project will take and how much it will cost" (McConnell, 2006, p. 3). In this thesis an effort estimate is understood as the most likely number of work hours necessary to complete a software development project as assessed by the managers and developers responsible for delivery. For more about the different interpretations of the term estimate, see Grimstad, Jørgensen & Moløkken-Østvold (2006).

Estimating the effort of a software development project is difficult and far from a straightforward work process. The reason for this is twofold. First of all, estimation is a future-oriented task that is concerned with making predictions about a system that is yet to be developed. This future-oriented aspect can be perceived as the most challenging part of software estimation work since predictions are made on grounds of uncertainty. Moreover, software effort estimation rests on the assumption that it is possible to plan for the future and thereby control the environment to a certain extent. Estimation work can therefore be conceived as a planning activity due to this future-oriented aspect.

Secondly, a description of the kind of software system that is to be developed is needed for conducting the estimation work. In the work process, software professionals have documents called requirement specifications, which often are structured documents describing the technical details of the system and how the system should function (Sommerville, 2007). Usually the requirement specification constitutes the main source of information for the estimation work. In spite of this, no official standards exists for exactly what a requirement specification should include or how it should look; hence, they come in all sizes and shapes of varying quality. Due to this variation in quality, it constitutes a challenge for software professionals to interpret this document and understand what kind of system that is requested is being developed and thus estimated. Moreover, when the estimation work is conducted in teams, achieving a shared understanding of the requirement specification is crucial.

A typical estimation situation usually includes a customer and a software supplier and may play out as follows: What happens first with regards to estimation is that a customer has a request about how much the development of a software system will cost and how much time it will take. The software system has been described, to the best of the customer's ability, in a requirement specification. This requirement specification is then handed over to the software supplier so that they can provide an estimate of the costs of developing the system. Usually, several software professionals come together to discuss and make sense of the requirement specification. In this process, the software professionals need to come to an understanding of, for example, the technological aspects of the system and the need for databases, among other things, so as to understand what kind of system is being requested for development and thus needs to be estimated. This can be a time-consuming activity, as iterations with the customer might be needed to clear up ambiguities in the requirement specification. Further, making sense of the requirement specification is crucial for understanding the kind of system that is described. When such an understanding is achieved and discussed, an initial estimate of the costs to develop the requested system is provided and presented to the customer. In software development, effort estimates are provided for quite different purposes such as bidding and budgeting. Thus, an initial estimate is just the first time the software system is being estimated. This means that estimation work is not a one-time engagement. Often there is a need to go back and re-estimate as the software project moves and, for example, more accurate estimates are required for budgeting purposes. Therefore, the work of estimating the effort of a software development project is a complex task that engages several software professionals multiple times as well as at different times during the duration of the software development project.

2.2 Different approaches to conduct estimation work

There are a number of different estimation methods and approaches that exist to achieve an effort estimate of a software system. These range from algorithmic models to what are known as human judgement-based or expert judgement estimations. One classification of different ways of achieving an estimate can be found in Boehm (1984). The research presented in this thesis focuses on the method that is known as judgement-based estimation. More specifically, the focus is on the collaborative activity of achieving an effort estimate in teams of software professionals. Even though judgement-based estimation relies on judgemental processes as opposed to deliberate mechanical calculation (Halkjelsvik & Jørgensen, submitted), the distinction between formal estimation models and judgement-based estimation is not clear cut because formal estimation models also rely on human judgement as input to the model.

Moreover, judgement-based effort estimation is not one unified method to perform estimation work. It consists of judgement-based processes that can be conducted based on pure intuition or be more or less structured and supported by historical data. Over the years, numerous documents, processes and guidelines have been developed to support judgement-based estimation.

When using a judgement based estimation method, there are several ways of approaching the estimation work. In this thesis, I have chosen to explicate the two approaches known as the bottom-up approach and the top-down approach, because these are the most relevant ones for the in-depth studies that have been conducted in Articles I and II. In addition, techniques for estimating in teams are presented in which the planning poker technique is emphasised, as this is the technique that is studied in depth in Article III. The planning poker technique can also be perceived as a variant of the bottom-up approach, though specifically designed for teams.

2.2.1 Describing the bottom-up and the top-down approaches

When following a *bottom-up approach*, the project work is usually divided into different project activities or components before the effort of each activity or component is estimated. The total effort estimate of the system development is then the sum of each effort estimate of the different project activities, with a possible addition of a buffer to cover unexpected events (Heemstra, 1992; Sommerville, 2007). A work breakdown structure can be useful to assist in the process of breaking the project down into the different project activities (Tausworthe, 1980). When following a bottom-up approach, expert knowledge on how to develop a software system is necessary because the details of the system need to be investigated.

When following a *top-down approach*, on the other hand, the total effort of a software project is estimated without breaking the system down into different project parts or activities. Instead, the estimators look for similar previously completed software projects to compare the current project with, and adjust for differences before a total effort estimate is agreed upon. Thereafter, the total estimate is distributed over the different project activities (Heemstra, 1992). The top-down approach can be employed in various ways, depending on what is used as a point of departure for finding similarities between completed software projects. This could, for example, be an amount of resources such as number of people or previously completed projects that are similar in size or complexity. The common aspect among the different ways of employing a top-down approach is that the system level is what is taken as the point of departure and not the different components (Heemstra, 1992).

Arguments in favour of a top-down approach state that it is more efficient, i.e., that it is less time consuming and thereby cheaper to apply than a bottom-up approach (Boehm, 1984). Other researchers have stated that it is an approach that is possible to apply without much knowledge of how to build software (Jørgensen, 2004b; Moløkken-Østvold & Jørgensen, 2005). Also, that it may reduce the bias towards over-optimism in estimation, because it makes use of historical data from previous projects to a greater extent (Moløkken-Østvold & Jørgensen, 2005). However, Hughes point out that even if an independent group could produce a reasonable top-down estimate, "subsequent detailed planning would still require low level estimates for individual activities which need the participation of people with more detailed knowledge of the system to be implemented" (Hughes, 1996, p. 70).

Both of these approaches have their advantages and disadvantages. Researchers have suggested that these are complementary in the sense that what are recognised as disadvantages of the bottom-up approach are recognised as advantages of the top-down approach and the other way around (Boehm, 1984; Hughes, 1996; Sommerville, 2007).

2.2.2 Describing techniques for estimating in teams

The work of estimating a software project, whether a top-down or a bottom-up approach is employed, is usually not a one-man job. Estimating in teams, also called group estimation, can be done in various ways and a range of different techniques has been developed to structure this kind of work. In the following, some of these techniques will be described. The techniques that I have chosen to include here are judgement-based estimation methods and can be categorised as ways of combining individual estimates where teamwork is facilitated to a greater or lesser extent. The process of combining decisions from several individuals can be done in a number of different ways. It can be open or anonymous, structured or unstructured, mechanically/statistically or it can be done through discussions. What distinguishes these different ways of combining individual estimates is how and to what degree teamwork is facilitated in the process.

One technique is called the Delphi technique. This technique was originally developed by the RAND GROUP in the 1950s for the US Air Force, but has been applied over a range of fields such as the health care industry, information systems, education, transportation and engineering (Rowe & Wright, 1999). Rowe and Wright (1999) summarised the key characteristics that are essential for the Delphi technique as consisting of anonymity, iteration, controlled feedback and statistical aggregation of group response. The technique uses questionnaires as a means to express the different software professionals' meanings and judgements anonymously. Thereafter, iterations are performed where the different participants are given opportunities to change their opinions. Before these iterations are conducted, the participants receive controlled feedback about their anonymous team members' opinions. The feedback often takes the form of a statistical summary of the response of the whole group as mean or median values: however, information and arguments about values that deviate can also be provided. After a number of iterations regarding the answers on the questionnaires, a group decision is made by using the statistical average of the participants in the final round (Rowe & Wright, 1999). The Delphi technique as summarised above can be categorised as anonymous, where the combination of individual estimates is conducted by statistical methods. Therefore teamwork as such is not facilitated.

A modified version of the Delphi technique is called Wideband Delphi (Boehm, 1981). What differentiates this technique from the original Delphi technique is that Wideband Delphi facilitates team discussions and allows for the participants to meet and discuss issues of relevance both before and during the iterations of estimates. Boehm summarises the wideband Delphi technique in six different steps: “1. Coordinator presents each expert with a specification and an estimation form. 2. Coordinator calls a group meeting in which the experts discuss estimation issues with the coordinator and each other. 3. Experts fill out forms anonymously. 4. Coordinator prepares and distributes a summary of the estimation on an iteration form. 5. Coordinator calls a group meeting, specifically focusing on having the experts discuss points where their estimates varied widely. 6. Experts fill out forms, again anonymously, and steps 4–6 are iterated for as many rounds as appropriate” (Boehm, 1981, p. 335). The Wideband Delphi technique opens for team discussions concerning issues where the estimates vary to a large extent. However the quantification step is completed and combined anonymously.

Another technique that facilitates group discussion even more than the wideband Delphi is what is named unstructured group discussion. Here, the software professionals first provide an individual estimate before a team of software professionals is formed that discusses the different estimates and through these discussions agree on a total estimate for the task (Moløkken-Østvold & Jørgensen, 2004; Passing & Shepperd, 2003). This technique can be characterised as open and unstructured.

There are also ways of achieving an estimate in teams that are based on carrying out all of the estimation work in teams. One method that has been developed as judgement-based group estimation is what Taff and colleagues called “estimeetings”

(Taff, Borchering, & Hudgins, 1991). This approach was developed and reported on as a way of achieving an estimate in the early phases of large software projects. The estimeetings were organised as standardised working meetings, which experienced software professionals with different areas of specialist competence attended on a regular basis. How these differ from a regular meeting is the methodology that is used to ensure that the estimeetings are "productive, successful and produce a consensus estimate" (Taff et al., 1991, p. 843). The idea behind estimeetings is to achieve an accurate estimate by bringing specific elements together. The different elements mentioned are a group of estimators, a high-level requirement specification, the responsible engineers who have the authority to change the requirements in the meetings, the design proposal together with the engineers who created it and the engineers who are specialists in the area of the proposed feature that is going to be estimated. This method is quite complex and also demanding of resources, as it requires attendance in meetings over a period of several weeks.

A fairly new technique that proposes a semi-structured approach when estimating in teams is called *planning poker* (Grenning, 2002). This technique is investigated in depth in Article III in this thesis. In planning poker, the participants are developers. This includes programmers, testers, database engineers, user interaction designers, etc. The number of participants typically does not exceed ten people (Cohn, 2006). In planning poker, the team members are estimating what is called user stories. A user story is a form of requirement that is formulated as a one-sentence statement about what the user should do or wants to achieve in a particular situation. There is no mandatory way to formulate such user stories but it can follow the following format: "As a <type of user> I want <capability> so that <business value>" (Cohn, 2006, p. 26).

What is characteristic of planning poker is the special deck of cards that is used to achieve effort estimates. The different numbers on the cards are called story points, which are measures to express the overall size of a particular feature (Cohn, 2006) and thus they represent the amount of work that is needed on a specific task. Story points are relative, and in order to use them in a meaningful way, a baseline or a reference task will have been estimated, which the participants use as a reference for estimating new tasks in their estimation work. This baseline reference task can for example be chosen by finding the smallest user story the team will work on, and giving it one story point, or choosing what seems to be a medium user story, which is given a value that would be in the middle of the range that is expected to be used (Cohn, 2006). All the other user stories are then estimated by comparing it to the first that was estimated as reference. The story points have a relative value and a card with two story points represents twice the amount of work of a card that has one story point written on it. In the beginning of a planning poker session, each team member is given a deck of cards where each card has a number written on it that is used as the valid estimates. A deck of cards can for example include the numbers: 0, 1, 2, 3, 5, 8, 13, 20, 40: see Figure 1.

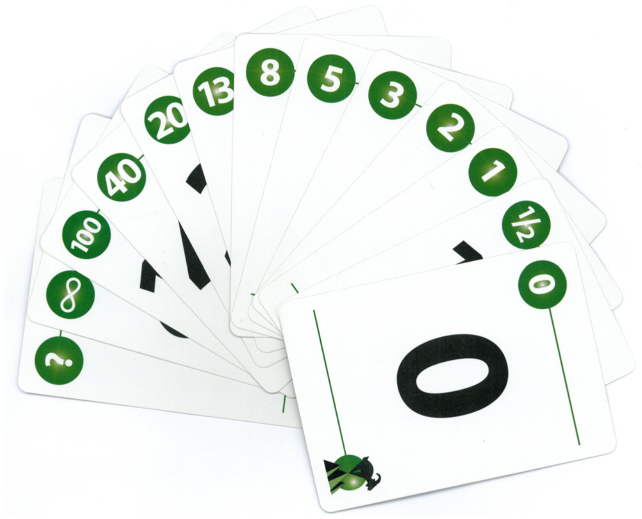
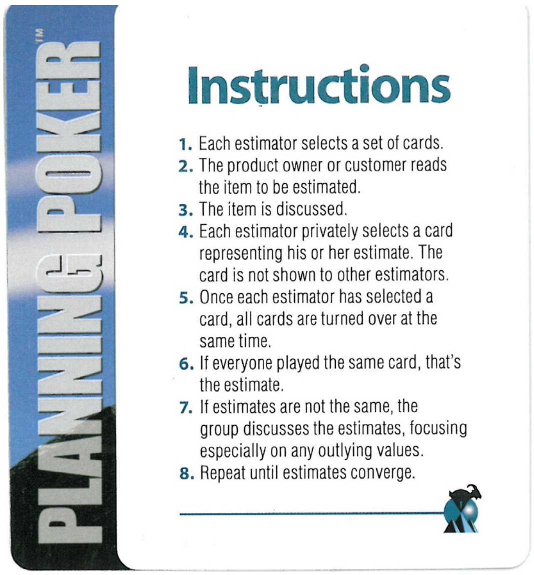


Figure 1 Illustration of a planning poker deck of cards.

A planning poker session often starts with a presentation of the user story that is requested for estimation. The user story is often explained by a customer or by an analyst from the estimation team. After the explanation, the team discusses the user story and the work that is needed to implement it. This discussion continues until the team feels that they share an understanding of the information that is needed to estimate the effort. Next, the team members individually estimate the user story by choosing a card from the card deck they have been given that represents their estimate. The cards from each team member are kept a secret until everyone has chosen a card. Then, by flipping the card around and placing it on the table, the numbers on the cards are shown to the rest of the group simultaneously. The team members that have provided the lowest and the highest estimate are then asked to justify their estimate to the group. After the justifications are provided, the team continues the discussion a few more minutes before a new round of planning poker is played, in which the team members re-estimate by selecting a new card. Again, this is kept private until all the team members have selected a card and it is time to flip the cards and show them to everyone simultaneously again. If the team did not reach a consensus estimate, then the process is repeated again until a consensus is achieved (Cohn, 2006; Grenning, 2002). Figure 2 shows the planning poker instructions, as they are included in the planning poker deck of cards.


PLANNING POKER™

Instructions

1. Each estimator selects a set of cards.
2. The product owner or customer reads the item to be estimated.
3. The item is discussed.
4. Each estimator privately selects a card representing his or her estimate. The card is not shown to other estimators.
5. Once each estimator has selected a card, all cards are turned over at the same time.
6. If everyone played the same card, that's the estimate.
7. If estimates are not the same, the group discusses the estimates, focusing especially on any outlying values.
8. Repeat until estimates converge.

Figure 2 Instructions for planning poker.

What is special about planning poker is the way in which it is organised that ensures all the team members' participation, regardless of position, role or experience. In addition, planning poker as a technique combines the opinions from several software professionals representing different areas of expertise in software development. Hence, this technique can be described as being open and facilitating team discussion to a large extent.

Above, I have described what kind of work software effort estimation is and argued for why this is an important work practice to investigate. In addition, I have outlined some of the different approaches to estimation work and some of the techniques that describe how estimation works can be organised as teamwork. This description provides a background for understanding the phenomenon of software effort estimation, which is investigated in this thesis. In the following, I present a review of the research that has been conducted within judgement-based effort estimation. The aim of this review is to provide an understanding of the research field of judgement-based software effort estimation. To do this, I need to include studies of teamwork in estimation as well as a few significant studies that focus on the software professional conducting estimation work as an individual expert.

2.3 Review of research on judgement-based software effort estimation

A review of studies of software development projects shows that 70% to 80% of such projects overrun their estimates and spend on average 30% to 40% more effort than estimated (Moløkken-Østvold & Jørgensen, 2003). This indicates that providing realistic effort estimates of software development projects is a challenge.

Judgement-based software effort estimation has existed as an estimation method for a long time and is the most applied estimation method in the industry (Bratthall et al., 2001; Heemstra & Kusters, 1991; Hihn & Habib-Agahi, 1991; Jørgensen, 2004a). However, this estimation method did not receive much attention in the research field before 1990. A systematic review of software cost studies published in journals showed that only three journal papers focusing on judgement-based effort estimation had been published before 1989. Further, 22 journal papers were published between 1990–99 on judgement-based estimation and as many as 21 journal papers were published in the period 2000–04 (Jørgensen & Shepperd, 2007). This systematic review has resulted in a database containing an overview of all the papers on software cost estimation. The database is called BEST and can be found here: http://home.simula.no/BESTweb/.

The aim of developing this database has been to provide researchers with an overview of research papers addressing the issue of software cost estimation. Hence, the selection of papers included in this review is based on searches in this database.

As the main selection criteria, I have used judgement-based estimation, categorised as expert estimation in the database. In addition, I have searched the database for studies on teamwork and group estimation. The search in the BEST database showed that the number of studies focusing on teamwork in judgement-based software effort estimation was small. Out of the 64 publications focusing on expert judgement estimation as method, only eight studies focused on teamwork or group estimation techniques.[4] In the following review, all eight studies are presented and discussed.

2.3.1 Studies of teamwork in judgement-based effort estimation

One of the first studies reporting on group estimation was conducted by Taff and colleagues (Taff et al., 1991). They both developed and reported on the previously described method called "estimeetings". Accordingly, their study can be characterised as exploratory. The method of "estimeetings" was used to estimate three different releases of a large software-intensive project. The results that were reported highlighted several benefits from employing the estimeeting technique. The accuracy of the total effort estimates provided for the different releases tracked well with the actual cost of the releases. Estimeetings used in practice facilitates collaborative work between people with different backgrounds and from different organisations. Thus, knowledge about project-wide issues is shared. Taff and colleagues highlight 12 different benefits when employing estimeetings. The first five mentioned benefits as follows: 1) better estimates; 2) earlier and closer subsystem involvement because owners of subsystems participate in the work meeting; 3) fostering of teamwork between different organisations and thus early direct relationships are facilitated; 4) early expert high-level designs for the next release is a positive by-product; and 5) problems are easier to detect because more people can deliberate on the different issues. Apart from this initial study where the method was developed, I found no other studies of how the approach is used.

A couple of studies have investigated the effect group discussions have on the accuracy of effort estimates. Passing and Shepperd (2003) reported on a study investigating how checklists and group discussion could help estimators in improving the accuracy of their estimates. They conducted a study with 13 software engineering students as participants. In this study, the participants were asked to

[4] Search performed 13.12.2010

estimate the size and the effort of a one-week software engineering project they were supposed to complete as a part of their course. The data material collected consisted of data from the forms the participants filled out and transcripts and protocols from the group discussions. The study consisted of three rounds of estimation. First, the participants estimated the project size and effort using their preferred estimation method. Next, the participants were presented with estimation guidelines and then given a chance to update their initial estimate. However, this time the form that they should register their estimate on included a checklist that listed important items, one for size and one for effort. Thirdly, the participants were introduced to group estimation techniques, especially the Delphi technique. Then, the groups were given statistics of the previous two rounds of estimation available for input and discussed the estimation task in separate rooms. After the discussion session, the participants individually estimated the project once again using the same form as in the second round. The results that were reported related to group discussions showed that group discussions improved the accuracy of the size estimate. In addition group discussions improved the participants' confidence in their own estimates.

Another study that also investigated the effects of group discussion was conducted by Moløkken and Jørgensen (2004). Individual estimates tend to be over-optimistic and therefore group discussion was investigated as a way of combining individual estimates to see if it would reduce the bias towards over-optimism. Hence, an experiment was conducted with 20 software professionals as participants to investigate this issue. In the experiment, the participants first provided individual estimates of a software project that was going to be developed. After this session, the participants formed teams of four consisting of one representative from each of the different company roles (engagement manager, project manager, user analyst/designer and technical programmer). The task of each estimation group was to discuss and agree on an estimate of the same project that they had estimated as an individual task. The main result from the experiment was that after the group discussions, the estimates that were provided were less optimistic in four out of the five groups. Further, in the group discussion the participants were able to identify more of the tasks required to complete the software project than what each of the team members were able to do on their own.

McDonald (2005) has reported on a study that compared the impact of a team's experience on the team's estimation of a project's costs. The participants were software professionals who attended a course in software project management and received a task where they should develop a detailed project plan for a software development project and estimate the project's costs. Participants were placed in groups of 6–8 students working together on the project plan. A questionnaire was used to gather data about the participants' experience prior to assigning them to the different groups. This was done to ensure a diversity of the technical knowledge that is typically needed for planning this kind of software development project. Further, the different participants' years of experience in relevant areas were gathered. Statistical analyses were performed and the results showed that there was a relation between the estimated costs of the project and the average team experience. The results also showed a relation between the estimated costs and the fact that a team member had experience from a similar project. Thus, experience was shown to correlate with higher and more realistic software cost estimates.

Haugen (2006) compared the use of planning poker with unstructured group estimation to investigate whether introducing planning poker improved the estimation performance. In this empirical study, data were collected from four subsequent releases of a software development project. In the release planning, unstructured group estimation was the common way of estimating. Data were gathered from the first two releases using unstructured group estimation. Then, an intervention was performed where planning poker was introduced as a technique for estimating the next two releases. The intention of introducing planning poker was to make the estimation process more efficient and also to involve all the team members in the estimation work. Data were then collected from two releases that used planning poker as estimation technique. The results of the study showed that by using planning poker, estimates on familiar tasks were more accurate than using unstructured group discussion. However, for unfamiliar tasks the use of planning poker increased the inaccuracy of estimates.

The second study that investigated planning poker was conducted by Moløkken-Østvold et al. (2008). The purpose of this study was to explore the group processes of using planning poker and to compare the accuracy of estimates provided by planning poker and estimates achieved individually. Data from an experiment and from interviews with the participants conducted after the experiment were gathered. The estimation method that was normally used in the company was based on individual estimates of the team member in charge of that task. In the experiment half of the specific tasks that were estimated for an upcoming sprint were re-estimated by a version of the planning poker technique. The existing way of estimating in the company, individual estimates, constituted the control group in the study. The results showed that the tasks that were estimated by using planning poker were less optimistic than the results from a statistical combination of the individual estimates. In addition, the tasks that were estimated with planning poker were more accurate than the same tasks estimated individually and the statistically combined ones. Moreover, the results regarding the group processes showed that the planning poker technique influenced the work by making it easier to identify sub tasks and challenges.

Also, more explorative experimental studies of estimating in teams have been conducted. In 2004, Jørgensen reported on a large industrial study in which seven teams of software professionals estimated two different real software projects employing two different estimation approaches (Jørgensen, 2004b). The data that were collected from this study consisted of video recordings of the different teams' discussions, documentation, questionnaires and measurement. The analytic schema that was used was a content analysis, in which the videotaped discussions were categorised according to the different topics that were discussed before a quantification of the time spent on the different discussion categories was made. In this way, the estimation process of employing both a bottom-up approach and a top-down approach was documented. The typical top-down estimation process was reported to be a repeated sequence of searching for similar completed projects and discussions of issues concerned with understanding the project or the requirement specification. In between, there were instances of discussion related to how the teams' own estimation work should be conducted. When this sequence resulted in a useful project analogy, the total estimate of the project was provided. Regarding the estimation process of the bottom-up approach, this was divided into two types. One was named "sequence", where the different activity estimates were provided in roughly the same sequence in which the software development project would be executed. The second estimation process used when employing the bottom-up approach was termed “inside out”. Here, the activities related to programming, i.e., the inside of the project was estimated first before the other activities were estimated as proportions of this. Further, the content analysis revealed that the teams had difficulties in finding similar completed projects to use as comparison when employing the top-down approach and that a considerable amount of time was spent on understanding the requirement specification in the bottom-up approach. The results further showed that, on average the bottom-up approach gave the most accurate effort estimates.

The team discussions from the large industrial study described above have also been the subject for subsequent analysis. Here, the aim was to understand how software professionals arrive at an estimate understood as the step from understanding the problem to quantifying the effort of solving the problem (Jørgensen, 2005). The result from the analysis of the team discussions showed that software professionals did not explicitly explain how they reached a quantification of an estimate. Instead, the negotiations about whether to increase or decrease estimates were in most part based on references to feelings and not rational details of how the decision was made. The explanations provided by the software professionals were all concerned with detailed explications of the steps preceding the quantification. This finding was also supported by a second data set reported in the same article. Thus, the step from understanding what the problem is all about to quantifying the number of work hours to solve the problem was termed the “magic step” in judgement-based software effort estimation because software professionals could not make a rational account of how they decided upon an estimate.

These different studies of teamwork in software effort estimation show that the most-used research approach appears to be experimental studies, in which different aspects of teamwork or team discussions have been tested out to see if they might yield more realistic estimates. Further, the reported results seem to be promising in this respect. However, providing an overview of the research on teamwork in software effort estimation is not adequate to obtain a solid understanding of the research field of judgement-based software effort estimation. I the following I will therefore explicate some of the significant and relevant studies from the research that have focused on the individual expert in judgement-based software effort estimation.

2.3.2 Studies of individual expert work in judgement-based effort estimation

At Simula Research Laboratory, a research group has focused on investigating judgement-based software effort estimation in depth. This research group, called BEST (Better Estimation of Software Tasks), has focused on conducting small-scale experiments with roots in the cognitive tradition in order to understand the cognitive steps involved when software professionals decide upon an estimate. Findings from these experimental studies show that software professionals, when deciding upon an estimate, are influenced by different factors such as anchor information (Jørgensen & Carelius, 2004), request format (Jørgensen, 2006; Jørgensen & Carelius, 2004; Jørgensen & Halkjelsvik, 2008) and wishful thinking (Jørgensen & Grimstad, 2008; Jørgensen & Sjøberg, 2001). Further, the studies have shown that software professionals are over-confident regarding their own estimates (Jørgensen, Teigen, & Moløkken-Østvold, 2004) and that they are inconsistent (Grimstad & Jørgensen, 2007b). Some of the interesting findings reported by this research group are related to irrelevant and misleading information in the requirement specification and different ways of analysing the requirement specification for estimation purposes. These studies are of particular relevance to the research conducted as part of this thesis

As mentioned earlier, in estimation work, the requirement specification is considered to be the main source of information. This applies regardless of how the estimation work is organised. It is therefore important for software professionals to interpret this document and come to an understanding of what the described software system is all about. Jørgensen (2004b) briefly reported on this issue in his large estimation study of estimation teams. In the study, it was reported that a significant amount of time (49% in the bottom-up estimation approach) was spent discussing issues related to understanding the requirement specification or the project context. This implies that ways of formulating requirement specifications might have a significant impact on software professionals' understanding of the system that is going to be estimated, as almost half the time was spent on discussing such issues.

In 2007 and 2008, Jørgensen and Grimstad reported on a series of experiments that looked into different ways of formulating and analysing such requirements to investigate if irrelevant or misleading information have an impact on the effort estimates provided (Grimstad & Jørgensen, 2007a; Jørgensen & Grimstad, 2008).

The first two separate experiments investigated the impact of irrelevant information by letting the participants estimate four different versions of the same requirement specification. Estimation-irrelevant information had then been added in two of the versions of the requirement specification, one that was high level and one that was detailed. In addition, a question was added in half of the estimation tasks in the second experiment. In that case, the participants were asked to assess the relevance of the different parts of the requirement specification for their effort estimates. The results from the statistical analyses showed that participants who received the requirement specifications in which irrelevant information had been added, estimated on average, a higher number of work-hours than the participants estimating the requirement specifications without irrelevant information. Further, the results from both experiments showed that the most likely effort increased when software professionals estimated the requirement specification that contained extra information that was considered as irrelevant for estimation work (Grimstad & Jørgensen, 2007a).

Jørgensen and Grimstad (2008) also reported on three different experiments in which they investigated how misleading information could affect the decisions on effort estimates. The reason for this focus is that when estimating software projects in the industry, the estimators often have knowledge of, for example, the clients' cost expectations for the project or information about future business opportunities.

The first two experiments on misleading information investigated separately two specific circumstances: how knowledge of a desired outcome could affect the decision on an effort estimate and how information about future business opportunities could affect the estimates provided. The results reported from the first experiment indicated that the information given about the client's expectations affected the estimates provided by the estimators quite strongly. The group that received information that the clients expected effort was for 800 work hours, estimated on average 300 work hours. The other groups that received information on expected effort at 40 and 4 work hours, estimated on average 100 and 60 work hours respectively. The control group, which did not receive any information regarding the client's expectations, estimated on average 160 work hours on the task. The result from the second experiment showed that the group that received information that was believed to induce low effort estimates, estimated a lower number of work hours than the control group, 40 and 100 work hours respectively on average (median).

In the third experiment that was designed to study the effect of misleading information, ways of formulating the requirement specification were investigated. All the participants received the same specification for a programming task. However, the words that were used to describe some of the requirements varied slightly. One specification was written with words that are typically associated with small and simple tasks, another included words that usually are associated with large and complex development tasks, and a third was formulated with neutral wording. The results revealed that the way requirements are formulated strongly affected the estimates provided by the different groups. The specification with words associated with small and simple tasks was estimated to an average of 40 work hours, the specification with words associated with large and complex tasks was estimated to an average of 80 work hours while the neutral specification was estimated to an average of 50 work hours.

After discovering that software professionals' decisions on estimates were affected by the type of information available when conducting the estimation work, ways of reducing this impact were investigated. In an experiment, different techniques of analysing the requirement specification were tested to see if they could illuminate the impact of irrelevant information. The two techniques that were tested were to highlight the information that was relevant with a marker pen while reading the specification, and to use a black pen and strike through the irrelevant information in the specification before reading it again and then estimate it. The results from this study indicate that there was no effect from highlighting the relevant information when analysing the requirement. The group who stroked through the irrelevant information with a black pen indicated a positive effect, but did not come close to removing the impact.

These findings highlight important aspects regarding the use of the requirement specification in estimation work. Further, they show that both ways of formulating requirement specification and providing information—for example, about future opportunities—can have an impact on the effort estimate of a particular task or requirement. Through conducting the above-described experiments, it was possible to detect and establish this connection (Grimstad & Jørgensen, 2007a; Jørgensen & Grimstad, 2008). However, if one wants to understand more about the processes involved, i.e., how the requirement specification influences, guides and frames the estimation work, one needs to look at the communication in order to understand what is happening in the work process. My interpretation of these findings is that it is important to investigate the sense-making processes that are at stake when software professionals interpret a requirement specification, if aiming at understanding estimation work. To do this, a closer examination of teams and particularly the communicative work that software professionals do when estimating a software project needs to be opened up and investigated in depth. Thus, the social aspects of estimation work needs to be attended to in research on judgement-based software effort estimation.

2.3.3 Studies of social aspects in software development

Studies focusing on the social aspects in software development have been addressed broadly in a number of information systems (IS) journals and in related areas such as computer-supported cooperative work (CSCW) and agile development. In an opinion paper discussing the ranking of top IS journals the authors list 11 different IS journals focusing on the social study of ICT (Willcocks, Whitley, & Avgerou, 2008). To illustrate the broad use of theories and methods that have been used to investigate software development, special issues have been published; for example, the *Scandinavian Journal of information Systems* has a special issue focusing on ethnography in IS research (Pors, Henriksen, Winthereik, & Berg, 2002). Also in the journal *Information Technology and People*, a special issue on using social theory to make sense of information systems was published in 2009, volume 22 (1). Other journals in related areas have also published special issues focusing on social aspects; for instance, the journal *Computer Supported Cooperative Work (CSCW)* published a special issue on software development as cooperative work in 2009 (18 [5&6]) (Dittrich et al., 2009).

In software engineering research the focus on social aspects and the use of qualitative research methods is more scarce (Dittrich, John, Singer, & Tessem, 2007). There are several reasons for this; however, in Fugetta's (2000) road map on software processes, the author points out that there is a need to acknowledge that software development is a human-centred process where teams of people engage in creative activity and that process research from other fields should be looked into. In 2007, a special issue on qualitative software engineering research was published in the journal *Information and Software Technology*, with the aim of making qualitative research more visible and known in the software engineering field of research (Dittrich et al., 2007). Thus, what have been called "soft" issues have been accepted as valuable components to understand software engineering practices and the use of research methods from other fields have been introduced to the research field of software engineering. For example, Rönkkö introduced an ethnographic method called the documentary method of interpretation as an explanatory framework to the software engineering field through his PhD work (Rönkkö, 2005, 2007).

In the following, I will present some studies of particular relevance for the questions addressed in this thesis, which highlight the significance of social processes in software development. Although the presented studies do not address software effort estimation as such, some of the findings are relevant for the research presented in thesis.

Cohn, Sim & Lee (2009) have investigated what counts as software processes by investigating the role different types of artefacts and the participants' talk has when negotiating between what the software process model prescribes and the eventualities that arise from the software process enactments. By emphasising that, on the one hand, software processes can be modelled and, on the other hand, are enacted in a particular context, they propose that software processes are a bounded set of practices that emerge through participants' talk about models and their enactments. Martin, Rooksby, Rouncefield & Sommerville (2008) have examined cooperative work in software testing and showed how testers took the perspective of users in order to decide what tests to run. Moreover, Rooksby, Rouncefield & Sommerville (2009) have investigated testing as a situated practice and argue that testing is concerned not only with technical details. Rather, it needs to be understood and treated as cooperative work if the aim is to improve testing practices. These studies provide valuable knowledge toward understanding how software development work is carried out by focusing on social aspects and collaborative work in software development.

When dealing with future-oriented work such as software effort estimation, planning is significant. Hence, studies focusing on this aspect are of relevance for this thesis. In software development for example, it has been shown that planning is a recurrent process. Rönkkö, Diettrich and Randall (2005) investigated the use of plans in a software development project over several project phases. They showed how coordination problems were dealt with in various ways and revealed that planning documents, such as project plans and requirement specifications, provide the means to identify and act upon deviations in addition to guiding the development work. In the reported study, the requirement specification was identified as a planning document, which did not prescribe the work needed for implementation. The project members thereby had to develop new requirements and new plans in their ongoing work to create a basis for the future project work. Also in software effort estimation, the requirement specification forms the basis for attending to the planning aspect of the activity, but how this interactional work is conducted has not been examined in previous software effort estimation research. However, the study conducted by Rönkkö et al. (2005) is relevant to the research presented in this thesis as they demonstrate what software professionals do when the plan does not work out.

The short outline of studies above shows that the focus on qualitative studies and social issues in software development is increasing and is considered as an important contribution to the research field of software development (Cohn et al., 2009; Rönkkö, 2005, 2007).

2.4 Summing up: What we know and what we need to attend to

In this chapter, I have provided a description of the phenomenon judgement-based software effort estimation that is investigated in this thesis. Further, I have provided a review of relevant research within the areas of group estimation and individual estimation.

The review of studies on teamwork in software effort estimation provided an overview of the research that has been conducted of team-based software effort estimation. Firstly, it showed that the amount of research on teamwork in this field is small. Secondly, the results that have been reported about estimating in teams have been promising in terms of achieving more realistic estimates. Also, teams remember more tasks that need to be considered in estimation work than is possible by software professionals working alone. Forgotten tasks have also been reported as one of the major reasons for estimates being too low (Hughes, 1996). Thirdly, there seems to be a shared focus on measuring the accuracy of the outcome estimate in the research on teams. This implies that different aspects of teamwork have been tested to see if they lead to more realistic estimates. As a result, the main data that has been gathered and analysed when investigating group estimation have been quantitative. Only a few studies have collected interactional data in the form of verbal protocols (Passing & Shepperd, 2003) and video recordings (Jørgensen, 2004b). However, the analyses conducted have mainly been quantitative in which the commonly used unit of analysis has been the individual expert.

The research focusing on the individual expert showed that the interpretative work software professionals do in order to understand the software system or task that is described in a requirement specification is significant. However, the research conducted did not provide any explanations as to how the interpretations are done and how a shared understanding is achieved in the team. From a sociocultural perspective, it can be argued that in order to answer questions related to how this work is done and also how the requirement specification is used in estimation work, we need to look at the communicative work conducted in teams. Doing this provides a possibility for investigating the reasoning process of software professionals and can provide an understanding of how an estimate is achieved in teams.

Rönkkö and colleagues (Rönkkö, 2007; Rönkkö et al., 2005) have pointed out the need for suitable methods and concepts for understanding and incorporating the social aspects involved in software engineering. Further, a focus on social aspects in software development work is considered to be important to understand and improve this type of work. The research presented in this thesis is a contribution with respect to the focus on social aspects; however, there are a few important differences between my study and previous research. In relation to previous studies of social aspects in software development, I investigate a different phenomenon; namely, software effort estimation. This is a specific task, which takes place at different times for different purposes during the course of a software development project. The difference in relation to previous research on software effort estimation is that I have chosen a different unit of analysis than what has been commonly used. While previous research has mostly taken the individual as the unit of analysis, I have taken the social interaction as unit of analysis. Further, I focus on the micro processes of the moment-to-moment interactions when specific estimation tasks are solved. This means that the time span that is investigated is short, which provides an opportunity to investigate the specificities of the collaborative work in software effort estimation and thus provide a detailed analysis of specific tasks.

Software development is a complex practice consisting of many different software processes and tasks. Accordingly, an understanding of different types of tasks in these processes is also necessary to improve the understanding of software development work. By providing detailed analysis of the collaborative work in team-based estimation, the research presented in this thesis focuses on one such specific task; namely, software effort estimation. This has not, to my knowledge, been employed in software effort estimation research prior to the work in this thesis.

In the following chapter, I will present the theoretical perspective employed in this thesis and provide a conceptual framework for how to understand and investigate collaborative work in software effort estimation.

3 Theoretical framework

As a work practice, software effort estimation can be understood as a collaborative activity in which software professionals work together to achieve a predefined goal. Roschelle and Teasley (1995, p. 70) define collaboration as "a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem". Rather than accomplishing a task by dividing the work between the participants, Roschelle and Teasley emphasise the participants' mutual engagement and face-to-face interaction when a problem or task is solved together. The review section in Chapter 2 showed that previous research on software effort estimation has rarely taken the social aspects into account. Rather, the research has been based on methodological individualism. In this thesis, I argue that the work of estimating the effort of a software development system consists of two different but equally important dimensions. First, software effort estimation is a problem-solving activity in which problem solving is collaboratively accomplished through social interaction. Second, the task of estimating is a planning activity because it is concerned with predicting the future. Software effort estimation is therefore a quite complex problem-solving process in which absolute control is not possible. This is because of the future-oriented aspect of dealing with a system that is yet to be developed. The research in this thesis addresses the social aspects of software effort estimation work and argues that estimation work should be perceived as collaborative planning activity rather than as individual decision making regarding the quantification of the number of work hours.

As a collaborative activity, software effort estimation takes place in a context that is negotiated, shared and constructed through an external mediational framework. In this framework, the participants share the language used, the situation in which the work takes place and the activity of solving the task (Roschelle & Teasley, 1995). When solving a particular task together, a shared understanding of what the task is all about is needed (Roschelle, 1992). In software effort estimation work, the participants often come from different backgrounds in software development, which means that they bring different types of expertise or specialist competence to the problem-solving activity. The main source of information about the estimation task is the requirement specification document. This document is marked by a professional terminology in which the software system that is going to be estimated is described with text and diagrams that show the logic of the workflow in the software system. Moreover, this requirement specification is often written and developed by software professionals other than those who are conducting the estimation work. This means that the software professionals who are conducting the estimation work need to make sense of and achieve a shared understanding of this document in order to solve the task. In addition, the participants also have to align their different areas of expertise. Sharing skills and understandings are emergent in social interaction and can therefore be understood as an accomplishment that is achieved through communication.

3.1 A sociocultural perspective on software effort estimation

When studying the collaborative activity of software effort estimation, it is necessary to have a theoretical perspective that provides an understanding of human activity at different levels (Valsiner & Van der Veer, 2000; Wells, 1999). The three interrelated aspects of human activity that need to be understood in relation to software effort estimation work are 1) the institutional practice, 2) the individual development and 3) the dialogue and activity.

The institutional practice of software effort estimation can be understood as historically driven. In this regard, concrete artefacts represent what can be named collective knowledge or cultural resources that have evolved over time. These artefacts include the requirement specification, a professional language from the knowledge domain of software development and different techniques and approaches to conduct the estimation work. For example, perceived as institutional practice, software effort estimation rests upon certain ways of doing the work that are embedded in routines, rules and different cultural tools. This means that in an institutional practice, certain ways of understanding the activity prevail. For example, estimation work is often organised in meetings. These are meetings that are designed for the specific purpose of conducting the work of estimating a software project. Thus, the interactional and communicative work that is carried out in an institutionalised setting has a specific purpose, i.e., the work has an agenda (Suchman, 2007). This agenda needs to be well known by the different participants in order for them to carry out the work as intended.

Secondly, the software professionals who take part in the institutional practice of software effort estimation hold sets of individual and personal knowledge and experiences that they bring into the institutional practice and make use of when solving the task. This individual knowledge is acquired both through specialised education and work practices. It therefore needs to be made relevant to be utilised in software effort estimation work.

Thirdly, when working in a team, it is through dialogue and interactional work that the task of estimating a software development project gets solved. This takes place in the intersection between the collective knowledge and cultural resources that the institutional practice holds, and the personal knowledge and experiences that the different participants bring into the practice. Thus, the dialogue and the interactional achievements develops in the intersection between enacting upon cultural resources and the personal knowledge held by the participants. Software effort estimation work is accordingly conducted through series of moment-to-moment interactions amongst the participants. Through this interactional work, they draw upon their own individual knowledge and knowledge that are embedded in the social practice in which the estimation work is conducted. Hence, when following a sociocultural perspective, it is through social interaction that knowledge is used, sustained, developed, and created.

The sociocultural perspective, also called the socio-genetic perspective, rests upon a set of assumptions that makes it possible to understand human activity at these different levels. This perspective has its origins in the work of Lev Vygotsky (1896–1934) (Vygotsky, 1978, 1986) and rests upon an ontological and developmental assumption that human minds are social in nature and that the human personality emerges through social experiences (De Graaf & Maier, 1994; Valsiner & Van der Veer, 2000). The sociocultural perspective also distinguishes between development at different levels and in different time spans. The level of sociogenesis is concerned with describing and explaining the development of social practices in their social and historical context. It thereby relates to the development of institutional practices. The level of ontogenesis is concerned with the development of individuals, which is socially constructed through their lives whilst at the level of micro genesis, a description of the moment-to-moment actions of different individuals when engaging in situated activities is the focus (Billett, 2003; Ludvigsen, 2009). It is at the level of micro genesis that the phenomenon of software effort estimation work is played out. This happens in interrelation with the individual's ontogenesis, i.e., the participants in the practice, and the sociogenesis, which is the institutional practice of software effort estimation. In this thesis, therefore, it represents the empirical level of interest (Ludvigsen, 2009).

The second important assumption in the sociocultural perspective is the epistemological one, which rests on the idea of the co-construction of knowledge (Valsiner, 1994). This means that participants in a practice construct knowledge together as part of an intellectual interdependency (Valsiner & Van der Veer, 2000). The sociocultural perspective thereby takes as point of departure the fact that human beings are inherently social in character and that we interact with each other and with objects in our environment as ways of learning, solving problems and sharing and creating knowledge. In order to understand the basic assumptions underlying the sociocultural perspective, there are some core premises that need to be explicated: 1) situated character of action, 2) mediated action and 3) actions as interactional accomplishments.

3.1.1 The situated character of action

A core premise in the sociocultural perspective is that all human action takes place in a cultural, institutional and historical context, which means that human action is institutionally, historically and culturally situated. This implies that action is shaped by the different situational factors that are available and the way we solve problems, reason or act is related to the context and the cultural tools that are made available in a particular context. Situated actions are thereby emergent through moment-to-moment interactions between the participants in a practice and between the participants and the environment in which their actions take place (Suchman, 1987, 2007). Linell (2009) makes a distinction between realised context and contextual resources. The realised context is what participants make relevant through communication in situ. The contextual resources are what can potentially be made relevant and meaningful in a given context. One such contextual resource is the categories that are embedded in an institutional practice. Such institutional categories are collective ways of understanding, clarifying and categorising people, actions and events (Bowker & Star, 1999). Further, they are resources for sense making and contextualising tools in which perspectives and sense making can be collectively shared (Mäkitalo, 2003; Mäkitalo & Säljö, 2002). These categories are part of the historical development and it is through using specific categories that certain aspects of the institutional practice are made relevant and thus connect documents, drawings and collective knowledge. A specific institutional practice, such as software effort estimation, incorporates expectations, procedures, technologies and goals "that only find actual form and purpose when they are enacted in particular ways in particular circumstances" (Billett, 2003, p. 136). This means that the ways in which estimation work is enacted will be shaped by different types of situational factors such as local ways of interacting, ways of solving the estimation tasks, negotiations and different kinds of artefacts.

Software effort estimation is thus situated and relational because the properties of people, groups, communities and also material artefacts are developed and given meaning only in relation to other people, groups or artefacts. These properties are not independent of each other, they are not stable and they cannot be separated from their social and historical context (Østerlund & Carlile, 2005). This makes software effort estimation a situated activity and, by investigating situated activities, it is possible to understand the ways in which problem solving is collectively accomplished in specific situations.

In relation to the planning dimension of estimation work, the way planning is understood is important. A well known and also common way of conceiving planning is that plans determine action. This view is embraced by cognitive science among others and has had a huge impact on how planning is understood. According to this planning model, a plan is treated as a description of a particular sequence of actions that are developed to achieve a particular goal (Suchman, 2007). This way of conceiving planning was criticised by Suchman (1987), who proposed that plans need to be understood as resources and discursive artefacts for the practical deliberation that people engage in regarding their actions (Suchman, 2003, 2007).

Following the sociocultural perspective, planning is thereby understood as a situated action, in which people plan through communicative work by attending to what is and what could be in the future. Hence, planning takes place through interactional moves that go back and forth, and is situated in the institutional practice.

3.1.2 Mediated action

Following a sociocultural perspective, the second core premise I would like to emphasise is that all human action is mediated. The notion of mediation implies that we as humans use different artefacts and tools, known as mediational means, as integrated parts of our social practices. This means that we do not interact with our environment and surrounding world in a neutral and immediate way. Rather, we perceive and make experiences through the use of different tools, that are products of cultural and historical development, when interacting with our environment and each other (Wertsch, 1991). The action that individuals do when employing mediational means is referred to as "individual(s) acting-with-mediational-means" (Wertsch, 1991, p. 12). When following a sociocultural perspective, how individuals act with mediational means is fundamental for understanding human learning, problem solving and knowledge creation. The concept of mediational means or tools is expanded beyond the common interpretation that a tool is a physical artefact. In addition to physical or material artefacts also language, concepts, structures of reasoning and types of discourse are considered as tools (Resnick, Pontecorvo, & Säljö, 1997). Originally, Vygotsky distinguished between tools and signs (Vygotsky, 1978, 1986). However, this distinction has been shown not to be fruitful because, often, the use of various types of tools requires some kind of intellectual capability. Instead of separating the two, researchers have suggested that cultural tools should be perceived as having a physical side and an intellectual side (Säljö, 2005).

Knowledge and experiences, which have been accumulated and developed over time, are built into these different artefacts that we use in our everyday lives. The most important mediating tool is considered to be language. It is a resource for creating knowledge and meaning of the world, sharing experiences and solving problems. This is because of the flexible relation between linguistic expression and the phenomenon that the expression is referring to. All communicative activities are mediated through intellectual tools such as language. In addition, language is an important part of the communicative and interactional processes between people. It is a tool for acting in practices as well as an arena for social interaction (Säljö, 2001b). Also, specific types of languages make it possible to achieve a shared understanding of a particular phenomena in expert work. Wertsch argues that "human action typically employs 'mediational means' such as tools and language, and that these mediational means shape the action in essential ways" (Wertsch, 1991, p. 12). To understand mental action, there is a need for understanding the "semiotic devices used to mediate such action" (Wertsch, 1991, p. 13). Hence, mediated action has as a premise that there is a close link between communicative processes that are inherently social and certain aspects of individual mental processes.

Wertsch has studied mediated action in depth and, in his book *Mind as Action* identified ten different properties of this kind of action (Wertsch, 1998). Related to the study of collaborative work in software effort estimation, I will highlight some of these properties in the following as important for understanding the role of mediated action in estimation work.

The first point that Wertsch makes is that "there is an irreducible tension between an agent and the mediational means". What this means is that certain mediational means hold particular sets of qualities that make it possible to use and thus solve specific problems. However, the mediational mean in itself is not enough to, for example, solve a problem, and neither is the practitioner seen in isolation. Rather, it requires active sense making by the practitioners. This implies that without such mediational means it will be difficult to solve certain problems. In software effort estimation, the requirement specification as a mediational mean holds a particular set of qualities, which are needed for the task of estimating the effort of a software project. Without a requirement specification, the estimation work would be even more difficult as no description of the software system to be developed would exist other than perhaps in the heads of the people or customers who want or need the particular system being developed. However, the presence of a requirement specification does not magically solve an estimation task; instead, it has to be used and made sense of by software professionals who are skilled in understanding requirement specification documents.

The second point that is of relevance, which Wertsch put forth, is related to the materiality of mediational means. As previously mentioned, mediational means can be physical objects, which are often termed artefacts or tools. For example, physical objects or artefacts can be writings, such as books or documents, or technical drawings involving the use of diagrams and different types of schemas, which are turned into tools and used in social practices. Due to the materiality of such objects, they can, as Wertsch puts it, "continue to exist as physical objects even when not incorporated in the flow of action" (Wertsch, 1998, p. 30). In software effort estimation work, for example, the requirement specification will be one such material tool that will continue to exist throughout the software development process. Even though it might be changed and improved as the development work goes along, it will, in the end, become historical data about how a particular system was planned and can thereby function as a resource when new tasks for solving arise.

3.1.3 Social interaction and interactional achievements

The third core premise in the sociocultural perspective that I would like to explicate is that knowledge and understanding needs to be achieved in social interaction. It is in social interaction that people use language, concepts, align their expertise and activate different types of resources that constitute what they achieve in interaction. It is through social interaction that knowledge is shared and constructed and it is where interpretations and meaning making of objects, events and actions occurs. Knowledge is therefore co-constructed as part of an intellectual interdependency where people interact, not only with other people, but also with cultural tools and objects in the environment. Thus, it is in the interaction between participants in a practice and the different cultural tools that knowledge comes into use (Greeno et al., 1996). The interdependency is a result of the communicative actions that are joint activities in which people engage in interaction by taking turns at talking. Communicative actions can thereby be conceived as collaborative accomplishments (Linell, 1998a) in which knowledge is a resource that is made relevant for use. It is therefore important to understand how these interactional accomplishments come about. This means that we need to understand how software professionals make use of the cultural resources available in the institutional practice as well as how they draw upon their own specialist knowledge and previous experience in the estimation work. Following the sociocultural perspective, interactional accomplishments are carried out by activating individual knowledge and making sense of cultural resources and this is how the task of estimating a software development system gets solved.

As previously mentioned, the estimation task has an agenda, which the different participants need to know well in order to carry out the work of estimating a software project as intended. Even though there is an agenda for the interactional work, the face-to-face interaction makes it possible to attend to and solve problems that can emerge as the work proceeds. Hence, the agenda and other constraints do not control the interaction. Rather, it is resources for the meaning-making actions and the struggle for achieving shared understandings of the different problems that are attended to in the team's collaborative work.

However, relying on institutionalised and established ways of conducting this work is not sufficient. The relational interdependencies among the participants, i.e., among the software professionals, among the software professionals and the material artefacts that they use and among individuals and collectives can be perceived as the core of conducting estimation work in teams.

One of the main challenges in software effort estimation work is to achieve a shared understanding of the requirement specification, i.e., the described software system that is going to be estimated (Sommerville, 2007). Understanding this document posits a challenge as it is marked by a professional terminology and often written and developed by someone other than those who are doing the estimation work.

Seen from a sociocultural perspective, such understanding is achieved and shared through interpretative work in which meanings and interpretations are produced on a turn-by-turn/moment-to-moment basis through talk in interaction. People's meanings become known through social interaction, plus new ways of thinking, reasoning and also acting can be apprehended when solving new or complex problems. Communication can thereby be perceived as serving as a link between what we think and what we do (Säljö, 2001b). However, people do not always reveal what they think nor are they able to give an account of what they think. In short, some cognitive processes are thus not possible to "be brought in to language in an accountable manner" (Linell, 2009, p. 15).

According to a sociocultural perspective, what takes place when people solve problems and make decisions is the result of social interaction and communicative work. From this perspective, what is attended to and what is said is "thus not an external version of what is clear in our minds but rather an attempt to communicate ideas to respond to initiatives in situated practices" (Säljö, 2001a, p. 114). The focus on the interactional accomplishments provides possibilities for explicating how estimation teams reason, what kind of work they do and also how they reach a decision about an estimate. Above, I have explicated some of the relevant core premises of the sociocultural perspective that are relevant for the research presented in this thesis. This thesis focuses on software effort estimation as collaborative activity and sees estimation work as an emergent practice that takes place within an institutional environment. To understand the collaborative activity, there is a need for investigating estimation work through social interaction. By following the situated activity of software effort estimation and investigating the ways in which the institutional level and the personal knowledge and experiences influence and play a part in the collaborative problem-solving activity, we can develop our understanding of the social aspects of software effort estimation. In the following, I will explicate some of the important theoretical concepts that are needed to understand what kind of interactional achievements are needed to accomplish the task of estimating a software project.

3.1.4 Distributed expertise

Firstly, I would like to explicate the notion of expertise being distributed, which, following a sociocultural perspective, comes as a consequence of the distributed character of knowledge. That the expertise is distributed means that it is not stable or well-defined acquired knowledge, rather it is emergent in the ongoing activity (Engeström, 1992). Expertise is here seen as interactional accomplishments, which means that expert knowledge resides not only in individuals' heads but is alive in people's interaction and the environment that people interact in. Hence, expertise can be perceived as a collective resource. As concerns to software effort estimation, the software professionals that work together in a team often hold different kinds of specialist competences in different areas of software development, such as programming, databases, architecture, testing and project management. Though they are sharing a professional language, the situation and the work activity, the team members perceive and understand the different artefacts that are used, such as the requirement specification, from their respective areas of expertise. The different kinds of specialist knowledge and skills they hold therefore need to be aligned and shared with the other team members so that they are able to accomplish the estimation task collectively (Schatzki, 2001). Eklund, Mäkitalo & Säljö (2010) have shown in their study of IT-support units the importance of sharing knowledge and expertise to be able to continue in the work process.

The traditional view of expertise can be said to be vertical in the sense that some individuals or participants in a practice have more knowledge than others. This view is differentiating what can be termed levels of expertise, i.e., levels of knowledge and skills, and assumes a uniform and singular model of what counts as expertise in a particular practice. This vertical dimension of expertise is important to acknowledge but, for a more complete understanding of how expert knowledge is acquired and used, we also need to consider the horizontal dimension of expertise (Engeström, Engeström, & Kärkkäinen, 1995). A horizontal dimension of expertise takes into account how experts are moving between different contexts and therefore need to negotiate and combine ingredients across contexts to solve particular tasks (Engeström et al., 1995; Tuomi-Gröhn, Engeström, & Young, 2003). The horizontal dimension of expertise also relates to the more-recent notion of networked expertise because networked expertise arises through the co-construction of knowledge and the collaborative work of solving problems (Hakkarainen, Palonen, Paavlova, & Lehtinen, 2004).

In previous research on software effort estimation, the focus has been on the vertical dimension of expertise. The reason for this is that a deeper and more specialist knowledge about estimation work and relevant experience is believed to lead to the achievement of more-realistic estimates. The horizontal dimension of expertise in estimation, on the other hand, is concerned with how software professionals can apply their specialist knowledge across contexts, as well as how they can realise the different artefacts' potential. This is an important aspect when collaborative problem solving is at stake.

Developing and acquiring specialist or expertise knowledge in different fields shows that knowledge is closely connected to the context in which it is acquired. This means that knowledge is dependent on communication to be kept alive across generations (Linell, 2009). The use and also the development of different kinds of expertise take place in a series of interactional achievements. It is then communication that makes it possible to develop specialised forms of knowledge through discursive practices in which precise communication about certain aspects of the world is made possible in specific settings (Säljö, 1999).

3.1.5 The role of sense making in social interaction

To accomplish solving the estimation task through social interaction, interpretations and meanings of different aspects have to be generated. This is achieved through situated interaction and communication and depends on what is of interest to the participants (Greeno, 1998; Rommetveit, 1992). The process of making meaning can be termed sense-making practices. Following Linell (2009, p. 235) "sense-making is about what is meant and made known in real-life situations, in which people make certain interpretations relevant there and then". In sense-making practices, it is important to differentiate between meaning and meaning potential. While meaning is related to situations, meaning potentials belong to what Linell calls "traditions of languaging".

The notion of meaning potential can be understood as a set of semantic resources that together with contextual factors are used to achieve situated meaning. The meaning potentials of lexical items and grammatical constructions have been defined by Norèn and Linell as

> the set of properties which together with contextual factors, including features of the linguistic co-text as well as various situational conditions, make possible all the usages and interpretations of the word or construction that language users find reasonably correct, or plainly reasonable in the actual situations of use (Norèn & Linell, 2007, p. 389).

It is the meaning potential that moves between contexts and inhabits both history and structure, but the meaning potential needs to be realised in a particular context for it to give meaning. Linell (2009) uses a set of concepts to describe distinctive features of meaning potentials, which are open, abstract, structured and rich. Being open means that meaning potentials are open to a certain extent, because no meanings exist that are "entirely fixed, stable and always valid" (Linell, 2009, p. 342). That potentials are structured refers to the possibility that there may be an abstract core aspect. That meaning potentials are rich means that the boundary between what are defined and what is encyclopaedic semantics is not clear cut. Since meaning potentials are abstract, they can only contribute to the making of meaning in certain contexts. Meanings, hence, what people mean in situ, are interactional accomplishments. Thus, meaning making occurs through the articulation of meaning potential and interpretations of, for example, texts or communicative actions that are relevant for the participants' interaction and exchanges of talk in a particular context (Linell, 2009; Rommetveit, 1992; Van Oers, 1998).

To demonstrate the importance of meaning potentials, I would like to recount the story about Mr. Smith created by the sociologist Herbert Mentzel and extended by the psychologist Ragnar Rommetveit (2003) as an illustration: Mr Smith, who is a fireman, one Sunday morning is out in the garden pushing a machine around his lawn. His wife Mrs. Smith, is in the kitchen drinking her morning coffee. What is interesting about this setting is the different ways sense-making of what is going on in the garden occurs:

> A neighbour prying into the miserable marital relations of the Smiths may tell us that she "sees" Mr. Smith avoiding the company of his wife. And that may indeed be what Mrs. Smith feels, left alone in the kitchen with her morning coffee. But when the phone rings and her friend Betty asks, "That lazy husband of yours, is he still in bed?" Mrs. Smith answers, "No, Mr. Smith is working, he is mowing the lawn."
>
> A short time afterward, Mrs. Smith receives another call, this time from Mr. Johnson, who, she takes for granted, is ringing up to find out whether her husband is on the job or free to go fishing with him. So, when he asks, "Is your husband working this morning?" she answers, "No, Mr. Smith is not working, he is mowing the lawn." Mrs. Smith is telling Betty that her husband is working and Mr. Johnson that he is not working, but on both occasions is telling the truth(s) about Mr. Smith's mowing their lawn. (Rommetveit, 2003, p. 215)

The meaning potential of the word "working" is here situated and realised by the two people who are having the conversation at the time when Mrs. Smith is talking to

Betty and when she is talking to Mr. Johnson. Even though both Betty and Mr. Johnson share a common language, the ways of understanding Mr. Smith's activity is quite different. This story demonstrates quite nicely that a word or linguistic expression does not have just one lexical meaning that is common across contexts. Further, the story shows that we always have to make meaning in a concrete context and share this understanding so that it can be collectively understood amongst the participants.

Engaging in sense-making practices and thus realising the meaning potential of different cultural tools is an important and also a large part of software effort estimation work. Achieving a shared understanding of the main cultural tool, i.e., the requirement specification, is both necessary and crucial for both the problem-solving aspect and the planning aspect of estimation work. In such meaning-making activities, a tacit shared understanding exists of the kinds of actions that are appropriate. As concerns software effort estimation, this takes the form of topics that are acceptable to talk about and how elaborations about these certain topics are performed. Suchman (2007) calls this mutual intelligibility of action. To be able to participate in this action and elaborate on relevant topics, different types of knowledge need to be made sense of and used.

3.1.6 Recontextualisation and collective remembering

As previously mentioned, the software professionals that take part in the estimation work hold a set of individual and personal knowledge and experiences that they use in their work process. However, using knowledge and experiences across contexts is not a straightforward process. Before knowledge and experiences can be used in different contexts and in different ways, they need to be articulated, made relevant and adapted so that they fit the specific context in which they are intended to be used. This process of adaptation or transformation is known as recontextualisation. Recontextualisation is defined as "the dynamic transfer-and-transformation of something from one discourse/text-in-context to another" (Linell, 1998a, p. 154). This means that some parts or aspects from one context are adapted or transformed to fit a different context. For example, this can be knowledge, facts, arguments or ways of seeing, thinking or acting. Through a recontextualisation process, parts or aspects are often subject to change such as simplifications, explications or shortenings and thus “recontextualisation is never a pure transfer of a fixed meaning” (Linell, 1998a, p. 155).

Recontextualisation can occur in at least three different ways (Linell, 2009). Firstly, it can occur within the same conversation. Secondly, it can occur as reuse between discourses. Thirdly, it can occur between genres or activity types. When recontextualisation occurs within the same conversation, it implies that the participants in a particular interaction are making use of the same or similar ideas and expressions multiple times. Recontextualisation as reuse between discourses implies that elements from one specific discourse are related to a different discourse. The third type of recontextualisation is more abstract and involves, for example, the use of routines from one event that are used in a different event in the sense that genre or activities are borrowed from one context or discourse to another. What is common across these three types of recontextualisation is that they involve, though at different levels, the generation and reinterpretation of meanings (Linell, 1998a; Van Oers, 1998). Realising meaning potentials, interpreting documents and sharing understandings through social interaction require the use of knowledge in some ways. As a result of this knowledge needs to be remembered before it can be recontextualised and used.

Moreover, when using knowledge and previous experiences to solve present working tasks, it is crucial that the appropriate knowledge and experiences are remembered. Following the sociocultural perspective, the act of remembering in itself can be conceived as an interactional accomplishment. The act of remembering is thereby understood as a “social activity where individual experience is necessarily mediated by collective experience” (Middleton & Brown, 2005, p. 12). This means that rather than seeing memory as a way of storing past experiences that are retrieved whenever needed, as would be the case for a cognitive perspective, memory is conceived as “accomplishments that occur in the course of communicative action” (Middleton & Brown, 2005, p. 85). Social interaction is therefore fundamental to the act of remembering. Understanding the act of remembering in this way opens up a discursive approach to remembering and thus understand how remembering is accomplished and organised through communicative action.

When people remember, they often reconstruct something from their past in the present so that it can be used in the actions that they are engaged in at the moment. In this sense, remembering is shaped by the fact that we are living in a sociocultural world, sharing our experiences with others. Further, what is remembered is selected and transformed by means of communicative actions and thus mediated through language (Middleton & Brown, 2005; Wertsch, 2002). Hence, remembering is situated. There are, however, some key elements that need to be taken into consideration when treating remembering as a situated communicative practice and thus a discursive practice.

When people are communicating, they take turns at talking by acting towards and responding to one another. Thus, the conversation is sequentially organised. This is necessary to keep a conversation going. Such conversational action where people engage in joint remembering, also known as collective remembering (Middleton & Brown, 2005), is quite common as part of both our daily lives but also in work situations where, for example, teams are solving problems and need to remember their past experiences. As software effort estimation work is interactional and communicative, it is by engaging in collective remembering that software professionals recall their experiences and jointly remember appropriate experiences and knowledge that can be used in the work process.

Closely connected to remembering as interactive and sequentially organised is the co-option of other speakers into engaging in remembering activities. As Middleton and Brown (2005, p. 92) put it, “In the course of such conversations, there are particular points at which the participants seem to ‘stand back’ from what they are doing and put into words that they are remembering or trying to remember or had forgotten”. Moreover, to assist in remembering activities, different artefacts and cultural tools are used. Language is one such tool and the construction of narratives is therefore one way that remembering is mediated by the recall of past events (Wertsch, 2002). Being co-opted into a project of remembering is linked to the membership of recollecting an event or a past experience. Through experience claims, others that participate in the conversation become interactively committed to the relevance of these claims and thus it becomes a collective concern rather than only an individual one.

I have briefly stated that acts of remembering are mediated by cultural tools. Radley (1990) states that some objects are even designed for the purpose of helping us remember. We need to take into account that we live in a material world where not only social interaction with other people helps us remember, but also our interactions with different objects and tools in our surrounding environment are a part of our remembering activities. Such objects and artefacts are a part of the material world and have the capability that they can outlive humans. Therefore, they not only function to evoke memory but to sustain ideologies, traditions and knowledge about individual people, as well as cultures (Radley, 1990; Wertsch, 2002).

Connecting the notion of collective remembering to estimation as a work practice, we know that in this work practice different resources for remembering are used. However, as estimation work is interactional and communicative, it is through what is termed conversational remembering that software professionals recall their experiences and engage in joint remembering. Moreover, both sense-making and remembering are mediated by different types of cultural tools or artefacts (Linell, 2009).

3.1.7 The use of cultural tools

In an institutional practice, cultural tools have developed over time and contain what can be named collective knowledge or resources. Related to software effort estimation, there are several cultural tools that are used as mediational means. First of all, there is the communication and interaction between the different team members. In this interaction, a professional language is used and to be able to follow and participate in the communicative work, knowledge about and familiarity with this professional language is necessary. Language, as a cultural tool, is thereby crucial for the software professionals to interpret, make meaning and construct knowledge of the different material tools to accomplish the estimation work. Thus, language and concepts can be conceived as the most important cultural tools that we as humans can use (Wertsch, 1991, 1998).

The main material artefact that is used in estimation work is the requirement specification document. This document is marked by a professional terminology in terms of concepts from a particular knowledge domain as well as particular ways of representing software systems. Design solutions and flow diagrams are such ways of representing the logic of the workflow in software systems. The diagrams, called flow diagrams, have the character of being inscriptions, as they are “graphical representations recorded in and available through some medium (e.g., paper, computer monitor)” (Roth & McGinn, 1998, p. 35). In estimation work, these inscriptions represent the software system, and by existing in a material form, they can be shared with others. Further, they can be the object that is talked about when problems are solved, thus transposing the activity of developing a software system from the heads of the designers of the system to social arenas where the different parts of the software development process take place. These inscriptions may facilitate communication between participants in a practice, though knowledge about what is represented in the inscriptions is needed to be able to participate in the related activities.

In an institutional practice like software effort estimation, these cultural tools are resources that are used when solving new estimation tasks. Hence, the estimation work is mediated through communicative actions that take form as professional language in talk and also in physical artefacts where professional language is used in written form. It is through communicative action like interpreting and negotiating that shared understandings of the described system are achieved in the team. This is crucial to be able to estimate the effort of a software system.

The cultural tools make it possible to engage in the planning activity of estimation work. The requirement specification is of particular relevance in this aspect as it is a document describing what should be or is going to be developed in the future. The use of the requirement specification as a planning document in software engineering has been studied by Rönkkö et al. (2005), but they did not focus on the requirement specification as a resource for planning activity. In software effort estimation, the requirement specification as a cultural tool is a resource for the planning activity and functions as a framework for solving the task, though it is dependent on particular social and material circumstances in this context (Suchman, 2007). It is by interpreting the requirement specification that the software professionals engage in the planning activity. Further, as the requirement specification is in need of interpretation, it also posits challenges and problems that the team needs to elaborate on and achieve shared understandings of. Hence, investigating how the requirement specification is used in estimation work provides a possibility to understand the situated character of planning (Suchman, 2007).

These different cultural tools that are available in the institutional practice create direction as well as serve as a framework for conducting the estimation work. Thus, the cultural tools are a point of departure as well as guiding principles in the work process. The routines, rules and different cultural tools have been developed, improved and changed over the course of time and can therefore be conceived as historically driven events. They represent the collective knowledge available when solving an estimation task. Thus, it is crucial to understand how software professionals use these different cultural tools in their collaborative problem solving.

3.2 Summing up and presenting a conceptual framework

A theoretical perspective provides a set of lenses for how the world, society and human development can be perceived. It provides ontological and epistemological assumptions about human activity and development and guides and structures research by providing a set of concepts as well as themes and parts that are interesting to focus on. The point of departure for the research within this thesis is to investigate the collaborative activity in software effort estimation. In this chapter, I have outlined a theoretical perspective with core concepts that are central to how software effort estimation can be understood as collaborative planning activity.

The sociocultural perspective has as a premise that all human action is historically and culturally *situated*. A second premise is that all human action is *mediated* by different types of mediational means such as language and physical artefacts. The third premise is related to the importance of *social interaction* because it is through social interaction that knowledge is constructed shared and developed. With these core premises as a backdrop, a set of concepts that explicates how software effort estimation can be understood as collaborative planning activity is summarised into a conceptual framework for how it is studied in the research that is presented in this thesis.

When following the sociocultural perspective, it is through *social interaction* that the estimation work is conducted. This is also how the estimation task gets solved.

Software effort estimation is part of an *institutional practice* that has developed over time and holds a set of *collective knowledge* that is used as a resource in the estimation work. This knowledge is represented in different types of *cultural tools*. These cultural tools can, for example, be the professional language from the knowledge domain of software engineering, the requirement specification document that describes the software system, the different estimation techniques and guidelines that are used in the estimation work. In addition to the collective knowledge in the institutional practice, the different software professionals bring with them their *personal knowledge and experiences* into the practice. It is in the intersection between the resources available in the institutional practice and the personal knowledge the participants bring to the practice that the estimation work gets done through social interaction.

At the core of this conceptual framework are *interactional accomplishments*. In software effort estimation work, participants in a team need to engage in different types of *tool-mediated actions*. They engage in *meaning-making processes* to make sense of the particular estimation task, achieve shared understandings, close gaps, make knowledge relevant and use the different material cultural tools, which makes it possible to move on in their work. This is accomplished by realising *meaning potentials* in situ. Tools and concepts do not have a fixed meaning that can be used across contexts. Rather, the meaning potentials need to be realised in a particular context for them to have meaning that is relevant here and now. Engaging in meaning-making processes is, however, not the only type of interactional work that participants in software estimation work need to engage in. For the software professionals to be able to utilise their personal knowledge and experiences, these have to be transformed and adapted to fit the context where such are intended to be used. The process of adapting knowledge is called *recontextualisation* and it is a crucial process for understanding how software professionals draw on and use their individual knowledge, experiences and collective resources when solving present working tasks. Moreover, as *expertise is distributed* it also needs to be aligned, shared and made relevant in the interactional work. Closely connected to the recontextualisation process is the act of remembering. Before previous knowledge and experiences can be recontextualised, they need to be made sense of and the appropriate knowledge also needs to be remembered. The act of remembering is referred to as *collective remembering*, because it is understood as an interactional accomplishment that occurs through joint effort.

The theoretical perspective employed in this thesis has some implications for the analytical approach that is chosen. When following a sociocultural perspective, it is what people say and do that is the focus of attention and thus possible to follow and not how and what people think (Säljö, 1994, 2001a). This does not mean that thinking is excluded, but no claims are made about thinking. Language can therefore be conceived as a medium for action and is used for concrete purposes in specific contexts. In Säljö's words, "when you talk, you do thing" (Säljö, 1996, p. 14).

When investigating the collaborative activity in software effort estimation, it is important to grasp how software professionals' work is mediated by the different cultural tools. This means that it is the level of social interaction that needs to be attended to and investigated. Accordingly, the research conducted as part of this thesis is a study of practice following moment-to-moment interaction in the micro-genesis but it also takes into account how the socio- and ontogenesis implicates what goes on in this activity. The level and unit of analysis thereby needs to be adapted to this specific research project (Säljö, 2009), which aims at understanding the collaborative work in software effort estimation. Thus, the level of analysis that is chosen is social interaction, in which the moment-to-moment interaction constitutes the unit of analysis.

In addition to the theoretical conceptual framework that explicates the interactional achievements, I also need a set of intermediate concepts as sensitising means that can open up the interactional achievements and meaning-making processes. To open up the different meaning-making processes and grasp the different types of interactional achievements and specify the different situated actions that occur, I have used the following intermediate concepts: *elaboration, clarification, specification, orientation, positioning* and *justification*. As an analytic concept, on the other hand, recontextualisation is used to grasp what happens with knowledge over time and across contexts. These analytical concepts will be explained in the next chapter describing the method and analytical approach in depth.

4 Description of empirical material and methods

Investigating the collaborative work in software effort estimation requires a solid data set. The research presented in this thesis consists of three separate in-depth studies of three different estimation approaches to explicate software effort estimation as collaborative planning activity.

The first section of this chapter provides a description of the empirical context and the data material that have been gathered through two separate studies—one quasi-experimental study and one naturalistic study—that constitute the empirical basis for investigating the three estimation approaches: bottom-up, top-down and planning poker. The second section discusses the methodological considerations and the analytical procedures for how interaction analysis has been employed in the three different articles. This includes how the data have been prepared, selected and analysed by using a set of analytical concepts as sensitising means. The third section discusses the credibility of the research that has been conducted as a part of this thesis in regards to the issues of validity, reliability and generalisation.

4.1 A quasi-experimental study of the bottom-up and the top-down approaches

Articles I and II in the second part of this thesis build upon a set of data that was gathered through a quasi-experimental design as part of a large software effort estimation study. The study was conducted in a Norwegian branch of an international IT-consultancy company in 2002. It was designed to investigate the two different estimation approaches known as top-down and bottom-up. As part of the data collection from this study, video recordings were made of the different teams' discussions when they conducted the work of estimating software projects. The results from this large industrial estimation study have previously been reported in Jørgensen (2004b; 2005), in which the analytical schema used has been content analysis and quantification. I was given permission to use the video recordings as part of the empirical data for my PhD project. The in-depth analyses that have been conducted and reported in Articles I and II in this thesis thereby build upon and expand the results that Jørgensen previously has reported.

The setup of this large software effort estimation study was quasi-experimental in character. The following description is adapted from the detailed description of the study provided in Jørgensen (2004b). In this estimation study, seven teams of software professionals were organised and asked to estimate two legitimate software projects. Even though the study is quasi-experimental in character, authenticity of the estimation process was aspired to in the design of the study. Therefore, the real requirement specifications received from the company's customers were used as estimation tasks, and software professionals working in the given firm were hired as participants. Each team consisted of one project manager and one or two developers. A senior manager in the company put the teams together and ensured that the teams had sufficient development and estimation competence to perform realistic estimation of the two software projects.

The estimation teams had access to the information they normally have when they are estimating projects in their daily work at the company. These consisted of real requirement specifications of software projects, the company's online database of completed projects and the opportunity to phone colleagues and to collect documents from their own offices and computers while conducting the estimation work. The software projects the participants were estimating in this study had already been or were in the progress of being developed by other employees in the company at the time of data collection. The participants were therefore not allowed to talk to the employees who had worked on the development of the system during the quasi-experiment. Further, the participants received instructions on how to employ the two estimation approaches: bottom-up and top-down together with a work breakdown structure that was identical to what was applied to most of the projects in the company when estimation work was conducted. These descriptions are enclosed as appendixes in Articles I and II in Part II of this thesis.

The participants were divided into two groups and first estimated project A and then project B. Group 1 were instructed to employ a bottom-up estimation approach on project A and a top-down estimation approach on project B. Group 2 employed the estimation approaches in opposite order. Table 1 below shows the order in which the estimation approaches were employed when.

Participants group	Project A	Project B
Group 1	Bottom-up	Top-down
Group 2	Top-down	Bottom-up

Table 1 The order of estimation approach employed in the two groups.

The study was conducted at the company's premises and the participants were seated in a meeting room during their work process. In addition to the team, an experiment leader was also present in the room, and a video camera recorded the team's discussions. The camera was placed on a tripod in a corner of the room facing the participants. As preparation, each of the team members had spent approximately 30 minutes reading and understanding the requirement specification and the instructions regarding which estimation method to employ and how to employ it. After the preparation phase, the teams started the collaborative work of estimating the software project. This phase of the study was videotaped. When the teams had agreed on an estimate of the project, the team members individually answered a short questionnaire that was collected after the estimation session. The time frame for the two tasks was set at approximately 90 minutes for the bottom-up estimation task and 60 minutes for the top-down estimation task. This difference in time use aimed at reflecting the believed difference in workload between the two estimation approaches.

4.1.1 Description of data

As shown in Table 2, the data material that was gathered through this experimental setup consisted of video recordings and a set of documents. These documents were the requirement specifications, the participant's notes and drawings, answers to a questionnaire and the experimental information. The 17 hours of video recordings that captured the collaborative work done by the seven different teams when estimating the two software projects constitutes the core empirical data in Articles I and II in this thesis. The different documents have been used to contextualise the analysis.

The previously reported results from this estimation study are found in Jørgensen (2004b; 2005). The analytical scheme that Jørgensen used for analysing the video recordings was based on a content analysis and quantification of the time spent on different discussion categories. In addition, quantitative analysis was conducted to measure estimation accuracy. As a part of the work of reporting these results, rough transcripts had been made of the participants' talk. The interactional work of achieving estimates in teams of software professionals was, however, not attended to in the previous reported results from this estimation study. I was allowed to use the existing rough transcripts as the point of departure for my own analytical work. However, they were refined through multiple viewings of the video recordings to fit my level of analysis.

Study	Type of data	Description	Status of data
Quasi-experiment	Video recordings	17 hours in total. Capturing the different teams' discussion when estimating two software project using a bottom-up estimation approach and a top-down estimation approach.	Core data analysed
	Documentation	Requirement specification. Technical description of the software projects that were estimated.	Background and contextualising information
	Documentation	Notes and drawings. The team members' notes and drawings taken during the discussion	Additional secondary data
	Documentation	Form filled out by the different teams with the resulting total estimates together with answers on a short questionnaire.	Additional secondary data
	Experimental instructions	Top-down and bottom-up instructions. Work breakdown structure.	Background and contextualising information

Table 2 Type of data that were collected through the quasi-experimental setup and its status.

4.2 A naturalistic study of the planning poker approach

Article III in the second part of this thesis builds upon a set of data collected as part of a case study of an ongoing large software development project for administrating pensions and loans (PERFORM) during the spring 2010. The aim of this case study is to investigate release planning in practice in a large agile software development project. In short, agile development methodology comprises a set of best practices for delivering software on time that is of high value for the customer (Fowler & Highsmith, 2001). At the core of agile development are customer collaboration and the capacity of responding to changes in the requirements. This is an ongoing case study where the first set of data was collected over a period of four months from December 2009 to March 2010. The data collection was in large part conducted by members of the research project: Planning in large agile software projects (PLASMA) at Simula Research Laboratory, where the aim is to understand and address challenges in planning effectiveness specific to large agile projects.

The following description is based on the information provided in Hannay and Benestad (2010). Three software suppliers are involved in the development of the project that is investigated. One is the in-house team from the customer whose project is developed, and the two other software suppliers are external consultancy companies. A total of 88 developers and 88 business experts participate in the work and they are organised according to Scrum in 11 different teams. Scrum is an "iterative and incremental process for developing any product or managing any work" in agile software development (Sliger & Broderick, 2008, p. 324). To support these 11 Scrum teams, there are one crosscutting architectural team, one crosscutting test team and one crosscutting development environment team. The project has its own premises where all the project members from the different software suppliers are co-located and conduct their daily work.

The actual development of this large software system is planned to last for three years altogether. The system is released incrementally with about three releases per year. Each release is divided into a set of five to six sprints in which each sprint last for about three weeks. The three different software suppliers are contracted on an individual basis to deliver an agreed set of features for a given release. As a part of realising this large project, all the development tasks that need to be completed in the project are specified in a master plan backlog. Figure 3 shows how the different tasks in the master plan are divided into different releases where one release has its own specific release backlog containing the different sub-tasks that are going to be completed in that release. Prior to the start-up of the work on a release, the key stakeholders meet to achieve a joint understanding of the release backlog. During this meeting, tasks are specified, changed and refined. Then the release backlog is split into three different sprint backlogs, one for each subcontractor containing the different sub-tasks that are going to be completed in a sprint. Each of the subcontractors estimates the tasks in their designated sprint backlog using their preferred estimation method.

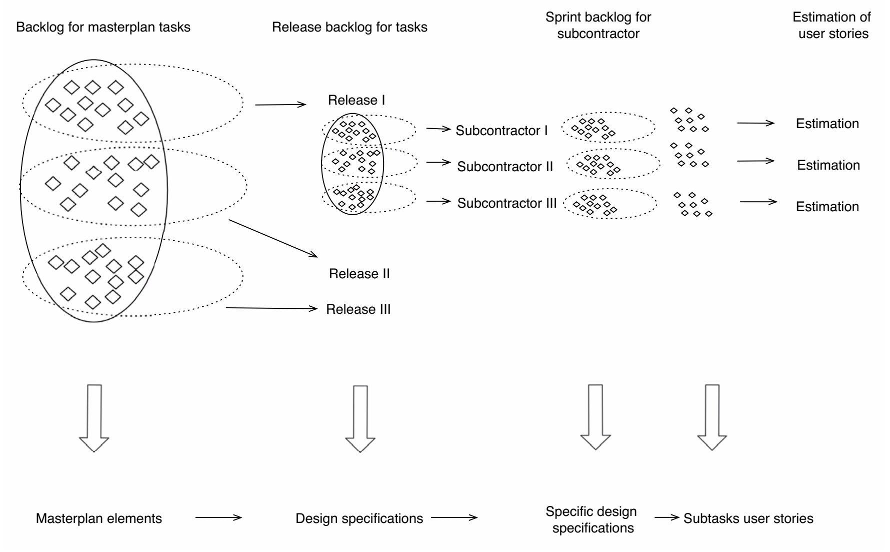


Figure 3 Illustration of how the software project is organised into the sub-tasks that are estimated.

In addition to splitting up the tasks in different types of backlogs, the tasks need to be specified as requirements at various levels of abstraction. These levels of abstraction correspond to the different backlogs of tasks as shown in Figure 3. At the master level, what are named master plan elements have been developed, approximately 300 in total. These master plan elements serve as high-level requirement specifications and are written as user stories, one for each master plan element. A user story is a description written in a natural language of what a user should do in a given situation from the users' own perspective. A typical example from the case reads: "As an agency official, I can reconstruct NAV- and AORD information so that I can see which data was registered at a certain point in time". Each master plan element is also given a priority according to how important or urgent the functionality represented by the element is for the user. Based on the priorities and initial estimates that were collected at project inception, the master plan is divided into three releases.

As part of the release planning, the master plan elements are further specified and developed into design specifications that describe how the master plan elements and the subtasks to be included in the release should be implemented. The work of developing a design specification is a continuous task that goes on over a longer period of time. In the design specification, the requirements are also here formulated as user stories but the descriptions are more detailed than the ones in the master plan. In addition to user stories, the design specification also comprises flow charts and detailed descriptions of information useful for understanding how the task should be implemented as well as how the user interface should be designed. Each subcontractor develops design specifications for the tasks they are going to perform. Prior to the start-up of the development, the number of work hours on the different tasks needs to be estimated. The estimation work of one release was conducted in a series of meetings where representatives from the (scrum) teams that were going to develop the task participated. The number of participants in the estimation meetings range from 3 to 12 persons at the different subcontractors depending on the estimation approach employed. The estimation work of one release was spread out on different meetings that were held during one week in March 2010.

When the estimation work of the release was finished, the total estimate that was achieved through these meetings was presented to the project management and used as a starting point for negotiating a target cost for the release. At this point, adjustments to the estimates can be made. The responsibility for the target cost is shared 50–50 between the project management and the different subcontractors for over/under runs.

4.2.1 Description of data

The data material that has been collected in this case study regarding the estimation work consists of a set of video recordings from three estimation meetings, one at each of the three subcontractors. This was recorded during the week the estimation work took place. At each recorded estimation meeting, a camera was placed on a tripod in one corner so that it captured all the team members as well as what was projected on a whiteboard. The videotaped meetings lasted 2.04, 3.41 and 2.43 hours respectively. To complement the video recordings, I collected the relevant documents that were used in the meetings. These documents were registered in the project-planning tools JIRA and Confluence [5]. The documentation consisted of the master plan elements with the associated design specifications and flow diagrams that were discussed and presented with a projector on a whiteboard during the meetings. Two of the subcontractors used the estimation approach referred to as planning poker, which is studied in depth in Article III in this thesis. The third subcontractor used a variant of the bottom-up estimation approach supported by the company's own model, which was based on historical data from the company. The project management in the PERFORM project approved the use of this material for my research purpose. The following Table 3 provides an overview of the type of data and how it has been used in the analysis.

Study	Type of data	Description	Status of data
Planning poker study	Video recordings	Three estimation meetings. In total 8.5 hours. Two meetings where planning poker was used that amounts to 6 hours and 20 minutes. One meeting where a bottom-up estimation approach was used.	Core data
	Design specification	More-detailed specification of user stories comprising description of how the user stories should be developed and flow diagrams.	Background and contextualising information
	Master plan elements	High-level requirement specifications formulated as user stories.	Additional, secondary data
	Planning poker cards	A special deck of cards that the teams used when estimating the effort.	Background and contextualising information

Table 3 The type of data collected through the case study and its status.

4.3 Methodological considerations

To examine the collaborative activity of software effort estimation, interaction analysis is used as a means to investigate the communicative and interactional work

[5] For more information about the project planning tool JIRA and Confluence, see the developer site where the software can be purchased http://www.atlassian.com/software/jira/

that software professionals do in depth. The data materials that are used as an empirical basis for addressing these issues are, as explained above, from two quite different settings, namely a quasi-experimental setup and a real-life estimation meeting taking place as part of the work of developing a large software system.

The dataset from the quasi-experimental setup was analysed in depth in Articles I and II in the second part of this thesis. Using data from a quasi-experimental setup is not unproblematic in a sociocultural tradition. However, the experimental setup was regarded as being very close to a real-life estimation situation. First of all, real requirement specifications were used from customers who had hired the specific company to develop the systems that were estimated. Secondly, software professionals with sufficient estimation competence from the company were hired as participants. Thirdly, the information available for the participants when conducting the estimation work was similar to the information the participants would normally have when estimating software projects. In addition, Jørgensen (2004b) has also stated that there was a striving for realism in the design of the study and that the realism was assessed as high.

The second data set, which is analysed in depth in Article III, was collected from authentic real-life estimation meetings. After repeated viewings of the data material, it became apparent that even though the settings in which the data material were gathered were different, the situations did not differentiate significantly between the two studies. Rather, similar patterns were discovered across both data sets. Further, three different approaches to the conducting of estimation work have been investigated and also here similar patterns were discovered across the different approaches.

Even though the two studies have been designed for other purposes than investigating the interactional work that teams of software professionals do when estimating software development projects, the video recordings provided opportunities for focused examinations of teamwork in software estimation at a later point. The recording from the estimation study made it possible to investigate the kind of work that was conducted in teams when the two most common estimation approaches were employed. This opportunity to compare models of estimation approaches with what actually goes on when the work is conducted provides valuable information in regards to how the estimation practice can be supported and improved.

The video recordings from the case study provided data from real-life estimation meetings. Hence, it was a unique opportunity to study the estimation practice in teams. Also, this second data set provided opportunities to confirm and strengthen the results reported in the first two articles. Although the estimation work that is conducted in the two different studies is organised according to different development traditions, software effort estimation as a task needs to be done regardless of how the software development work is organised. As my interest is to study effort estimation as a phenomenon, it is not a problem to look at studies that are situated in different software development traditions.

The increased use of teamwork in estimation (Haugen, 2007) calls for research that takes teamwork processes under scrutiny and examines them in order to reveal critical instances that may be of help to understand, support and improve the estimation practice. The empirical material available for analysis was therefore considered as suitable for investigating the collaborative work of software effort estimation and to address the research questions that were raised in the introduction, section 1.2.

4.4 Analytical procedures

Investigating the collaborative work of software effort estimation requires analytical procedures that open up the communicative and interactional work of estimating a software development project so it can be investigated in depth. The methodological approach chosen as a means to do this has been interaction analysis.

4.4.1 Interaction analysis

As a method to study video recordings, interaction analysis (Jordan & Henderson, 1995) builds upon different methodological approaches, which have in common that they approach talk and interaction. Among others these are ethnomethodology (Garfinkel, 1984), conversation analysis (Silverman, 1998) and discourse analysis (Wetherell, Taylor, & Yates, 2001). In their now almost classical article Jordan and Henderson describe interaction analysis as:

> an interdisciplinary method for empirical investigation of the interaction of human beings with each other and with objects in their environment. It investigates human activities such as talk, nonverbal interaction, and the use of artefacts and technologies, identifying routine practices and problems and the resources for their solution (Jordan & Henderson, 1995, p. 39)

What is special about interaction analysis is the way in which Jordan and Henderson emphasise that it is not only concerned with verbal talk but also takes into consideration the participants use of artefacts as an important part of the analysis. Interaction analysis rests on a sociocultural notion and

> finds its basic data for theorizing about knowledge and practice not in the traces of cranial activity (for example, protocol or survey interview data), but in the details of social interactions in time and space, and particularly in the naturally occurring everyday interactions among members of communities of practice (Jordan & Henderson, 1995, p. 41)

The work by Jordan and Henderson has been taken further by Derry et al. (2010), who addresses four specific challenges that are of importance when analysing video recordings in depth. The four challenges that they discuss concern selection, analysis, technology and ethics. In the following, I will explicate how I have addressed issues concerned with the selection and segmentation of data and in addition, how I have approached the analytical work.

The use of video recordings makes it possible "to analyze 'situated' action; as it emerges within its ordinary ecologies" (Heath & Hindmarsh, 2002, p. 103). Hindmarsh and Heath (2007) have shown how video-based studies are important for understanding different types of work practices. Video-recorded data provide a unique possibility for investigating details in interactional work. The reason for this is because the data can be viewed multiple times and at different times during the research project. Further, it can be viewed with different people. This can be quite useful when analysing the data because a particular segment of the video can be discussed and analysed jointly to gather different interpretations, which strengthen the validity of the results. In addition, video data captures what people are doing and how they are using different artefacts, which are important issues for in-depth investigations (Derry et al., 2010). In the three video-based studies that are a part of this thesis, interaction analysis has been employed to investigate the work that teams of software professionals do when estimating a software project. Interaction analysis can be performed in different ways according to the purpose. In this thesis, the analysis that has been conducted follows the content dimension and the moment-to-moment interactions in the different teams to reveal the different problems the teams engage with in their collaborative work of achieving an estimate.

In order to explore the collaborative work of software effort estimation, the details of the interactional work of the different teams needed to be opened up and made accessible for analysis. To do this, a set of intermediate concepts has been used as sensitising means. These concepts vary to some extent between the different articles as different topics have been investigated. In addition, two types of concepts are used, empirically driven ones and theoretically driven ones. The empirical concepts are close to the empirical material and are used to capture the kind of work that the teams are conducting. The theoretically driven concepts are used analytically in the analysis. These two types of concepts make it possible to focus on both the content and the interaction that unfolds during the teams' work process. Thus they make it possible to look beyond what is obvious (Lindwall, 2008).

In Article I, the aim was to investigate the kind of work software professionals do when estimating. The intermediate concepts used were orientation, elaboration, clarification and positioning. Furberg and Ludvigsen (2008) and Rasmussen (2005) originally developed these concepts as ways of opening up the details in interactional work. In the analysis in Article I, these concepts have been adapted and fine-tuned so that they fit the specific purpose of the analysis conducted. Further, they are empirically driven, which make them suitable for explicating interactional accomplishments.

In Article II, the aim was to identify and investigate the kind of challenges that software professionals face in using historical data when solving present working tasks. The intermediate concepts that were used to open up the details of the interactional work for analysis were recontextualisation, elaboration and clarification. Again, the concepts of elaboration and clarification were adapted to fit the specific analysis conducted in Article II. In addition to these two empirical concepts, recontextualisation was used as an analytical term. This concept has been derived from theory, as described in Chapter 3. However, in the analysis of Article II, the concept is used as an analytical term to capture the ways in which the teams are able to adapt and make meaning of past experiences and knowledge.

In Article III, the use of concepts in professional work is investigated with the aim to achieve an understanding of how software professionals invoke different types of knowledge when reasoning and reaching a decision on an effort estimate of a software development project. The intermediate concepts that were used in this article were elaboration, clarification, justification and specification, which are all empirical. Also, here the concepts of elaboration and clarification were adapted to fit this specific analysis. In addition, the two concepts of specification and justification were used to grasp how a decision was reached in the teams' collaborative work.

4.4.2 Selection and preparing for analysing interactional data

Video-based studies usually generate lots of data. This poses a challenge for the researcher in terms of data selection. It is not possible to investigate all the collected data in depth. The video-recorded data material that forms the empirical basis of this thesis amounts to a total of 25.5 hours. Analysing all of this data in depth is too time consuming for a PhD project, thus, a selection needs to be made. Selecting particular segments of data that would at a later time be analysed in depth has been done in accordance with the research questions of interest and the theoretical framework that is used (Derry et al., 2010). The selection of data material that I have done coordinates with the main topic of the three articles that have been written. However, certain precautions were made when the data set was selected.

In Article I, the bottom-up approach was investigated in depth. First of all, the data material were narrowed down to include only the team discussions concerning the bottom-up approach because we wanted to investigate the kind of work software professionals do when estimating a software project. The bottom-up approach is the most commonly used judgement-based estimation approach amongst software professionals. Also, a common problem in software effort estimation is over-optimistic estimates, and thus the data material was further reduced to a team that displayed this problem. Making the selections based on these criteria was possible because of the quantitative data analysis that Jørgensen (2004b) had conducted and reported on earlier.

In Article II, the challenges software professionals face when utilising historical information in present working tasks was investigated. First of all, I narrowed the data down to include only the team discussions in which a top-down approach was employed. Before selecting teams, I did an overall analysis of all the teams' interactional processes, which showed a large variety in the teamwork that was conducted. To display the large variety, two teams were selected that solved the task in quite different ways, to ensure a solid dataset. Further, I ensured that the teams that were chosen had conducted the estimation tasks in the same order since the data was from an experimental setup. The first task was estimated by employing the bottom-up approach and the second task was estimated employing the top-down approach. Another careful decision was regarding the estimation accuracy of the selected teams. Again, it was the analysis already conducted and reported by Jørgensen (2004b) that made this possible. In addition, continuity in the empirical data was sought so that one of the teams that was selected in Article II was the same team that was followed in Article I.

In Article III, we investigated the use of concepts in professional work, with the aim of understanding how software professionals invoked different types of knowledge when reasoning and reaching a decision on an effort estimate. In this article, the data was first narrowed down to the two teams that employed the planning poker approach when estimating. The choice of investigating the planning poker approach was because this made it possible to investigate the negotiations that follow a round of planning poker in which team members need to justify and explain their individual estimate to the group. Thereafter, we decided to analyse in depth the interactional work conducted by one of the teams to get a closer look at the collaborative problem solving in this work. The selection of teams was based on the richness of the interaction. One of the teams that employed the planning poker approach was the in-house team to the customer. This team therefore knew the project to be developed hands on and their discussions therefore reflected that a lot was taken for granted as common knowledge amongst the team members who were working on the project.

Hence, we chose to follow the team from the external subcontractor that used planning poker as estimation approach.

The richness of the interaction in the teams was important to ensure a solid dataset for the in-depth analysis. It thereby formed a general selection criterion across the three articles. Further, across Articles I and II the quantitative analysis of the entire corpus of data conducted by Jørgensen (2004b) also provided important information for selecting which teams to follow.

After selecting the different teams that would be subject for in-depth analysis, transcripts of the teams' interactions were written. Transcripts are an important way of preparing the data for analysis. The level of details included in a transcript varies across research traditions and can be done in several ways. As the analysis focuses on the moment-to-moment interaction and the content, the level of details and elaboration corresponds to these analytical interests (Jordan & Henderson, 1995).

In the data material from the experimental setup rough transcripts of the different teams' interactions had already been made and I was allowed to use them as a point of departure for my own work. In the naturalistic study, however, rough transcripts were written after choosing which team to follow in depth. A master student was employed to conduct the work of transcribing the team's interactions roughly. These rough transcriptions were used to "flesh out", as Derry et al. (2010) puts it, which segments of data were important to pursue in-depth analysis of. Thereafter, I wrote more-detailed transcriptions of the different segments that were selected.

4.4.3 Segmentation and presentation of data

An empirical sensitive analysis was first conducted of the different teams' interactional processes. This was done through repeated viewings of the video recordings together with repeated readings of the rough transcripts. Through this empirically sensitive analysis, I formed an understanding of what kind of work the teams conducted, and specific patterns emerged in the data that allowed me to select extracts. The segmentation of extracts in the different articles was done in slightly different ways.

In Article I, extracts were chosen that followed the sequential order of events with an aim to demonstrate the kind of work that was conducted to achieve an estimate. Hence, the analysis followed the content dimension of the different events that occurred along a timeline. In Article II the interactional process of two teams was investigated. Through the initial empirically sensitive analysis, I was able to indentify the key challenges that the software professionals faced when using historical information in their work. In this case, the selection of excerpts were an event sampling of the key challenges; instead of following a sequential order of events, the analysis is presented thematically to demonstrate the different ways of overcoming the challenges that were faced. In Article III, the collaborative work of one team was followed. The video recordings were used as an ethnographic frame to give an overview of the material and the setting for estimation. After multiple viewings of particular episodes a few episodes that demonstrated typical actions and sequences of talk were chosen. These episodes were selected based on what seemed to be important to the data and the activity itself, which in these data was the goal of reaching consensus. Further, it was also a theoretical sampling based on how estimation practice is conceptualised in software development.

After choosing particular segments that were interesting to pursue for in-depth analysis. I wrote up detailed transcriptions, ensuring that both talk and important actions that were performed were included. To display the transcripts, I have followed what Jordan and Henderson (1995) called "parallel columnar transcripts". Here, it is possible to include both verbal talk and also different activities that happened in side-by-side columns. Three different columns were used in Articles I and II and two columns in Article III. The first column included the numbered turns in the interaction together with initials indicating who did the speaking. The second column included the verbal communication between the participants placed horizontally in line with the speaker in column 1. The third column was included in the transcripts in Article I and II and consisted of the description of actions. These were placed horizontally in line with the utterances that were spoken when the action took place. In Article III, the description of relevant actions was included in the running interactions. After I finished making the detailed transcripts for the different extracts, I began the analytic work.

In conducting the analytic work, I read the different extracts multiple times and also discussed them with supervisors and colleagues in sociocultural research groups at both the University of Oslo and the University of Gothenburg. Typically, several more extracts were analysed than what was included and displayed in the different articles. All the analyses were conducted on transcripts in Norwegian—hence, the transcripts were translated into English for the purpose of communicating the research in international articles. Doing this ensured that important information was not lost for analytical purposes in the translation (Temple & Young, 2004). Further, a professional translator with a background in software development was hired to translate the transcripts into English.

4.5 Doing credible research

Doing credible research is just as important when using qualitative data as it is when using quantitative data. However, the way credibility is reached is slightly different due to the difference of the data material. While quantitative research has established forms and ways of doing this such as retesting and sampling data, qualitative research has other ways of achieving credibility. In the following, I will explain what I have done to ensure the credibility of the research that is a part of this thesis. I will start by explaining how reliability is achieved before I continue with discussing the validity of the research and issues concerning generalisation.

4.5.1 Reliability

Reliability in qualitative research concerns the quality of the data material that has been gathered to investigate a particular phenomenon. In my research project, the phenomenon investigated is collaborative work in software effort estimation. To investigate this phenomenon, I have used two different types of data, in which three different estimation approaches were employed. The first data set that was used in Articles I and II was video recordings gathered through an experimental setup in which teams of software professionals estimated real software development projects. The second set of data, used in Article III, was video recordings of real-life software estimation meetings where teams of software professionals estimated parts of a specific release of a software development project. Even though these two data sets were gathered from two different settings, one experimental and one naturalistic, the data displayed the same type of characteristics concerning how the teamwork was conducted. For example, the extensive sense-making processes that were necessary to achieve a shared understanding of the requirement specification, i.e., estimation task, were found in both data sets. Also, the need to elaborate beyond the information given in the different requirement specifications was evident across the data sets. That the same characteristics were identified in both the data from the experimental setup and the data from the naturalistic setting strengthens the reliability of the present research.

Whether tape recordings of natural occurring talk might influence the participant's interactions in some way is an issue that has been much debated. Will the presence of a recording device inhibit the interaction, or affect what occurs in the estimation work? Jordan and Henderson point to the fact that participants get accustomed to the presence of a camera quite quickly: "In the long run, and in particular as people become involved in tasks other than worrying about the camera, camera effects visibly wear off" (Jordan & Henderson, 1995, p. 56). The research presented in this thesis focuses on how the participants reason to reach a decision on an effort estimate in their collaborative problem solving and not their actions in other respects. The actual estimation work was therefore not particularly affected by the presence of a camera. In the two data sets used as an empirical basis for this thesis, orientation towards the recording devices occurred just a few times. In the experimental setup a joke was made concerning the use of the recordings, that it was a good thing that the recordings were not for management purposes. In the naturalistic study, in particular, in one episode, orientation towards the camera was obvious. This occurred when a new participant entered the meeting room a while after the meeting had started, and another participant gestured that he should move to the left a little bit so as to not block the camera view. However, it is difficult to be certain that the participants' collaborative work was not influenced at all. One way of assessing the potential effects of the camera's presence is to analyse the interactions in which the participants orients towards the fact that their talk is being recorded (Speer & Hutchby, 2003). This kind of assessment has not been done in a systematic way on the video-recorded material included in this thesis; however, this does not affect the quality of the data in relation to the phenomenon that is being investigated.

The two sets of video recordings were analysed by means of interaction analysis. The ways in which the video recordings are described and transcribed are an important part of increasing reliability. My efforts regarding this issue have been to make the data and research process as transparent as possible. Therefore, I provided thorough descriptions of the context in which the two different data sets have been gathered. In addition, I have provided a thorough description of the way the data material have been segmented and the selection criteria of choosing particular episodes for an analysis in depth. This thesis is article based, which means that the research has been written up as scientific articles. The journals to which these articles have been submitted to or published in usually have a word limit, thus it is often the empirical description that suffers on this account. Therefore, a detailed description of the empirical context and the segmentation of data are included in this extended abstract.

Further, the transcripts have been written and presented in such a way that they are easy to follow, in terms of both readability and of following the analysis that has been conducted of the different excerpts. When employing interaction analysis, the level of detail that is included in the transcripts is adapted to the analytical interest of the researcher. As the level of analysis and the focus of the research within this thesis are not on a detailed linguistic level, but focus on the content dimension and the interactional work, I have chosen to include a set of conventions that I perceive as relevant to the level of analysis that is employed in my research. In the following Table 4, the different transcription symbols that have been used throughout the transcripts in the three articles are presented.

Notation	Meaning
[ ]	Indicating overlapping talk between participants
(x,x)	Indicating a pause in the interaction timed in seconds
…	Indicating a short break in a participants utterance
< >	Comments on actions
( )	Indicating that it was not possible to interpret what was said
. , ?	Punctuation, commas and question marks have been used to ease the reading of the transcript

Table 4 Transcript notations used in the three articles.

The way in which the transcripts are displayed in the three articles makes it possible for readers to follow the analysis step by step and also to read and follow the interaction as it unfolded. In addition, what the participants did and how they orient towards the material artefacts in their work process could be followed, as descriptions of the participants' actions were included in a separate column in Articles I and II. Thus, it provides the reader with a possibility to form his or her own opinion of the analytic work that has been conducted and if the interpretations that are made are reliable.

As a last point, I want to emphasise that the analysis was conducted on the Norwegian transcriptions, thus the transcriptions were not translated into English before the analyses were complete. This was done to avoid losing important information in the translation, which can be problematic due to language differences (Temple & Young, 2004).

4.5.2 Validity

The issue of validity is concerned with whether the claims that are made about the phenomenon that is studied can be regarded as solid and thus fits the phenomena that it refers to (Silverman, 2006). Hence, validity is concerned with how the data material is interpreted and the premises that the interpretations are built upon. My main aim with the research that has been conducted as part of this thesis was to investigate the collaborative work in software effort estimation. In the following, I will argue that the video recordings of estimation teams, where software professionals worked together to achieve an effort estimate of a software development project using three different estimation approaches, served the purpose of answering the research questions outlined in this thesis.

This extended abstract describes in detail the context and setting of the study, the participants and also how I have used interaction analysis as a means to investigate the video recordings. The intention with providing such a thorough description is to make it easier for the reader to follow how the analysis has unfolded as well as assess the claims that are being made about the phenomenon investigated. It is thereby a means to strengthen the validity of the research that has been conducted (Creswell & Miller, 2000).

In the process of writing up the analysis, resources from two different research communities have been drawn upon in terms of theory, methods and domain knowledge. My point of departure when starting the work on the research presented in this thesis was within the domain of software effort estimation. Given that this thesis is interdisciplinary, knowledge and understanding of the domain software effort estimation as well as of the theory and methods employed were needed. To obtain this, supervisors from different fields were recruited that have the competence necessary for studying estimation and employ the chosen theoretical perspective and methods. Further, the theoretical perspective and methods that have been used were suitable for studying the collaborative work of software effort estimation. Establishing this close connection to two communities of research, by having supervisors representing both, was an important part of the work to ensure that the interpretations and analysis conducted were regarded as valid across both research communities. This also strengthens the validity of the research that has been conducted as part of this thesis.

The different excerpts and initial analysis have also been presented and discussed with colleagues in sociocultural research groups at both the University of Oslo and the University of Gothenburg. The critical comments and also the collaborative efforts of analysing jointly the excerpts strengthen the validity of the analysis in the three articles that have been conducted, because the quality has been controlled on different stages of the work process (Kvale & Brinkmann, 2009). Further, in the different articles that constitute Part II of this thesis, all the transcripts that have been analysed in depth have been included in the running text. Hence, possibilities for readers to form their own alternative interpretations are made available.

One validity issue that has been highlighted in the study of social interactions is validation through the next turn (Peräkylä, 2011). In interactions, the utterances produced are interpreted by the other participants, which is usually displayed in the next set of actions. By including the transcripts of the analysed episodes in the different articles, it is made transparent if the interpretation and analysis of the utterances are in accordance with how the participants treat the utterances in the next turn or not. Validation through the next turn also relates to what Maxwell (2002) named interpretative validity, where the interpretative validity is concerned with the inferences made from actions made by the participants in the context they are studying. Such interpretative validity has been ensured by drawing on resources from the two research communities that have been involved in this work through supervision. The articles have also been submitted to scientific journals and thus have been or are in the process of being peer reviewed as part of the process of being accepted for publication. Accordingly, I will argue that this also strengthens the validity of the claims that are being made in the analysis.

4.5.3 Generalisation

In qualitative research, the type of generalisations that are made are called analytical. Analytical generalisation is based upon an analysis of similarities and differences between two situations. In order to analyse similarities and differences, thick descriptions of both the context in which the study was conducted and the context to which one wishes to generalise are necessary (Schofield, 2002). Schofield proposes three targets of generalisation: to what is, what may be and what could be (Schofield, 2002, p. 80). Generalising to "what is" concerns findings about what is typical, general or common in a situation. When aiming at generalising to "what may be", issues for the future is the main concern, in terms of what are the current trends and what is likely to happen in the future. The third target for generalisation is "what could be". In this context one focus on situations that are ideal or exceptional in some ways and investigate them to see what is actually taking place here. Another concept that has been raised concerning generalisation issues is "possibility". "Generalising to social practices that are possible, i.e., possibilities of language use, are the central objects of all conversation analytic case studies on interaction in particular institutional settings" (Peräkylä, 2011, p. 375). Given these different ways of generalising in qualitative research, Derry et al. (2010) emphasise the importance of explicating the logic of inquiry that is used in the study. This is important to be able to meet criticism about the generalisability of findings. It therefore includes explicating the approach that was used for selecting episodes and recordings and the processes for which explanations and claims have been generated.

The research within this thesis has studied different approaches to estimation work independent of each other and under different conditions. As part of increasing the reliability and validity of the research, I have provided thorough descriptions of the context in which the studies were conducted, the methods used and the analytical procedures for analysing the video recordings. Thus, the logic of inquiry that Derry et al. (2010) emphasise as important has been thoroughly explicated. Moreover, to ensure valid interpretations, resources from two different research communities have been drawn upon in terms of theory and domain knowledge. The way in which reliability and validity issues have been attended to and ensured in this research has made it possible to make some generalisations regarding general characteristics across the three estimation approaches that have been investigated. The most evident one was the need for sense-making processes. Across the analyses, it was found that the different teams studied had to go through sense-making processes in order to make sense of and achieve an understanding of the estimation task. This is a finding that has both theoretical and empirical support across the estimation approaches investigated and across the teams that were investigated. Thus, it can be considered a general characteristic in this type of collaborative work and can be considered to be of the "what is" kind of generalisation using Schofield's (2002) terms. Further, the analytical generalisations that are made in this thesis are raised as more general claims in the concluding section of each of the three articles, in which generalisations are based on relating the findings to the theoretical point of departure and the review of related research.

5 Summary of the articles

5.1 Article I

Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation work in a multi-specialist team. *Scandinavian Journal of Information Systems*, 22(2), p. 65–98

This article examines how a team of software professionals goes about estimating the effort of a software project using a judgement-based bottom-up estimation approach. Providing realistic software effort estimates have been shown to be a huge challenge in the software industry. It has been reported that as much as 70–80% of software development projects overrun their estimates (Moløkken-Østvold & Jørgensen, 2003). Software effort estimation is also an important part of software development as it is used for purposes such as budgeting, project planning and control (Boehm et al., 2000; Grimstad, 2006). Previous research on judgement-based software effort estimation has its roots in the cognitive tradition of empirical research and has mainly been concerned with investigating the individual reasoning when expert judgements are made.

Estimation work is increasingly carried out as teamwork in the industry (Haugen, 2007). In spite of this the research on judgement-based estimation processes in groups seems to be limited. Studies of collaborative work have, however, been conducted in other areas of software development. For example have cooperative work in software testing been researched (Martin et al., 2008), how collaborative processes are organised in distributed software teams have been investigated (Boden & Avram, 2009) and the way planning are recurrent process in software development have been studied (Rönkkö et al., 2005). The use of qualitative methods in software engineering research is scarce (Dittrich et al., 2007). Thus, few studies provide a minute analysis of planning activities as such. This article examines the details of how software professionals identify, explore and negotiate the issues at stake in the different estimation steps and how they make further collective decisions.

By employing a social practice perspective that highlights the distributed character of expertise and perceives actions as mediated by cultural tools, this article analyses the interactional process through which the estimation tasks were collectively explored, negotiated, and accomplished. An analytical focus on the communicative and collaborative aspects of software effort estimation has not been employed in estimation research before. The analysis of the team's interaction was guided by the following research questions:

– *What characterises software effort estimation as a collaborative activity in a multi-specialist team?*
– *What types of communicative and explorative work are needed to accomplish the planning task and agree on an estimate?*

The data that constitute the empirical basis in this article consisted of video recordings that were gathered as part of a large software effort estimation study in the Norwegian branch of an international IT-consultancy company in 2002. The design was quasi-experimental in character in the sense that seven estimation teams were organised and asked to estimate two legitimate software projects using two different estimation approaches known as bottom-up and top-down. The data material was narrowed down to include in-depth analysis of one team that employed the bottom-up estimation approach. The analytic focus is on what kinds of problems the team needs to explore and what they collectively achieve by the actions taken. The video recordings were analysed by means of interaction analysis.

The findings show how software effort estimation is carried out through a complex series of explorative and sense-making actions, rather than by applying assumed information or routines. The quantification of the number of work hours is accomplished through extensive elaboration and clarification in which expertise is mobilised and coordinated through social interaction. Interpreting the requirement specification posited a challenge for the team and the sense-making processes called for quite thorough technical elaborations because the information provided in the requirement specification was ambiguous and thus not possible to use in a direct manner. The team members had to move back and forth between technical elaborations and integrative windups as new information or ways of understanding emerged. The sense-making processes thus alternated between addressing the planning aspect of the estimation practice—that is, questions related to how the software project could be developed and organised—and the problem-solving aspect of the practice, reflected in issues and questions that needed further elaboration and clarification. The conducted planning work is both imaginative and future oriented and thus it does not rest upon existing skills and routines, but is accomplished through a series of tool-mediated explorative and sense-making actions. In this work process, the requirement specification serves both as a recurrent object for exploration and as a tool for collaborative problem solving. Thus the requirement specification served several mediating functions in the interactional process, through which expertise was mobilised and coordinated. The article argues that to grasp the complexity of software estimation, there is a need for more research that accounts for the communicative and interactional dimensions of this activity and that existing research needs to be supplemented and expanded to incorporate methodological approaches that are sensitive to the communicative and interactional dimensions of this activity. Moreover, by revealing the interactional details of a planning activity, the article contributes to our understanding of the future-oriented and constructive dimensions of social practices.

5.2 Article II

Børte, K. (submitted). Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation.

Making use of historical information from databases and previous experiences when solving present working tasks is a complex issue. Numerous workplaces develop databases and archives to make historical information accessible; however, many also experience difficulties in utilising such information in productive ways. In this respect, software effort estimation is an interesting case as it is often organised as collaborative work. Moreover, specific estimation approaches request the use of historical information as a way of achieving an estimate. Software effort estimation is considered an important aspect of software development work (Kjærgaard, Nielsen, & Kautz, 2010), as it is used for planning, controlling and budgeting software projects (Boehm et al., 2000). This article examines the collaborative work of how teams of software professionals go about to use historical information, and what challenges they face, when applying a specific estimation approach called analogy-based, top-down estimation. An analogy-based, top-down estimation approach is an idealised model of how to achieve an estimate by using historical data from previously completed software projects, which are identified by analogies. Researchers have argued that this estimation approach is more efficient, less time consuming and therefore cheaper to apply than a bottom-up approach (Boehm, 1984). The argument is also made that it can be applied without much knowledge of how to build software, and that it may reduce the bias towards over-optimism, due to the greater use of historical data from previous projects (Moløkken-Østvold & Jørgensen, 2005). The use of analogies in software effort estimation has been investigated in previous research, which have revealed that there are difficulties connected both to identifying and using historical data. However, few studies have looked into what causes these difficulties and how they are dealt with.

By employing a sociocultural perspective on knowledge and communication, this article analyses the collaborative process through which two teams of software professionals achieve an effort estimate using an analogy-based, top-down estimation approach.

The following research questions are addressed:

– *What challenges do teams of software professionals face when applying an analogy-based top-down estimation approach and how do they occur?*
– *What kind of work is needed to utilise knowledge from former software projects when estimating new ones?*

The empirical basis for the analysis in this article was videotaped data from a large quasi-experimental software estimation study conducted in the Norwegian branch of an IT-consultancy company in 2002. Seven estimation teams estimated two real software projects employing two different estimation approaches known as top-down and bottom-up. The data material was narrowed down to include an in-depth analysis of two teams that employed the top-down estimation approach. The analytical focus is on explicating the challenges encountered by the teams when setting out to use historical information when solving present working tasks, how they occurred and how they were dealt with in the teams' collaborative work process. The video recordings were analysed by means of interaction analysis.

The findings revealed three concrete challenges related to the use of historical information in the teams collaborative work: 1) finding similar completed projects, 2) exploring and negotiating comparable dimensions and 3) making comparisons to quantify the number of work hours. To overcome these challenges, the teams engaged in collective remembering and recontextualisation processes and constructed boundary concepts. Through collective remembering, knowledge and experiences were shared and relevant historical information was jointly remembered and explored in detail to achieve a shared understanding for conducting the work. Achieving such shared understanding was important and both the project to be estimated and potential relevant historical information were investigated in detail through communicative work. The findings revealed how the teams performed a double sense-making process to achieve this and to find possible connection points between the present estimation task and historical information. Further, the teams performed extensive recontextualisation processes of the historical information to adapt it so that it could be utilised in the new task. In these processes, meaning potentials were realised and meanings were generated, articulated and reinterpreted in the teams' communicative work by way of elaboration and clarification. When possible connection points were found between the project to be estimated and historical information, the teams constructed boundary concepts that could facilitate the use of knowledge across contexts.

The article concludes by arguing that the idea of analogy-based, top-down estimation does not take sufficiently into account that knowledge needs to be recontextualised to become meaningful in new situations. The analogy-based, top-down estimation approach rests upon the idea that mapping knowledge between contexts is possible; however, the analysis in this article revealed the importance of sharing, exploring and adapting knowledge, experiences and understandings through communicative work to be able to utilise historical information across contexts. This implies that communicative work is necessary for utilising historical information across contexts and that problem solving is grounded in the communicative work conducted through elaborations, specification and negotiations between team members. This article therefore contributes to the research field of software effort estimation in that it provides an understanding of the communicative work required for using historical information when solving present working tasks. However, to understand and facilitate these processes in estimation work, more research is required to account for the communicative and collaborative dimensions of this activity.

5.3 Article III

Børte, K., Ludvigsen, S., & Mørch, A. (submitted). The role of concepts in expert work: Unpacking ‘the magic step’ in software effort estimation.

This article examines the use of concepts in professional work by analysing the specialised work of software effort estimation. The aim is to achieve an understanding of how software professionals invoke different types of knowledge when reasoning and reaching a decision on an effort estimate of a software development project. When engaging in problem solving, professionals draw on resources that originate in research and developmental work in a particular knowledge domain together with experience-based knowledge. Through education and training, professionals acquire a professional language and artefacts that they use to solve work-related problems. Language consists of concepts that do not in general have any set of fixed meanings, but rather they need to be understood in relation to what people are trying to achieve in collaborative activities.

Many studies of expert judgement-based estimation use the individual as the unit of analysis and the social aspects are rarely taken into account. In estimation work, the step from reasoning to deciding on an effort estimate has been referred to as “the magic step” in software effort estimation (Jørgensen, 2005). This is because software professionals have not been able to explain how they reached the decision on a particular number of work hours. However, new estimation techniques, such as planning poker (Grenning, 2002), which facilitate social interaction to a larger extent, have been developed. In this article, the planning poker technique is investigated in depth to study the use of concepts in the work of professionals as well as how explanations are produced. The planning poker technique provides a good case for studying this as it facilitates social interaction during the construction of an estimate by asking for justification when participants present different estimates. It is in social interaction that the individual participant’s knowledge gets connected to the concepts and artefacts that are part of the collective resources needed by the participants to activate and reason with when achieving an estimate.

We propose that by taking a socio-genetic perspective on concepts in activities, which allow for a focus of three interrelated levels of understanding—institutional practice, individual knowledge and dialogue and activity—the ways in which software professionals reach a decision can be unpacked. The following research questions raised in this article are:

- *Which types of concepts create the direction of the participants' talk and guide their decision-making process?*
- *What kinds of knowledge are invoked in the work of achieving an effort estimate, and where does this knowledge come from?*

To investigate these questions, the empirical focus is on how a team of software professionals creates estimates of the work effort needed to develop new components of a software system using the planning poker estimation technique. The data material that constitutes the empirical basis for the investigations are video recordings of real estimation meetings of a large ongoing software (re)development project governing public pensions and loans. In the analysis of data we used principles from interaction analysis, meaning that we used the ethnographic video-data to frame and analyse the selected episodes, considering them to be demonstrations of sense-making and participation structures. The interactional achievements were interpreted through the use of the analytical categories of clarification, elaboration, justification and specification.

The results from the empirical analysis showed that the user story mediates between the historical practices on the one hand and between the use of generic and specific knowledge on the other, to make assumptions clear and specify what the problem is about. The user story as mediated resource makes it possible for the participants to invoke the different types of knowledge involved. The analysis revealed how concepts from the knowledge domain are used as framing elements for how the task should be approached and accomplished. Through the communicative work, the participants reach a level of intersubjective understanding in which they share enough information that serves as a common ground for taking the next step in the problem-solving activity. In the talk amongst the participants, concepts are not used in a systematic way. Domain-specific concepts are used to frame the task but it is historical experience that seems to be the dominating orientation when it comes to specifying the work needed to solve the task. The domain-specific concepts seem to be invoked only when there is a problem that needs to be framed or reframed.

The phenomenon is interpreted at the intersection between the institutional practice, the individual knowledge and the dialogue and activity. The analysis shows that the participants bring systematic knowledge from the domain of information systems and personal experiences, when negotiating about and framing how the task should be solved. Moreover, the interaction oscillated between the institutional level and the individual level through the level of social interaction. At the institutional level, reaching a consensus is achieved through justification by connecting the social aspects and the use of different types of technical knowledge. The different analytical concepts explicate how this work is done at the level of social interaction, by processes of narrowing down and achieving a shared, satisficing understanding and decision making on an estimate, whereas the domain-specific knowledge serves to frame the task.

The article reformulates the notion of the magic step proposed by (Jørgensen, 2005) and concludes by arguing that the magic step is unpacked in the analysis of the social interaction. In social interaction, the concepts used are anchored in the knowledge domain of software engineering and in the historical experiences of the participants and subsequently become activated. Moreover, the step from reasoning about the task to deciding on an estimate is not perceived as magical, but a step that is categorically different and thus differentiating two conceptual schemas. The theoretical assumptions used as points of departure for investigating a specific phenomena matters in terms of understanding a phenomenon in different ways. In our view, it is through detailed multi-level analysis of how estimation practice is framed within larger social and knowledge structures that we can improve practical procedures for estimation work.

6 Contribution and conclusion

The research within this thesis focuses on the social interaction of software effort estimation work and seeks to understand it as collaborative planning activity. Through the review of relevant research within the field of software effort estimation that was presented in Chapter 2, a focus on the social aspects of estimation work was justified. To investigate this, an appropriate theoretical perspective and related methods are needed. In this extended abstract, I have provided a conceptual framework for how the collaborative work in software effort estimation can be understood and explicated. In addition, I have provided a thorough description of the methods and analytical procedures that have been employed in the research presented in this thesis and summarised the results of the three articles that have been written as part of this work. The three articles have together investigated and provided answers to the following three sub-research questions:

i) *What characterises software effort estimation as interactional accomplishments?*
ii) *What kinds of challenges do estimation teams face when utilising historical information in present working tasks?*
iii) *In what ways are estimation work mediated by cultural tools?*

In the following, I will synthesise the research results from the three articles that constitute Part II of this thesis and discuss the empirical, theoretical and methodological contributions of the research presented in this thesis. As a part of the empirical contribution, I will also explicate and discuss how software effort estimation can be understood as collaborative planning activity, which is the main research question addressed in this thesis.

6.1 Empirical and theoretical contributions

The empirical contribution in this thesis is twofold. Firstly, it provides an understanding of software effort estimation as collaborative planning activity and, secondly, it explicates the complexity of software effort estimation work. To discuss these issues, I will present three process models that describe different aspects of the collaborative work in software effort estimation. These models are derived from the findings and need to be understood in relation to each other. Together, they show how software effort estimation can be perceived as collaborative planning activity that plays out in social interaction by way of meaning-making and recontextualisation processes and different types of tool-mediated actions. Hence, they represent the theoretical contribution of this thesis.

The background for proposing these process models is an in-depth investigation of three different estimation approaches that have been explicated in the three articles that constitute Part II of this thesis. The three approaches studied are named bottom-up, top-down and planning poker (Cohn, 2006; Grenning, 2002; Heemstra, 1992; Jørgensen, 2004b; Sommerville, 2007). Previous studies of these approaches have been conducted with a focus on the individual. The bottom-up and the top-down approaches prescribe two quite different approaches to estimation work, which were described in Chapter 2. The planning poker technique can be understood as a variant of the bottom-up approach. It has been specifically developed and tailored for estimating in teams when agile development practises are followed. However, when employed in practice, all three of these approaches will be realised quite differently than the proposed course of action following a specific estimation approach. Hence, it will not be a one-to-one mapping between the idealised course of action proposed by the three estimation approaches and what actually takes place empirically when teams use these approaches. This is because the different estimation approaches represent principles and orders for actions and are thus guides to a process. It is therefore important that these models are treated as approaches to different ways of conducting estimation work and not as models of how estimation work is conducted.

The research presented in this thesis has, however, revealed a common problem across the three estimation approaches that have been investigated. This problem is related to the fact that the three approaches do not take sufficiently into account the necessity of sense-making and specification work that is needed in complex collaborative problem solving such as software effort estimation.

Although there is a need for normative approaches and models that give direction and guidelines for how to conduct estimation work, understanding estimation work and providing descriptive process models is also important if the aim is to understand and support the practice of software effort estimation. The research in this thesis contributes empirically in this respect by proposing three process models. The three process models proposed show how specification and sense-making processes are conducted and accomplished as interactional achievements in the collaborative work of software effort estimation. Through investigating collaborative work in software effort estimation it has been possible to explicate the challenges faced by software professionals in this work. The process models presented therefore mark a contribution with respect to expanding and clarifying the investigated estimation approaches in order to account for teamwork processes in software effort estimation. Moreover, the process models have an analytical purpose in contributing to the understanding of the complexity of software effort estimation work.

Other researchers have also been interested in the way models and methods in software development are enacted in software processes. For instance, Cohn et al. (2009) have investigated the interplay between software process models and software process enactments. They found that the software process is a generative system, which emerges through conversation in the interplay between models and enactments in which artefacts plays a central role. The research presented in this thesis relates to Cohn et al. (2009) in the sense of how to understand models of estimation approaches that prescribe a description for how the work can be conducted and what is actually taking place in practice. Cohn et al. (2009) have looked at the role of artefacts and software process in software development by following regular work in progress in two software companies over a week. They collected data through interviews, observations of meeting and work in progress as well as by documenting the artefacts that were used in the work. I have on the other hand investigated the work of solving one specific and important task in software development over a short time span. Here, the details of software professionals' collaborative problem solving in estimation work were outlined by following the micro processes in moment-to-moment interaction.

In the following, I will discuss and present three process models of software effort estimation work and show how they together provide an understanding of software effort estimation as collaborative planning activity.

6.1.1 Software effort estimation as collaborative planning activity

The three articles within this thesis have investigated three different approaches to estimation work in teams of software professionals: the bottom-up approach (Heemstra, 1992; Sommerville, 2007), the top-down approach (Heemstra, 1992; Jørgensen, 2004b) and the planning poker technique (Cohn, 2006; Grenning, 2002), which is understood as a variant of the bottom-up approach. Previous researches on judgement-based estimation that are related to these approaches have for the most part been conducted with a different theoretical underpinning than the research presented in this thesis. In most previous research studies, the individual has been taken as the unit of analysis, and experiments have been the dominating research method (Halkjelsvik & Jørgensen, submitted; Haugen, 2006; Jørgensen, 2004b; Moløkken-Østvold et al., 2008). These researches have, however, provided important contributions with regards to understanding what influences the decisions of software professionals on an estimate, which have been shown in, for instance, Jørgensen and Sjøberg (2001), Jørgensen and Carelius (2004), Jørgensen (2006) and Jørgensen and Halkjelsvik (2008). In addition, these researches have explored which estimation approach yields more realistic estimates, shown in, for instance, Jørgensen (2004b), as well as the question if team-based estimation might yield more realistic estimates than individual estimation, shown in, for instance, Moløkken-Østvold and Jørgensen (2004) and Moløkken-Østvold et al. (2008)

Since I have employed a sociocultural perspective in my research, this has allowed for a different focus, a different unit of analysis and a different set of premises for investigating and understanding estimation work. With a focus on the interactional and communicative work I have explicated the teamwork processes when teams have used the three estimation approaches investigated. The results from my analyses show that a similar type of work process is found across the three estimation approaches studied. However, there were some variations due to specific requests or rules following a specific estimation approach.

The main finding across the three estimation approaches, which also was common among all the teams studied, was the need for sense-making processes. Rather than acting on the assumed information directly, the teams needed to make sense of the task, by interpreting the requirement specification or the user story to achieve a shared understanding of what the task was about. Such an understanding needed to be achieved in interaction as a way of making the collaborative problem-solving work move on. The degrees of shared understanding that the participants achieve in their work can be labelled satisficing (Børte, Ludvigsen, & Mørch, submitted). The use of the label satisficing was proposed by Simon (1996) and refers to, here, the fact that the understanding achieved is "good enough" to take the next step in the problem-solving activity, rather than perfect or correct. Jørgensen (2004b) showed in his study of the bottom-up and top-down approach that a large amount of time was spent on the discussion category he called "specification and project understanding". He reported that when the teams employed the top-down approach, 31% of the time was spent on this discussion category and when the teams employed a bottom-up approach, 49% of the time was spent on this discussion category. The research presented in this thesis has, in addition to finding similar patterns as Jørgensen, expanded his findings. While Jørgensen (2004b) used an analytical scheme based on quantification and content, I have demonstrated and specified how the sense-making occurred through the teams' moment-to-moment interactions. In other words, I have explicated how the work is done. To achieve this satisficing, shared understanding, results across the three articles in this thesis showed that this takes place by way of elaboration, clarification and specification. For instance, through elaborating on technical details and clarifying issues of ambivalence, expertise was mobilised through the communicative and interactional work that was conducted. What is interesting about these sense-making processes that are conducted to achieve this satisficing, shared understanding is the way the communicative work goes beyond the information that is initially given in the requirement specifications. This will be discussed in the following.

Through a series of experiments, Jørgensen and Grimstad showed how the information provided in the requirement specification could influence the decisions on an effort estimate (Grimstad & Jørgensen, 2007a; Jørgensen & Grimstad, 2008). These authors were able to detect and establish a connection between what kinds of information were provided in the requirement specification, in terms of irrelevant and misleading information, and the achieved effort estimates. However, I have expanded these findings by describing how teams make sense of the information provided in the requirement specification. In addition, the analyses across the three articles have demonstrated how the requirement specification did not contain enough information to conduct the estimation work and the teams therefore needed to add information in the form of assumptions regarding the context, customers and technical platforms. Rönkkö et al. (2005) made a similar point in their study of the use of planning documents in software development, where it was found that the requirement specification did not prescribe the work needed for implementation. Therefore, the project members had to develop new requirements as the work went along. However, what differs between this study and my analyses is first of all the fact that the time span that has been investigated is different. While I have investigated the moment-to-moment interactions when a specific task is solved, Rönkkö et al. (2005) have looked at a much longer time span that goes across different project phases. The focus on the interactional details within a limited time span made it possible to reveal how issues of ambivalence that were discovered were clarified and specified through technical elaborations. It also made it possible to show how assumptions were made about an imagined future context through sense-making processes in the moment-to-moment interactions. Thus, the planning aspect comes into play as a way of solving the estimation task and not as a way of creating more concrete requirements for development purposes.

In this way, the sense-making processes open for explicating the planning aspect of software effort estimation by revealing how the teams made assumptions about a set of attributes that are assumed to exist in an imagined context. This was demonstrated, for example, in Børte and Nerland (2010), in which properties were assigned to the context and the customer to resolve issues of ambiguity and in Børte et al. (submitted), where the team assumed the existence of a GUI-table (Graphical User Interface) to be able to move on in the work process. By making these assumptions, problems that emerge during the teams' interactional work were solved but also the planning aspects of the estimation work were situated in the same sociocultural context as the problem solving (Suchman, 2007). In software effort estimation, the requirement specification can be perceived as a plan, and the way this plan is used as a resource in the planning activity has been shown in the communicative work software professionals' conduct in which the plan provides topics and issues of discussion as a point of departure. However, the teams also have to move beyond the information that is initially given in the requirement specification to fulfil the planning dimension of estimation work.

Across the three articles the analyses showed that the communicative work was both extensive and important. This was particularly apparent in regards to interpreting and understanding the requirement specification or user stories as cultural tools. In the conceptual framework, the importance of realising meaning potentials to achieve relevant meaning in situ and how this is an interactional accomplishment was pointed out. The results presented in Børte (submitted) showed how the sense-making process also had to include making sense of historical information to be able to utilise it in the present estimation task. Børte et al. (submitted) showed how the sense-making process also had to include the perspective of the users of the system in terms of what they would need to do in a particular situation. The focus on users here can be seen in connection with the agile development practices, which were followed in the project that was studied in this article. The importance of taking the perspective of users has also been demonstrated by Martin et al. (2008), who showed how testers of software systems took the perspective of users in order to decide which tests to run. However, I have, in addition, explicated the sense-making processes that occur through moment-to-moment interactions and showed how the participants created narratives, in terms of an imagined course of action the user of the system must take if the task is implemented in a particular way.

These processes were, however, not straightforward in dealing with one problem after the other as a routine. Rather, the work process went back and forth as achievements were of a temporarily character. This can be considered a result of the co-construction of knowledge, which occurs through the teams' interactional work. This new knowledge is available at a later point in time in the teamwork process and may thereby contribute to new understanding and a need to go back and rework issues that have been attended to in the past.

Another finding that was evident across the three estimation approaches investigated was the need for decomposing the task, i.e., the software system requirement specification, into manageable parts. This was done regardless of the fact that the top-down estimation approach studied in Børte (submitted) specifically requested that the software development system should be viewed as a whole and not broken down into parts. Jørgensen (2004b) also pointed out that the teams had problems following the instructions of the analogy-based, top-down approach and instead performed incomplete bottom-up processes. He argued that this was related to the teams' difficulties of finding similar previously completed projects and thus useful analogies. However, from a sociocultural point of view, we may argue that such decomposition is a necessary means in sense-making processes. This is because meanings have to be generated, articulated and communicated through social interaction in order for others to take part in collaborative problem solving and thus achieve a shared understanding of issues or topics that are attended to (Linell, 2009). This requires interpretations, which are relevant for the teams in the specific situation of use. Doing such interpretations at a surface level when working in teams is not possible because meaning potentials (Rommetveit, 2003) of the different parts need to be realised for the task to have meaning in situ and for the team to achieve the necessary and satisficing, shared understanding to continue their work process. The importance of achieving shared understandings when working in teams has been explored in previous research by Eklund et al. (2010) in their study of team shifts in an IT-support unit. They showed how continuity in collaborative work is a matter of coordinating and securing a minimum of shared understanding to make it possible to continue.

6.1.2 A process model of the interactional accomplishments of software effort estimation

The three estimation approaches that have been studied in this thesis have previously been outlined as idealised models to guide the estimation work. The bottom-up approach holds that an effort estimate is achieved by dividing the project work into different project activities or components before the effort of each activity is estimated (Heemstra, 1992; Sommerville, 2007). Thereafter the different estimates are added up with the possible addition of a buffer to cover unexpected events to form the total effort estimate of the system (Heemstra, 1992; Sommerville, 2007). The top-down approach suggests that a total effort estimate is achieved without breaking the system down into different project parts or activities. Instead, the estimators look for similar previously completed projects that are compared to the current project in which differences are adjusted for before a total estimate is agreed upon. Thereafter, this estimate is distributed over the different project activities

(Heemstra, 1992). There are different ways of employing a top-down approach depending on what is used as the point of departure for finding similarities between completed software projects; however, what is common is that it is the system level that is taken as the point of departure and not the different components. The top-down approach that has been studied in this thesis is a variant that can be termed analogy based, hence it emphasises that the participants shall find project analogies between previously completed projects and the project to be estimated (Jørgensen, 2004b). The third estimation approach studied is the planning poker technique, which is a variant of a bottom-up approach, but specially designed for use in teams when agile development practices are used (Cohn, 2006; Grenning, 2002). The instructions for playing planning poker roughly follows these steps: 1) the estimation task is presented; 2) the task is discussed in the team; 3) each participant privately selects a card representing his/her estimate; 4) once all the participants have selected a card, the cards are flipped over simultaneously; 5) if all the cards show the same number, then that is the estimate; 6) if the cards are not the same, the group discusses by focusing on the outlying values; 7) steps 3–6 are repeated until the estimates converge (Cohn, 2006; Grenning, 2002).

By employing a sociocultural perspective and taking an interactional approach when studying these three estimation approaches bottom-up, top-down and planning poker in depth, it has been possible to develop process models of how software effort estimation work takes place through a series of interactional accomplishments. Hence, the idealised way these approaches previously have been outlined has been expanded in terms of explicating the interactional work that actually takes place when these approaches are realised empirically. When understanding software effort estimation work as interactional accomplishments, the sense-making processes are central together with the oscillation between the planning aspect and the problem-solving aspect of the activity. In Figure 4, a process model of how estimation work takes place through series of interactional accomplishments is presented.

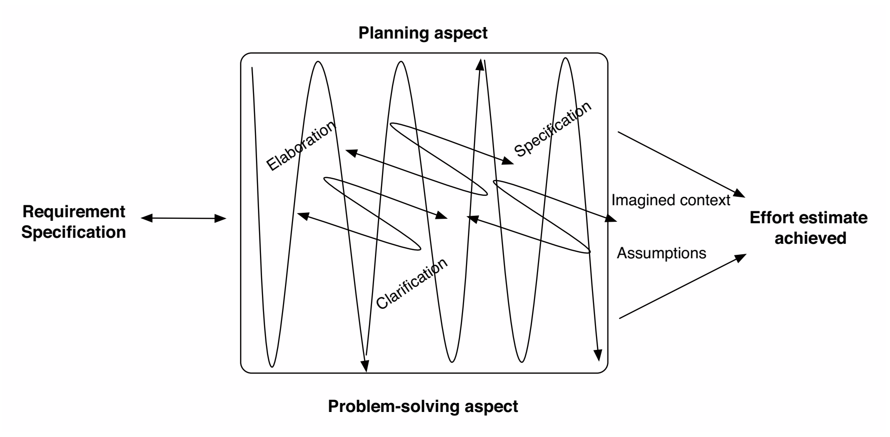


Figure 4 Process model of the interactional accomplishment of software effort estimation.

Across the three articles it was shown that the requirement specification document (formulated as user stories in the study in Article III) represents the main material cultural tool in estimation work. This document describes the software system that is going to be developed and is thus the main source of information about the system (Sommerville, 2007). Though there are no official standards for what a requirement specification should contain, the research in this thesis explicates through sense-making processes the kind of information that is missing and thus needs to be added in the form of assumptions. To estimate a software system, whether it is described in a requirement specification document or formulated as user stories, one crucial task in estimation work is to interpret and understand this document.

Figure 4 shows how the estimation work takes the requirement specification as the point of departure, but the sense-making processes concerning this document are not achieved once and for all time. The two-way arrow indicates the back-and-forth actions between attending to the requirement specification document and the interactional work of making sense of this document. Across the three articles, it was found that interpreting and making sense of the requirement specification was challenging and also a demanding task in the different estimation teams' work process. In previous research on judgement-based software effort estimation, it has been found that software professionals are influenced by irrelevant or misleading information that is provided in the requirement specification when estimating software tasks and that the way the information is written also influences the final decision on an estimate (Grimstad & Jørgensen, 2007a; Jørgensen & Grimstad, 2008). However, previous research has only established this influence through a series of experiments. The research in this thesis contributes to an understanding of the sense-making processes at stake in software effort estimation work and through this provides an understanding of how the information in the requirement specification guides and frames the estimation work conducted by teams.

Rönkkö (2007) has in previous research introduced the documentary method of interpretation from ethnomethodology (Garfinkel, 1984) as a framework for understanding how humans interpret information. This work can be seen in relation to the growing acceptance of what have been called “soft issues” in software engineering and the need for appropriate methods to investigate these issues. The term “soft issues” refers to the social and cultural dimensions of software engineering practice. In his work, Rönkkö (2007) introduced a model with belonging concepts for how to understand the ways in which people understand, interpret and interact with each other. What is different about the research within this thesis is that it opens up the interactional accomplishments and shows how meanings and meaning potentials are realised through communicative work in a specific task in software development in addition to providing analytical concepts for how to understand social interaction.

Through the analysis across the three articles, it has been made evident that the sense-making processes oscillate between the problem-solving aspect and the planning aspect in estimation work. This was particularly emphasised in Børte and Nerland (2010). Secondly, the results showed that these processes were conducted by way of elaborating on technical issues and clarifying issues of ambiguities by adding information, to take an example. This belongs to the problem-solving aspect of estimation work. One way the participants resolved ambiguities in the requirement specification was to make assumptions and, for example, in this way, attribute contextual dependencies regarding customers, existing software systems, technical platforms etc. to an imagined context (Børte et al., submitted; Børte & Nerland, 2010). Making such assumptions about an imagined context demonstrates how the planning aspect in estimation work comes into play in the participants’ interactional work.

The planning aspect of estimation work is situated in the here and now context. However, since the participants need to deal with the future-oriented aspect of a system that is yet to be developed, the planning work is also situated in an imagined context. This imagined context comes about through different imagined scenarios, in which solutions are tested, plus issues related to how the software project could be developed and organised are elaborated on. This was collaboratively achieved through elaboration, clarification and specification in which the planning aspect and the problem-solving aspect of the activity were attended to. For example, in planning poker, the perspective of the user is important in relation to understanding user stories. Throughout the interactional work, there are instances of problem solving, which often are solved by either making assumptions or achieving shared understanding. By revealing how estimation work oscillates between the problem-solving aspect and the planning aspect of the activity, the research presented in this thesis shows what makes estimation work difficult. Further, this research also opens up for understanding the kind of difficulties that occur connected to the sense-making processes of the requirement specification. Figure 4 illustrates how these processes are performed through interaction by oscillating between problem solving and planning by way of elaboration, clarification and specification. Hence, it shows how estimation work consists of a series of interactional achievements in which planning and problem solving constitute two equally important dimensions. Further, it demonstrates how the assumptions about a future imagined context situates and frames the interactional work, making it possible to achieve an effort estimate.

Explicating software effort estimation as interactional accomplishments also demonstrates the need for understanding different types of interactional achievements, in terms of understanding how and what can be achieved through social interaction. The next process model that I will present and discuss is related to understanding how historical information can be used when solving present estimation tasks. Accordingly, this next model provides a more detailed description of specific interactional accomplishments so that different dimensions of the interactional work be explored in more depth.

6.1.3 A process model of how historical information becomes recontextualised to be useful in new contexts.

Above, I have presented and discussed a process model of how estimation tasks get solved through social interaction and that this happens by way of elaboration, clarification and specification as interactional achievements. Software effort estimation work also takes place in an institutional practice that holds a set of collective resources the participants can make use of. In addition to the collective resources, the different participants bring into the practice their sets of individual personal experiences and knowledge. To make use of this knowledge, recontextualisation as a theoretical notion was emphasised in the conceptual framework as a way of understanding how knowledge is adapted and transformed to fit the context in which it is intended to be used (Linell, 1998b; Linell, 2009). Recontextualisation was also used as an analytical concept in Børte (submitted), to capture what happens with knowledge over time and across contexts. By placing an analytical focus on recontextualisation as a process, it was possible to explicate the kind of challenges that are associated with utilising historical information in present estimation tasks.

First of all, using knowledge across contexts is not straightforward (Konkola, Tuomi-Gröhn, Lambert, & Ludvigsen, 2007). This has been shown through several studies; for example, of the transfer between school and work published in Tuomi-Gröhn and Engeström (2003). The reason why such transfers are not straightforward is because of the close link between the knowledge that is acquired and the context in which it was acquired (Säljö, 2005). Knowledge needs to be adapted and transformed to fit the context in which it is intended to be used. Recontextualisation is a process that is accomplished interactionally by way of elaboration, clarification and specification, in which complex meaning-making processes take place and meaning potentials are realised (Linell, 2009). When different types of information are drawn upon when solving present working tasks, different ways of recontextualising knowledge are performed.

Linell (2009) articulated three different ways that recontextualisation could occur: within the same conversation, as reuse between discourses or between genres or activity type. By using recontextualisation as an analytical concept, I have explicated the mechanisms and challenges that are involved in recontextualisation processes and showed how knowledge is adapted and transformed through realising meaning potentials and creating meanings. For example, in the analyses presented in Børte (submitted), I have shown how recontextualisation occurred as reuse between discourses when the team made use of the same reasoning approach they had used previously when estimating another task. This happened even though the use of a different estimation approach was requested. Through the analysis, I also demonstrated how recontextualisation occurs within the same conversation through the ways the teams constructed boundary concepts. The need for recontextualisation has been demonstrated in Ackerman and Halverson's (2004) study of organisational memory. In addition to demonstrating the need for recontextualisation, I have expanded their findings by opening up the mechanisms and identifying the challenges in recontextualisation processes.

As historical information can include collective resources such as the requirement specification and databases containing information about completed software projects, it also includes the individual knowledge and experiences that the participants hold. Personal experiences and expert knowledge also need to be made relevant and articulated in the teams' interactional work to establish who speaks from where. In addition, when making meaning of collective resources like the requirement specification, the personal specialised knowledge that the participants holds needs to be aligned. This is done so that different interpretations and understandings can be made relevant and a shared understanding for how to conduct the work can be established. Further, as part of making relevant historical information, it is necessary to engage in collective remembering (Middleton & Brown, 2005). Collective remembering is conducted interactively and is thus a joint effort taking place through conversation, which makes it possible to make relevant and share the different types of knowledge and experiences that are distributed in a team of software professionals. Collective remembering is thereby a necessity for recontextualisation to occur because different types of knowledge and experiences have to be remembered.

In the following Figure 5, I will present a process model of how historical information becomes recontextualised so that it can be useful in new contexts and thus for solving present work tasks.

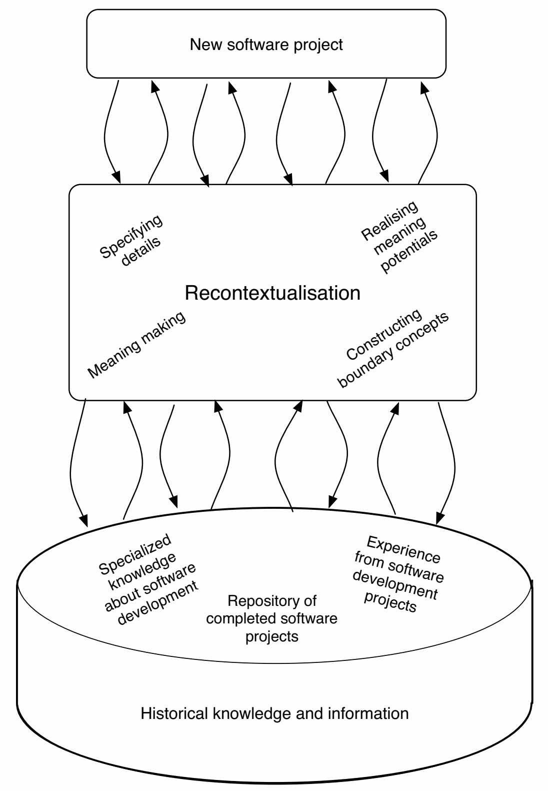

New software project
Specifying details
Realising meaning potentials
Recontextualisation
Meaning making
Constructing boundary concepts
Specialized knowledge about software development
Experience from software development projects
Repository of completed software projects
Historical knowledge and information

Figure 5 Process model of how historical information becomes recontextualised to be useful in new contexts.

In the process model shown in Figure 5, it is demonstrated how historical information becomes recontextualised through interactional achievements so that the knowledge can be used in other contexts. Being able to utilise historical information when estimating new software projects is considered to be useful when pursuing more realistic effort estimates. The point of departure for conducting the estimation work is the new software development project that needs to be estimated. This project is usually described in a requirement specification, as was discussed earlier, and consists of details at multiple levels.

In this thesis, historical information is understood as consisting of several types of knowledge. It includes the specialised knowledge about software development that the different participants bring to the work. It includes the experiences of the different participants from software development projects and includes the information about former and completed software projects that the institutional practice holds; for example, such that the company has collected in a database. Together, this historical knowledge and information can be considered as resources that can be utilised in the estimation work of a new software system. However, using knowledge and experiences across contexts rests on extensive communicative work.

In the process model proposed in Figure 5, recontextualisation is placed at the centre of the figure between the new project that is to be estimated and the available historical information. The curved arrows that oscillate between the new project and recontextualisation and historical information and recontextualisation demonstrate the interactional meaning-making processes the participants have to conduct to make sense of both historical information and the new project. What is interesting about these processes is that the details need to be attended to and meaning potentials need to be made relevant and articulated in order to find possible connection points that can be utilised across contexts and time. Through realising meaning potentials of, for example, different concepts in my study, recontextualisation processes in which knowledge was adapted and transformed occurred in the participants' interactional work.

One way to facilitate the use of knowledge across contexts is the construction of boundary concepts in the teams' interactional work (Børte, submitted). The notion of boundary concept is inspired by the term boundary object (Star & Griesemer, 1989), which holds common features from different contexts. The use of boundary object has been investigated in different types of workplace studies. For example, Ludvigsen, Havnes & Lahn (2003) investigated how sales engineers move in and between different activity systems and how drawings serve as boundary objects. They showed how a boundary object created a common ground for the communicative work and a joint focus for the participants. What I have shown, in addition, is how boundary concepts are constructed by finding possible connection points in which meaning potentials are realised through teams' interactional work.

Even though a drawing is an object, which is different from a concept, they are both cultural tools for which meaning needs to be attributed in different contexts. Thus, both boundary objects and boundary concepts are subject to recontextualisation processes because meaning potentials need to be realised in the context of use and by the participants that intend to use them (Linell, 2009; Rommetveit, 2003).

An example from the analysis is found in Børte (submitted), where the concept of windows was constructed into a boundary concept. In constructing this boundary concept, the participants adapted knowledge from former software projects and thereby made it possible to use historical information about the concept of windows across contexts to establish size and complexity related to the new software project. However, before this could happen, the meaning potential of windows needed to be realised (Linell, 2009; Rommetveit, 2003). In everyday language, one common interpretation of the concept of windows is that a window is something you can see through and it has a framework of wood or maybe metal, which contains a piece of glass that is built into a wall. This artefact thereby admits light and if opened up it can admit air. The situated meaning of windows in estimation work is, however, rather different. The following definition is provided in Wikipedia: “In computing, a window is a visual area containing some kind of user interface. It usually has a rectangular shape. It displays the output of and may allow input to one or more processes”.[6] When talking about windows in an estimation context, it is this last interpretation that the teams are using, but the meaning potential has to be realised even further concerning in particular the complexity of the windows and the number of windows that are needed in a particular system. After clarifying these issues, it was established that the concept of windows had the potential of becoming a boundary concept as it holds common aspects that could be used and understood across contexts. Windows were thereby constructed into a boundary concept, by adapting historical knowledge and information. In this way, it could facilitate the use of historical information across contexts because it could assist in assessing the complexity of the software system that was estimated in the task.

The use of boundary concept in this thesis has demonstrated how particular domain-related concepts are constructed into boundary concepts, which made it possible for

[6] Retrieved 12.02.2011 from http://en.wikipedia.org/wiki/Window_(computing)

the teams to use historical information in present working tasks (Børte, submitted; Børte et al., submitted). Constructing such boundary concepts was, however, only possible when investigating and making sense of the details in both the present task to be solved and the relevant historical information. This is because connection points need to be established and meaning potentials need to be realised.

The process model presented in Figure 5 demonstrates the importance of recontextualising historical information if the aim is to use it in present working tasks. It also provides an understanding of how recontextualisation is conducted through adapting and adjusting knowledge by way of realising meaning potentials and creating meaning. Moreover, it shows why using information from former software projects poses a challenge for software professionals and that storing knowledge in a decontextualised manner is not sufficient or possible if the intention is to use it as collective resources when solving new tasks.

Recontextualisation processes are, however, also mediated by different cultural tools, like software effort estimation work in general. Several different cultural tools may help in this process, which leads to the presentation of the third process model of software effort estimation work.

6.1.4 A process model of the tool-mediated actions in software effort estimation work

One of the basic assumptions following the sociocultural perspective is that all human actions are mediated by cultural tools (Wertsch, 1998). In software effort estimation work, several cultural tools are used in the interactional work to achieve an estimate. First of all, collective resources, such as the routines, rules and guidelines that follow a task with a specific agenda, are available through the institutional practice of software effort estimation. The different estimation approaches studied provide structure, a setting for what kinds of aspects and topics that are relevant to attend to in the interactional work and a known framework and specific guidelines for how to conduct the estimation work. The planning poker approach that was investigated in Børte et al. (submitted) can be considered the most distinct approach in which a set of rules for the game was provided that needed to be followed to accomplish the task and reach a decision on an estimate.

Further, there is the collective knowledge that belongs to the software engineering profession, which is crystallised in different documents, descriptions and ways of talking as a shared platform for conducting the estimation work (Cohn, 2006; Sommerville, 2007). The most dominating and also the most important material cultural tool that belongs to the collective resources is the requirement specification document, which also is formulated as user stories in agile development practices. This document is both a technical description of a software system and a planning document as it is describing something that is not yet developed but is going to be developed in the future. It also has a character of being an inscription device (Roth & McGinn, 1998), as it holds graphical representations of a software system and can therefore be shared amongst the participants.

In the estimation work the requirement specification as a mediational mean frames the estimation task, creates direction and provides the different discussion topics of interest in the team's interactional work. It is also the main issue of investigation in the meaning-making processes because the information provided is ambiguous, and meaning potentials therefore need to be articulated so that the team's interactional achievement generates meaning that is relevant in situ. These findings relate to the results reported in Cohn et al. (2009), who studied artefacts in use in agile software development. They found that, among other things, user story cards, as part of the documentation in agile development, are perceived as the most important artefact in software processes and that it served as a mediational mean not only in the work process of developing it but also as a tool for providing topics of discussion. However, in addition to establishing what the requirement specification was used for in estimation work, I have shown how it is used in the moment-to-moment interaction between software professionals and thus revealed how the meaning-making processes related to the requirement specification occur and how this is a challenge for the participants. This was demonstrated in the analyses across the three estimation approaches investigated.

Being able to create meaning of a requirement specification requires that the software professionals in the team be familiar with this way of representing software systems as well as the professional terminology and concepts used. Language and concepts from the knowledge domain of software engineering are therefore important semiotic tools (Sommerville, 2007; Wertsch, 1991), which are available as a part of the institutional practice from software development as a profession and also a part of the individual specialised knowledge that software professionals hold and bring to the practice. It is through language as a semiotic mediational means that the meaning making occurs. However, it is conducted through micro processes of moment-to-moment interaction by way of elaboration, clarification, specification and also justification.

In Figure 6, a process model of the complex structure of the tool-mediated actions in software effort estimation work is presented.

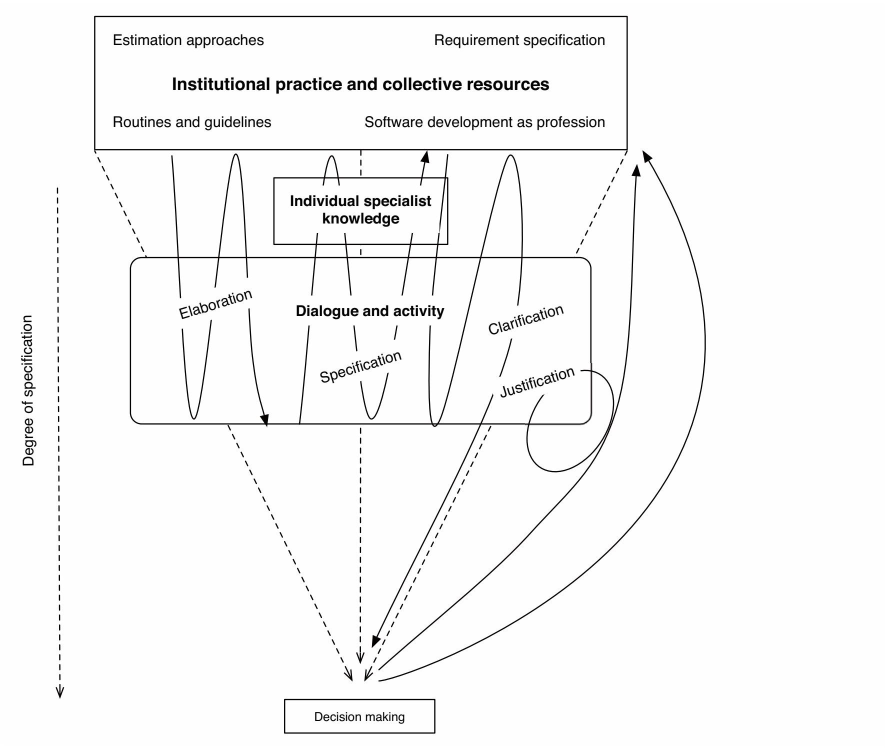


Figure 6 Process model of the tool-mediated actions in software effort estimation work.

The process model proposed in Figure 6 shows the complex structure of the tool-mediated actions that participants in estimation work conduct in the intersection of the three levels of human activity (Valsiner & Van der Veer, 2000). At the top is the institutional practice of software effort estimation with its belonging collective resources. Thereafter follows the individual specialist knowledge that the participants bring to the practice to solve the specific estimation task. Then it is the level of social interaction in which moment-to-moment interactions are mediated by the cultural tools through talk by way of elaboration, clarification, specification and justification. The solid arrows between these three levels show how the interactional work draws upon both the individual knowledge and the cultural resources form the institutional practice. The dotted arrows illustrate the structural level and show that over time this interactional work gets more and more specified in order to reach a consensus and agree upon an estimate.

An example of the mediating role of language is contained in Børte et al. (submitted), where the role of concepts in professional work was investigated in depth. The results showed that in the participants' talk, the frequency with which domain-specific concepts were used in the elaborations were low. Rather, the team used everyday concepts that were given estimation-relevant meaning in the situated work practice. Domain-specific concepts were, however, used when the teams encountered a problem that needed to be framed or reframed. The participants therefore combined their personal knowledge and experiences with the collective knowledge from the knowledge domain as part of their sense-making processes. This happened by way of giving the domain-specific concepts the characteristics of boundary concepts so that they facilitated the use of knowledge across contexts. Moreover, the domain-specific concepts as mediating tools narrowed down, framed the task and solved problems in the team's interactional work. This has been shown across the three articles through the use of imaginary scenarios (Børte & Nerland, 2010), the construction of boundary concepts (Børte, submitted) and the analyses of the use of concepts in Børte et al. (submitted). The domain-specific concepts therefore make relevant and connect the collective knowledge and resources available in the institutional practice with the individual specialist knowledge held by the participants.

It is through social interaction that meaning-making processes, recontextualisation processes or different types of tool-mediated actions are achieved. Moreover, these processes are all conducted with the planning dimension of the estimation work in mind. The planning activity in software effort estimation work is therefore situated through these interactional achievements. What is evident in this work is that even though plans, such as the requirement specification, can be elaborated on indefinitely. The participants elaborate only to the extent that they find it useful, until the elaboration is satisficing. This relates to estimation work in the sense that the work to achieve shared understanding of the task, and the work it takes to develop a task, is accomplished when the team reaches a satisficing, shared understanding. It is a matter of achieving a "good enough" understanding to take the estimation and problem-solving process further rather than achieving a perfect and correct understanding that makes it possible to solve the estimation task. This was particularly visible in the investigations of the planning poker technique in which a satisficing, shared understanding had to be achieved for the team to play a round of planning poker (Børte et al., submitted). Further, it is in particular the requirement specification that mediates the planning dimension of the estimation work, by serving as a resource in the work process. The use of plans in software development projects has been investigated in Rönkkö et al. (2005); however, their focus was on the situations when plans did not work out and what kind of means were necessary to provide for dealing with such situations in software development projects. Hence, they did not investigate how a requirement specification mediates the planning activity as such, which is what I have done.

Even though, a range of mediating tools is used in the work, language is the most important mediational means in software effort estimation, because it is an important part of the communicative and interactional processes between people. Thus, language becomes a tool for participating in specific practices (Säljö, 2001b; Wertsch, 1991). In a practice, such as software effort estimation, the participants can attend to and understand particular aspects of the phenomenon because they are sharing a professional language from the knowledge domain of software engineering. Language thereby makes it possible to articulate meanings in order to share expertise and achieve shared understandings. It is through the use of language that, in software effort estimation work, the participants communicate ideas and meanings, provide justifications or elaborate on a relevant topic to respond to other utterances or initiatives in the estimation work.

By presenting and discussing the three process models above— 1) the interactional accomplishments of software effort estimation, 2) how historical information becomes recontextualised to be useful in new contexts and 3) the mediating role of cultural tools in software effort estimation—it was possible to demonstrate the complexity of the interactional work conducted by teams of software professionals when estimating software development projects. The empirical contribution of this thesis is thereby twofold; firstly, it provides an understanding of software effort estimation as collaborative planning activity. Secondly, it explicates the complexity in software effort estimation work and thereby expands, for analytical purposes, the existing models of the three estimation approaches that have been investigated.

The existing models of the three estimation approaches investigated in this thesis, initially described in Heemstra (1992), Grenning (2002), Jørgensen (2004b) and Cohn (2006), are to be understood and treated as idealised models. The in-depth analysis of the approaches that have been conducted as part of the research in this thesis has provided an alternative understanding and way of interpreting the models other than those in which they have been treated and used in previous research. First of all, the research presented in this thesis expands the three estimation approaches by providing an understanding of the teamwork processes and in particular by incorporating and explicating the sense-making processes that are needed in estimation work. Both the bottom-up approach and the planning poker technique in some sense have considered the need for understanding the software system that is described in a requirement specification. The bottom-up approach does this by emphasising how the project should be broken down into different project activities (Heemstra, 1992). The planning poker technique, on the other hand, does this by emphasising the need for a satisficing, shared understanding before a round of planning poker can be played (Cohn, 2006; Grenning, 2002). However, the approaches did not take sufficiently into account the extent of the sense-making processes that needs to be conducted. Accordingly, the process model presented in Figure 4 contributes an expansion and an understanding of the interactional achievements in software effort estimation work.

The variant of the top-down approach that was investigated in this thesis rests on premises that are a bit more problematic because of the large gap between the idealised model of the approach and what empirically took place in the interactional and communicative work. Hence, the way in which the top-down approach was realised in the teams that have been studied in this thesis deviates from the idealised model to such an extent that it may benefit from a reconceptualisation. However, the research presented in this thesis has in relation to the top-down approach, expanded the understanding of the communicative work that is needed to use historical information across contexts, by explicating the recontextualisation process and demonstrating the challenges that are connected to adapting and adjusting knowledge. Hence, the process model presented in Figure 5 contributes an understanding of why it posits a challenge for software professionals to utilise information from previous software projects.

In addition to developing analytical and prescriptive models of processes of interactional achievements and communicative work, the importance and the use of different cultural tools in the estimation work have been explicated. This applies in particular to the use of the requirement specification and the user stories. None of the existing idealised models of the estimation approaches have sufficiently taken into account the significant role of the requirement specification in estimation work. The research presented in this thesis provides an understanding of how estimation work is framed and guided by the different cultural tools that are available and used in the estimation work. Also, the process model presented in Figure 6 illustrates the complex relationship of tool-mediated actions.

6.2 Methodological contribution

The research conducted as part of this thesis also contributes methodologically to the software effort estimation research field, by introducing interaction analysis (Derry et al., 2010; Furberg & Ludvigsen, 2008; Jordan & Henderson, 1995) as a way of investigating video recordings. Previous research on teamwork in software effort estimation has rarely taken the social aspects into account. Instead, such research has been based on research approaches, which take the individual expert as the unit of analysis. To investigate the collaborative work in software effort estimation, as the research within this thesis does, the unit of analysis has to move from the individual to the social interaction/mediated action. The methodological contribution of this thesis is first and foremost connected to the research method that has been employed to investigate the collaborative work of software effort estimation. It might be considered easy to argue the methodological contribution when a new approach is used to investigate a known phenomenon. However, introducing a new method and new ways of approaching and analysing data to a research field is important. First of all, by introducing a different method a different set of assumptions from a different theoretical perspective follows. This means that the basic underlying assumptions need to be thoroughly explicated together with the acknowledgement of any convincing results in a field where the common methods comprise experiments and statistical tests.

In the research presented in this thesis an interactional approach has been employed to make it possible to attend to the collaborative processes of achieving an estimate. By focusing analytically on the social interaction and the communicative work that is carried out in teams of software professionals, it was possible to open up and describe what goes on in the estimation practice. The reason for choosing an interactional approach and an analytical focus on social interaction is because software effort estimation as an institutionalised practice is increasingly carried out as teamwork in the industry. Further, it is through the use of the collective resources available in the institutional practice together with the specialist competence that the different software professionals bring to the practice that the estimation task is jointly solved through collaboration. This, in turn, implies that it is important to attend to the teamwork processes as such and investigate in depth the kind of work that is conducted to achieve an effort estimate if one aims at supporting and improving the estimation practice.

Interaction analysis is a suitable method for investigating such issues, as it is concerned with the interaction between people and between people and the artefacts that they use in their environment. Therefore, employing interaction analysis as a way of explicating teamwork processes can provide important information about how such work practice is executed and what kind of work the software professionals have to do to achieve an estimate of a software project. As a method interaction analysis stays close to the practice that is investigated, and thus claims about practice can be made in other ways than what experimental data and statistical analysis can provide. By employing interaction analysis as a means to investigate the interactional and communicative work of software professionals, it was possible to investigate the following in depth: 1) what kind of work it takes to achieve an estimate in teams; 2) how previous knowledge and experiences are used; 3) how participants make an account of their arguments; 4) how explanations are produced and perceived as legitimate in the team; and 5) how cultural tools mediate the work practice and team processes.

With this in mind, it is clear that taking an interactional approach to the collaborative work of software effort estimation explicates other dimensions of the estimation practice than what has been revealed by previous research. I will therefore argue that introducing new methods to a well-established field of research is an important methodological contribution, because it opens up for broadening the understanding of the phenomenon under study. This means that different interpretations and different claims provide different results about work, which in this case is software effort estimation. Further, as software effort estimation is a common work practice within software development, a broad understanding is necessary if one aims at improving the practice. It is not enough to attend to the individual expert and achieve understanding of the individual cognitive processes when deciding upon an estimate when the work is carried out in teams. The three process models that have been developed and presented above show how a shift in theoretical perspective, research methods and unit of analysis can shed light on different aspects of software effort estimation and thereby explicate the teamwork processes in estimation work.

6.3 Conclusion

In this thesis, I have argued that a focus on collaborative work and the teamwork processes in software effort estimation is necessary to provide solid insight into the estimation practice. The previous research that has looked into teamwork in estimation is sparse. As was shown in Chapter 2, I was only able to find seven studies in which the focus was on group estimation that had been conducted in a software engineering context with software professionals as participants. More importantly, most of this research had also taken the individual expert as the unit of analysis and thus the collaborative work of software effort estimation had not been addressed. The previous research on teamwork in estimation has therefore explored certain kinds of problems in which certain kinds of inferences were made, but it has left out important aspects of teamwork processes.

The research presented in this thesis have addressed the teamwork aspects and explicated the kinds of challenges that are present in software effort estimation tasks. Thus, it has contributed to a more complex understanding of software effort estimation work. However, this is just one step in the direction of understanding the social aspects in software effort estimation. Researchers have argued the need for attending to the social aspects in software development by making use of qualitative research methods (Dittrich et al., 2007; Dittrich et al., 2009; Fuggetta, 2000; Rönkkö, 2005). This thesis can thus be considered a contribution in this respect, within the specific area of software effort estimation.

The analysis of software effort estimation work that has been presented through the three articles only provides an understanding of software effort estimation at the level of social interaction. Moreover, it is the moment-to-moment interactions that have been taken as the unit of analysis to investigate the interactional work. This provides a somewhat detailed understanding of how software effort estimation work is carried out as practice in teams of software professionals. Further, the employment of the sociocultural perspective as a theoretical lens for understanding the communicative and interactional work allowed for certain types of claims to be made according to how the work is conducted, what is achieved through interaction and how meanings, explanations and decisions are produced.

Further research is needed to provide a thorough understanding of software effort estimation as a practice. The different levels of development that the sociocultural perspective distinguishes between consist of the sociogenesis, the ontogenesis and the micro-genesis, all of which provide opportunities for taking different types of unit of analysis in multiple time and space constellations. The work in this thesis has had an empirical focus on the level of micro genesis because this is where the software effort estimation work plays out. However, how it plays out was understood in interrelation with the participants' ontogenesis and the sociogenesis of the estimation practice.

One direction that future research can take is to place an empirical focus at the level of sociogenesis and investigate how the institutional practice of software effort estimation develops under different constraints; for example, by different software development practices/methodologies. However, by placing an empirical focus on the development of software effort estimation as institutional practice, this needs to be understood in interrelation to how participants enact the estimation practice in different circumstances as well as how situational aspects shape the ways in which estimation work is enacted. Software effort estimation is a recurrent task in software development, thus re-estimation might occur several times during a software project for different purposes. Therefore, combining an empirical focus on the level of sociogenesis by following the trajectory of estimation work in the course of a software development project might reveal important aspects of software effort estimation as institutional practice.

Another possible direction is to place an empirical focus at the level of ontogenesis and investigate how software professionals become skilled estimators through their participation in the estimation practice. Thus, the analytical focus will be on the intersection between social practices and the cognitive individual work that is conducted. Again, by placing an analytical focus on participants in a practice, this needs to be understood in interrelation to the institutional practice of software effort estimation and the activity of conducting the estimation work. Thus, following estimation work of software professionals across different software development projects might explicate important aspects with regards to the development of estimation competence.

In these ways, future research can broaden and expand the research that has been presented in this thesis and thus contribute to a more solid understanding of software effort estimation as a phenomenon.

7 References

Ackerman, M., & Halverson, C. (2004). Organizational memory as objects, processes and trajectories: an examination of organizational memory in use. *Computer Supported Cooperative Work, 13*, 155-189.

Billett, S. (2003). Sociogeneses, Activity and Ontogeny. *Culture & Psychology, 9*(2), 133-169.

Boden, A., & Avram, G. (2009). *Bridging knowledge distribution - The role of knowledge brokers in distributed software development teams*. Paper presented at the Proceedings of the 2009 international workshop on Cooperative and human aspects of software engineering, CHASE'09, Vancouver, Canada.

Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches - A survey. *Annals of Software Engineering, 10*, 177-205.

Boehm, B. W. (1981). *Software engineering economics*. New Jersey: Prentice-Hall.

Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering, 10*(1), 4-21.

Børte, K. (submitted). Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation.

Børte, K., Ludvigsen, S., & Mørch, A. (submitted). The role of concepts in professional work: Unpacking the 'magic step' in software effort estimation.

Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation practice in a multi-specialist team. *Scandinavian Journal of Information Systems, 22*(2), 65-98.

Bowker, G. C., & Star, S. L. (1999). *Sorting things out*. Cambridge: MIT Press.

Bratthall, L., Arisholm, E., & Jørgensen, M. (2001). *Program understanding behavior during estimation of enhancement effort on small Java programs*. Paper presented at the Product Focused Software Process Improvement, Kaiserslautern, Germany.

Cohn, M. (2006). *Agile estimating and planning*. Massachusetts: Pearson Education Inc.

Cohn, M. L., Sim, S. E., & Lee, C. P. (2009). What counts as software process? Negotiating the boundary of software work through artifacts and conversation. *Computer Supported Cooperative Work, 18*(5-6), 401-443.

Creswell, J. W., & Miller, D. L. (2000). Determining validity in qualitative inquiry. *Theory into Practice, 39*(3), 124-130.

De Graaf, W., & Maier, R. (1994). Sociogenesis reexamined: An introduction. In W. De Graaf & R. Maier (Eds.), *Sociogenesis reexamined* (pp. 1-16). New York: Springer-Verlag.

Derry, S. J., Pea, R. D., Barron, B., Engle, R. A., Erickson, F., Goldman, R., Hall, R., Koschmann, T., Lemke, J. L., Sherin, M. G., & Sherin, B. L. (2010). Conducting video research in the learning sciences: Guidance on selection, analysis, technology and ethics. *Journal of the Learning Sciences, 19*(1), 3-53.

Dittrich, Y., John, M., Singer, J., & Tessem, B. (2007). Editorial for the special issue on qualitative engineering research. *Information and software technology, 49*, 531-539.

Dittrich, Y., Randall, D. W., & Singer, J. (2009). Software engineering as cooperative work. Editorial. *Computer Supported Cooperative Work, 18*(5-6), 393-399.

Eklund, A.-C., Mäkitalo, Å., & Säljö, R. (2010). Noticing the past to manage the future: On the organization of shared knowing in IT helpdesks. In S. Ludvigsen, A. Lund, I. Rasmussen & R. Säljö (Eds.), *Learning across sites. New tools, infrastructures and practices.* (pp. 122-137). Oxford, U.K.: Pergamon Press.

Engeström, Y. (1992). Interactive expertise. Studies in distributed working intelligence. Helsinki: University of Helsinki, Dept of Education.

Engeström, Y., Engeström, R., & Kärkkäinen, M. (1995). Polycontextuality and boundary crossing in expert cognition: Learning and problem solving in complex work activities. *Learning and Instruction, 5*(4), 319-336.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*(8), 28-35.

Fuggetta, A. (2000). Software process: A roadmap. In A. Finkelstein (Ed.), *The future of software engineering* (pp. 25-34). New York: ACM.

Furberg, A., & Ludvigsen, S. (2008). Students' meaning-making of socio-scientific issues in computer mediated settings: Exploring learning through interaction trajectories. *International Journal of Science Education, 30*(13), 1775-1799.

Garfinkel, H. (1984). *Studies in ethnomethodology*. Cambridge: Polity Press.

Greeno, J. G. (1998). The situativity of knowing, learning and research. *American Psychologist, 53*(1), 5-26.

Greeno, J. G., Collins, A. M., & Resnick, L. B. (1996). Cognitions and learning. In D. C. Berliner & R. C. Calfee (Eds.), *Handbook of educational psychology* (pp. 15-46). New York: McMillian.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. Retrieved 28.05, 2010, from http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

Grimstad, S. (2006). *Software effort estimation error.* PhD thesis, University of Oslo, Oslo.

Grimstad, S., & Jørgensen, M. (2007a). *The impact of irrelevant information on estimates of software development effort*. Paper presented at the Australian Software Engineering Conference, Melbourne.

Grimstad, S., & Jørgensen, M. (2007b). Inconsistency of expert judgement-based estimates of software development effort. *The journal of systems and software, 80*, 1770-1777.

Grimstad, S., Jørgensen, M., & Moløkken-Østvold, K. (2006). Software effort estimation terminology: The tower of Babel. *Journal of Information and Software Technology, 48*(4), 302-310.

Hakkarainen, K., Palonen, T., Paavlova, S., & Lehtinen, E. (2004). *Communities of networked expertise. Professional and educational perspectives*. Oxford, UK: Elsevier.

Halkjelsvik, T., & Jørgensen, M. (submitted). From origami to software development: A review of studies on judgment-based predictions of performance time.

Hannay, J. E., & Benestad, H. C. (2010). *Perceived productivity threats in large agile development projects.* Paper presented at the International Symposium on Empirical Software Engineering and Measurement (ESEM 2010), Bolzano-Bozen, Italy.

Haugen, N. C. (2006). *An empirical study of using planning poker for user story estimation.* Paper presented at the AGILE 2006, Minnesota.

Haugen, N. C. (2007). Moderne systemutvikling og estimering [Modern system development and estimation] *Presentation held at Estimation seminar 24 October, 2007*. Retrieved 20 October, 2009, from http://simula.no/research/engineering/projects/best/seminars/Estimation%20Seminar%2024.10.2007

Heath, C., & Hindmarsh, J. (2002). Analysing interaction: Video, ethnograpy and situated conduct. In T. May (Ed.), *Qualitative research in action* (pp. 99-120). London: Sage Publications.

Heemstra, F. J. (1992). Software cost estimation. *Information and Software Technology, 34*(10), 627-639.

Heemstra, F. J., & Kusters, R. J. (1991). Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems, 1*(4), 223-237.

Hihn, J., & Habib-Agahi, H. (1991). *Cost estimation of software intensive projects: A survey of current practices*. Paper presented at the International Conference on Software Engineering, Austin, TX, USA.

Hindmarsh, J., & Heath, C. (2007). Video-Based studies of Work Practice. *Sociology Compass, 1*(1), 156-173.

Hughes, R. T. (1996). Expert judgement as an estimating method. *Information and Software Technology, 38*(2), 67-75.

Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences, 4*(1), 39-103.

Jørgensen, M. (2004a). A review of studies on expert estimation of software development effort. *Journal of Systems and Software, 70*(1-2), 37-60.

Jørgensen, M. (2004b). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology, 46*(1), 3-16.

Jørgensen, M. (2005). *The "magic step" of judgement-based software effort estimation*. Paper presented at the International Conference on Cognitive Economics. New Bulgarian University, Sofia, Bulgaria.

Jørgensen, M. (2006). The effects of the format of software project bidding processes. *International Journal of Project Management, 24*(6), 522-528.

Jørgensen, M. (2007). Forecasting of software development work effort: Evidence on expert Judgement and formal models. *International Journal of Forecasting, 23*, 449-462.

Jørgensen, M., & Carelius, G. (2004). An empirical study of software project bidding. *IEEE Transactions of Software Engineering, 30*(12), 953-969.

Jørgensen, M., & Grimstad, S. (2008). Avoiding irrelevant and misleading information when estimating software development effort. *IEEE Software*((May/June), 78-83.

Jørgensen, M., & Grimstad, S. (2009). Software development effort estimation: Demystifying and improving. In A. Tveito, A. M. Bruaset & O. Lysne (Eds.), *Simula Research Laboratory - by thinking constantly about it* (pp. 381-404). Heidelberg: Springer.

Jørgensen, M., & Halkjelsvik, T. (2008). The effects of request formats on judgement-based effort estimation. *Journal of Systems and Software, 83*(1), 29-36.

Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions of software engineering, 33*(1), 33-53.

Jørgensen, M., & Sjøberg, D. I. K. (2001). Impact of effort estimates on software project work. *Information and Software Technology, 43*(15), 939-948.

Jørgensen, M., Teigen, K. H., & Moløkken-Østvold, K. (2004). Better sure than safe? Over-confidence in judgement based software development effort prediction intervals. *Journal of Systems and Software, 70*(1-2), 79-93.

Kjærgaard, A., Nielsen, P. A., & Kautz, K. (2010). Making sense of project management: A case of knowledge sharing in software development. *Scandinavian Journal of Information Systems, 22*(1), 3-26.

Konkola, R., Tuomi-Gröhn, T., Lambert, P., & Ludvigsen, S. R. (2007). Promoting learning and transfer between school and workplace. *Journal of Education and Work, 20*(3), 211-228.

Krogstad, J. R. (2010). *Ikke akkurat på skinner. En analyse av iverksettingen av IKT-prosjektet Flexus i Oslo og Akershus [Not exactly on track. An analysis of the* *execution of the ICT-project Flexus in Oslo and Akershus].* Master Thesis, University i Oslo, Oslo.

Kvale, S., & Brinkmann, S. (2009). *Interviews. Learning the craft of quality research interviewing*. California, USA: SAGE publication Inc.

Lindwall, O. (2008). *Lab work in science education: Instruction, inscription, and the practical achievement of understanding.* PhD, Linköping University, Linköping.

Linell, P. (1998a). *Approaching dialogue: Talk, interaction and contexts in dialogical perspectives* (Vol. 3). Amsterdam: John Benjamins Publishing Company.

Linell, P. (1998b). Discourse across boundaries: on recontextualisation and the blending of voices in professional discourse. *Text: an interdisciplinary journal, 18*(2), 143-157.

Linell, P. (2009). *Rethinking language, mind, and the world dialogically*. Charlotte, NC: Information Age Publishing Inc.

Ludvigsen, S. (2009). Sociogenesis and cognition. The struggle between social and cognitive activities. In B. Schwarz, T. Dreyfus & R. Herskowitz (Eds.), *Transformation of knowledge through classroom interaction* (pp. 302-317). New York: Routledge.

Ludvigsen, S. R., Havnes, A., & Lahn, L. C. (2003). Workplace learning across activity systems: A case study of sales engineers. In T. Tuomi-Gröhn & Y. Engeström (Eds.), *Between school and work. New perspectives on transfer and boundary-crossing* (pp. 291-310). Oxford: Elsevier ScienceLtd.

Mäkitalo, Å. (2003). Accounting practices as situated knowing: Dilemmas and dynamics in institutional categorization. *Discourse Studies, 5*(4), 495-516.

Mäkitalo, Å., & Säljö, R. (2002). Talk in institutional context and institutional context in talk: Categories as situated practices. *Text, 22*(1), 57-82.

Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2008). *Cooperative work in software testing*. Paper presented at the Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, Leipzig, Germany.

Maxwell, J. A. (2002). Understanding and validity in qualitative research. In A. M. Huberman & M. B. Miles (Eds.), *The Qualitative Researchers Companion* (pp. 37-64). Thousand Oaks, California: Sage Publications.

McConnell, S. (2006). *Software estimation demystifying the black art*. Washington: Microsoft Press.

McDonald, J. (2005). The impact of project planning team experience on software project cost estimates. *Empirical Software Engineering, 10*(2), 219-234.

Middleton, D., & Brown, S. D. (2005). *The social psychology of experience. Studies in remembering and forgetting*. London: SAGE publications Ltd.

Moløkken-Østvold, K., & Jørgensen, M. (2003). *A review of surveys on software effort estimation*. Paper presented at the International Symposium on Empirical Software Engineering (ISESE 2003), Rome, Italy.

Moløkken-Østvold, K., & Jørgensen, M. (2004). Group processes in software effort estimation. *Empirical Software Engineering, 9*(4), 315-334.

Moløkken-Østvold, K., & Jørgensen, M. (2005). Expert estimation of web-development projects: Are software professionals in technical roles more optimistic than those in non-technical roles? *Empirical Software Engineering, 10*(1), 7-30.

Moløkken-Østvold, K. J., Haugen, N. C., & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *The journal of systems and software, 81*, 2106-2117.

Norèn, K., & Linell, P. (2007). Meaning potentials and the interaction between lexis and contexts: An empirical substantiation. *Pragmatics, 17*(3), 387-416.

Østerlund, C., & Carlile, P., R. (2005). Relations in practice: Sorting through practice theories on knowledge sharing in complex organizations. *The Information Society, 21*(2), 91-107.

Passing, U., & Shepperd, M. (2003). *An experiment on software project size and effort estimation.* Paper presented at the International Symposium on Empirical Software Engineering, Rome, Italy.

Peräkylä, A. (2011). Validity in research on naturally occuring social interaction. In D. Silverman (Ed.), *Qualitative research: Issues of theory, method and practice* (3rd ed., pp. 364-382). London: Sage Publications.

Pors, K. J., Henriksen, D., Winthereik, B. R., & Berg, M. (2002). Challenging divisions: Exploring the intersections of ethnography and intervention in IS research. *Scandinavian Journal of Information Systems, 14*(2), 3-7.

Radley, A. (1990). Artefacts, memory and a sense of the past. In D. Middleton & D. Edwards (Eds.), *Collective remembering* (pp. 46-59). London: Sage publications Ltd.

Rasmussen, I. (2005). *Project work and ICT: Studying learning as participation trajectories.* PhD thesis, University of Oslo, Oslo.

Resnick, L. B., Pontecorvo, C., & Säljö, R. (1997). Discourse, tools, and reasoning. Essays on situated cognition. In L. Resnick, R. Säljö, C. Pontecorvo & B. Burge (Eds.), *Discourse, tools, and reasoning. Essays on situated cognition* (pp. 1-20). New york: Springer.

Rommetveit, R. (1992). Outlines of a dialogically based social-cognitive approach to human cognition and communication. In A. Wold (Ed.), *The dialogical alternative: Towards a theory of language and mind* (pp. 19-45).

Rommetveit, R. (2003). On the role of "a psychology of the second person" in studies of meaning, language, and mind. *Mind, Culture, and Actvitity, 10*(3), 205-218.

Rönkkö, K. (2005). *Making methods work in software engineering. Method deployment - as social achievement.* PhD thesis, Blekinge Institute of Technology, Karlskrona.

Rönkkö, K. (2007). Interpretation, interaction and reality construction in software engineering: An explanatory model. *Information and Software Technology, 49*.

Rönkkö, K., Dittrich, Y., & Randall, D. (2005). When plans do not work out: How plans are used in software development projects. *Computer Supported Cooperative Work, 14*, 433-468.

Rooksby, J., Rouncefield, M., & Sommerville, I. (2009). Testing in the wild: The social and organisational dimensions of real world practice. *Computer Supported Cooperative Work, 18*, 559-580.

Roschelle, J. (1992). Learning by collaborating: Convergent conceptual change. *Journal of the Learning Sciences, 2*(3), 235-276.

Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In C. O' Malley (Ed.), *Computer supported collaborative learning* (pp. 69-97). Berlin: Springer.

Roth, W.-M., & McGinn, M. (1998). Inscriptions. Toward a theory of representing as social practice. *Review of educational Research, 68*(1), 35-59.

Rowe, G., & Wright, G. (1999). The Delphi technique as a forecasting tool: issues and analysis. *International Journal of Forecasting, 15*, 353-375.

Säljö, R. (1994). Minding action: Conceiving of the world versus participating in cultural practices. *Nordisk Pedagogik, 14*(2), 71-80.

Säljö, R. (1996). Minding action - Conceiving of the world versus participating in cultural practices. In Dall'Alba & Hasselgren (Eds.), Reflections on Phenomenography - Towards a methodology (pp. 19-34). Gothenburgh: Acta Universitatis Gothoburgensis. Retrieved from http://www.ped.gu.se/biorn/phgraph/misc/constr/mindac.html.

Säljö, R. (1999). Learning as the use of tools: A sociocultural perspective on the human-technology link. In K. Littleton & P. Light (Eds.), *Learning with computers Analysing productive interaction* (pp. 144-161). London: Routledge.

Säljö, R. (2001a). The individual in social practices. Comments to Ference Marton's "The practice of learning". *Nordic Educational Research, 2*, 108-116.

Säljö, R. (2001b). *Læring i praksis. Et sosiokulturelt perspektiv [Learning in practice. A sociocultural perspective]*. Oslo: Cappelen akademiske forlag.

Säljö, R. (2005). *Lärande och kulturella redskap. Om lärprocesser och det kollektiva minnet. [Learning and cultural tools. On processes of learning and collective remembering]*. Stockholm: Nordstedts Akademiska.

Säljö, R. (2009). Learning, theories of learning, and units of analysis in research. *Educational Psychologist, 44*(3), 202-208.

Schatzki, T. (2001). Introduction: Practice theory. In T. Schatzki, K. Knorr Cetina & E. von Savigny (Eds.), *The practice turn in contemporary theory* (pp. 1-14). London: Routledge.

Schofield, J. W. (2002). Increasing the generalizability of qualitative research. In A. M. Huberman & M. B. Miles (Eds.), *The qualitative researchers companion* (pp. 171-203). Thousand Oaks, California: Sage Publications.

Silverman, D. (1998). *Harvey Sacks - Social science & conversation analysis*. Cambridge: Polity Press.

Silverman, D. (2006). *Interpreting qualitative data* (Third ed.). London: Sage Publications.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge: MIT Press.

Sliger, M., & Broderick, S. (2008). *The software project manager's bridge to agility*. Boston: Pearson education Inc.

Sommerville, I. (2007). *Software Engineering* (8 ed.): Pearson Education Limited.

Speer, S., & Hutchby, I. (2003). From ethics to analytics: Aspects of participants' orientations to the presence and relevance of recording devices. *Sociology, 37*(2), 315-337.

SSB. (2008). Informasjonssektoren. Sysselsetting omsetning og verdiskapning [Information sector, Employment, turnover and value added]. from Statistics Norway http://www.ssb.no/iktoms/

Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: amateurs and professionals in Berkeley's museum of vertebrate zoology, 1907-39. *Social Studies of Science, 19*, 387-420.

Suchman, L. A. (1987). *Plans and situated actions: the problem of human-machine communication*. Cambridge: Cambridge University Press.

Suchman, L. A. (2003). Writing and reading a response to comments on Plans and Situated Actions. *Journal of the Learning Sciences, 12*(2), 299-306.

Suchman, L. A. (2007). *Human-machine reconfigurations. Plans and situated action 2nd edition*. New York: Cambridge University Press.

Taff, L. M., Borchering, J. W., & Hudgins, J. W. R. (1991). Estimeetings: development estimates and a front-end process for a large project. *IEEE Transactions on Software Engineering, 17*(8), 839-849.

Tausworthe, R. C. (1980). The work breakdown structure in software project management. *Journal of Systems and Software, 1*(3), 181-186.

Temple, B., & Young, A. (2004). Qualitative research and translation dilemmas. *Qualitative Research, 4*(2), 161-178.

Tuomi-Gröhn, T., & Engeström, Y. (Eds.). (2003). *New perspectives on transfer and boundary-crossing*. Oxford: Elsevier Science.

Tuomi-Gröhn, T., Engeström, Y., & Young, M. (2003). From transfer to boundary-crossing between school and work as a tool for developing vocational education: an introduction. In T. Tuomi-Gröhn & Y. Engeström (Eds.), *Between school and work: New perspectives on transfer and boundary-crossing* (pp. 1-15): Elsevier Science.

Valsiner, J. (1994). Bidirectional cultural transmission and constructive sociogenesis. In W. De Graaf & R. Maier (Eds.), *Sociogenesis reexamined* (pp. 47-70). New York: Springer-Verlag.

Valsiner, J., & Van der Veer, R. (2000). *The social mind: Construction of the idea*. New York: Cambridge University Press.

Van Oers, b. (1998). From context to contextualizing. *Learning and Instruction, 8*(6), 473-488.

Vygotsky, L. (1978). *Mind in society. The development of higher psychological processes*. Cambridge MA: Harvard University Press.

Vygotsky, L. (1986). *Thought and language*. USA: MIT.

Wells, G. (1999). *Dialogic Inquiry. Toward a sociocultural practice and theory for education*. Cambridge: Cambridge University Press.

Wertsch, J. (1991). *Vocies of the mind. A sociocultural approach to mediated action*. Cambridge MA: Harvard University Press.

Wertsch, J. (1998). *Mind as action*. New York: Oxford University Press.

Wertsch, J. (2002). *Voices of collective remembering*. Cambridge: Cambridge University Press.

Wetherell, M., Taylor, S., & Yates, S. (Eds.). (2001). *Discourse theory and practice*. London: Sage Publication.

Willcocks, L., Whitley, E. A., & Avgerou, C. (2008). The ranking of top IS journals: a perspective from the London School of Economics. *European Journal of Information Systems, 17*(2), 163-168.

Part II
The Articles

Article 1

Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation work in a multi-specialist team. *Scandinavian Journal of Information Systems*, 22(2), p. 65–98

Scandinavian Journal of Information Systems

Volume 22 | Issue 2 Article 4

12-31-2010

Software Effort Estimation as Collective Accomplishment: An analysis of estimation practice in a multi-specialist team

Kristin Børte
Simula Research Laboratory, kristin.borte@simula.no

Monika Nerland
University of Oslo, monika.nerland@ped.uio.no

Software Effort Estimation as Collective Accomplishment

An analysis of estimation practice in a multi-specialist team

Kristin Børte
Simula Research Laboratory, Norway
University of Oslo, Department of Educational Research, Norway
kristin.borte@simula.no

Monika Nerland
University of Oslo, Department of Educational Research, Norway
monika.nerland@ped.uio.no

Abstract. This paper examines how a team of software professionals goes about estimating the effort of a software project using a judgment-based, bottom-up estimation approach. By employing a social practice perspective that highlights the distributed character of expertise and conceives actions as mediated by cultural tools, the paper analyzes the interactional process through which the estimation tasks were collectively accomplished. The findings show how software effort estimation is carried out through complex series of explorative and sense-making actions, rather than by applying assumed information or routines. During the explorative work, the team alternated between the planning and the problem solving aspects of the activity. The requirement specification served several mediating functions in the interactional process, through which expertise was mobilised and coordinated. The paper argues that to grasp the complexity of software estimation, there is a need for more research that accounts for the communicative and interactional dimensions of this activity. Moreover, by revealing the interactional details of a planning activity the paper contributes to our understanding of the future-oriented and constructive dimensions of social practices.

Accepting editor: Samuli Pekkola

Keywords: Software effort estimation, distributed expertise, teamwork, mediated action, interaction analysis.

1 Introduction

A software project is developed through software processes in which common activities comprise specification, design and implementation, validation and evolution (Sommerville 2007). Several of these processes have a future-oriented character in the sense that they are concerned with planning activities and products that have yet to be realised. A critical challenge is to create predictable software processes so that the software project can be completed on time and in a cost-effective manner (Sommerville 2007). Moreover, the work related to software development is typically multi-specialist in character, in the sense that it depends on different kinds of specialist competencies being combined and aligned in different phases (Faraj and Sproull 2000). Thus, software development may be described as a knowledge-intensive and collaborative practice that unfolds over time by way of complex processes of exploration, negotiation, and decision-making (Barthelmess and Anderson 2002; Fuggetta 2000; Robillard 1999; Whitehead 2007; Ye 2006). To monitor and control the software processes, planning is significant. The estimation of work effort is a core task in this regard as it is used for purposes such as budgeting, trade-off and risk analysis, project planning and control, and software improvement investments analysis (Boehm et al. 2000; Grimstad 2006).

Software effort estimation, however, is a huge challenge, particularly as it is an area in which miscalculations can result in delays, low quality software, or the loss of contracts. A review of studies of software development projects shows that 70 to 80% of such projects overrun their estimates and spend on average 30 to 40% more effort than estimated (Moløkken-Østvold and Jørgensen 2003). Today, as in the early days of computer programming, software projects whose plans and budgets are based on effort estimates that are too low experience severe management and delivery problems. Thus, more knowledge about the estimation practice of software professionals and the challenges faced in such work is important for project management.

Most research on estimation has been concerned with developing different kinds of formal estimation models. However, judgment-based estimation is the most frequently applied estimation approach in the software industry today (Bratthall et al. 2001; Heemstra and Kusters 1991; Hihn and Habib-Agahi 1991; Jørgensen 2004a). This paper examines the work a team of software professionals does when estimating the effort of a software project using a judgment-based approach. An effort estimate is here understood as the most likely number of work hours necessary to complete a software development project as assessed by the managers and developers responsible for delivery (Grimstad et al. 2006). In this approach, the quantification step (i.e., "the step where an understanding of the software development estimation problem is translated into a quantitative measure of the required effort" (Jørgensen 2007, p. 450)) is based on a judgmental process as opposed to an algorithmic calculation. How this judgmental process comes about and how it is collaboratively achieved has, however, proved difficult to reveal (Jørgensen 2005). This paper aims to contribute to filling this gap by examining the work conducted to arrive at an effort estimate as a social and communicative practice. To disclose the details of this practice we focus on the social interactions through which the estimation tasks are collectively explored, negotiated, and accomplished.

A requirement specification describing the details of the project to be developed forms a point of departure for the estimation process. Usually, this document outlines what the software system should do in detail, including the services and functions the system should provide and the constraints under which the system must operate (Sommerville 2001). This paper sees the constitutive elements of the interactional process as the team's way of approaching the information provided in the requirement specification and the way of using this as a basis for collective exploration and negotiation. We commence, however, by positioning our analysis within related strands of research.

2 Related research

Given that our analysis focuses on teamwork processes in estimation, we will use two strands of research as a backdrop: studies of judgment-based estimation and studies of collaborative work in software engineering.

2.1 Studies of judgment-based estimation

Previous research on judgment-based software effort estimation has been largely rooted in the cognitive strand of empirical research with the aim of improving the accuracy of estimates. One avenue of this research focused on professionals' reasoning processes when estimating effort. Here, research showed that the reasoning process is based more on intuition than on deliberate reasoning (Hughes 1996). A series of experimental studies revealed that software professionals' decisions on an estimate were influenced by factors such as anchoring, over-optimism, and wishful thinking (Aranda and Easterbrook 2005; Jørgensen and Grimstad 2005; Jørgensen and Grimstad 2008).

Furthermore, researchers have explored the accuracy of different estimation approaches. In a large industrial study, two estimation approaches—top-down and bottom-up—to estimate software projects were compared (Jørgensen 2004b). The main results showed that the bottom-up approach, in which the project work is divided into different project activities and the effort of each activity is estimated separately and then added up, on average, gave the most accurate estimates. The results further show that almost half the time (49%) was spent on discussions related to understanding the requirement specifications and the project context and on discussions leading to breaking down the project into activities. An interesting finding of Jørgensen's (2004b) study is that, in many cases, the software professionals were unable to explain how they accomplished the actual quantification step of the estimation process.

As these studies illustrate, the research on judgment-based effort estimation has mainly been concerned with investigating individual reasoning when expert judgments are made. Estimation work is, however, now increasingly carried out as unstructured group work in the industry. A survey conducted at JavaZone in 2007, where answers from 101 software developers were collected, indicates that 40% use group work when estimating. This is an increase of approximately 30% over the last five years (Haugen 2007). In spite of this, research on judgment-based estimation processes in groups seems to be limited. For instance, Taff et al. (1991) developed and studied "estimeetings" as a structured method for estimating in groups. They found that the estimates achieved in "estimeetings" corresponded well with the actual effort expended on the project. Moløkken-Østvold and Jørgensen (2004) studied group discussions as a way of comparing and discussing estimates arrived at on an individual basis and found that the estimates were more accurate after group discussions when compared to the average of the individual estimates. However, neither of these studies has focused on group discussions as such, or attempted to reveal the collective explorations undertaken in these processes.

2.2 Studies of collaborative work in software development

Software development projects face a number of challenges due to size, geographic distribution, and multi-professional cooperation (Herbsleb and Moitra 2001; Mockus and Herbsleb 2001). This points to the need for coordination on different levels. One strand of research has focused on how software tools can facilitate coordination between software professionals in distributed software projects (Al-Ani et al. 2008; Happel et al. 2008; Panjer et al. 2008). Studies in this area have, respectively, developed a software tool (Al-Ani et al. 2008), proposed semantic technologies for sharing knowledge (Happel et al. 2008), and shown how obstacles in coordination occur when using software tools (Panjer et al. 2008).

Another strand of studies has focused on different types of roles in teams and their effect on the teamwork (Cherry and Robillard 2009; Sarkkinen 2004). For instance, by examining a planning discourse, Sarkkinen (2004) showed how ways of representing issues in a planning frame could become a power struggle, in the sense of not including the other participants in the planning discourse. A few studies, such as the one by Boden and Avram (2009), have looked at how collaborative processes are organised. They showed how small software companies rely upon the role of engaged team members as knowledge brokers to share knowledge across distributed software development teams. Martin et al. (2008) examined cooperative work in software testing and showed how testers took the perspective of users in order to decide what tests to run. These studies are relevant for research on estimation in that the capacity to imagine events that can occur during the development and use of a system is constitutive of the planning process.

Planning is significant when dealing with future-oriented work such as estimation, and studies that focus on this aspect are thus of particular relevance for the present article. Some studies have shown that planning is a recurrent process in software development. Rönkkö et al. (2005), who looked at the use of plans in a software development project over several project phases, show how coordination problems were dealt with in various ways. They revealed that planning documents, such as project plans and requirement specifications, provide means to identify and act upon deviations in addition to guiding the development work. In the reported study, the requirement specification was identified as a plan that did not prescribe the work needed for implementation. Hence, the project members had to develop new plans and requirements as the work went along to create a basis for the future project work. This means that the specification served as a tool for identifying aspects where re-negotiations and re-planning were necessary.

Also, in estimation work, the requirement specification constitutes the basis for the planning activity; however, the way different understandings play out and get negotiated in the interactional process among team members is not examined in this context.

While the research discussed above has focused on IT artefacts as planning tools in software development, few studies provide minute analyses of planning activities as such. This may be due to the generally limited use of qualitative methods in the software engineering research field (Dittrich et al. 2007). Micro-oriented studies have, however, been shown to generate important insights in the ordering and accomplishment of other types of collaborative work. Studies of "planning talks" in other areas have revealed how language practices often imply movement between different activities and types of work in ways that do not follow a predefined sequence of actions (Asmuss and Svennevig 2009). For instance, drawing on data from different studies and workplaces, Holmes and Stubbe (2003) showed how exploratory meetings involve multiple shifts between activities such as planning, brainstorming, and problem solving, and how the topical structure of such interaction both allows for and depends on digression rather than a linear development. Housley (1999) found that members of multidisciplinary teams in the area of social care regularly contest, negotiate, and accomplish various roles in meetings where they plan activities. In both cases, the shifts were mediated and accomplished by way of language-based communication. Opening up the details of the communicative practice may thus increase our understanding of how planning is actually carried out.

2.3 Research objectives

Even though the above described strands of research have produced sound knowledge about what influences judgments and accuracy of estimates and about collaborative work in software development more generally, few studies have combined the two. Moreover, most studies that examine collaborative processes in software development have selected longer time frames—stretching over days, weeks, and sometimes months—as their unit of analysis. As a result, the collaborative process of making sense of a requirement specification and arriving at an estimate is not sufficiently described. To gain insight into this process, we need to examine the details of how software professionals identify, explore, and negotiate issues at stake in the different estimation steps and how they make further collective decisions.

This paper addresses this gap by analyzing the collaborative practice of software effort estimation in a multi-specialist team. Even though effort estimation may be seen as a recurrent activity in software development projects, we argue that it forms a distinct process in such projects, which comprises distinct problems and challenges. By focusing on estimation tasks as such, we can gain a better understanding of this kind of work. To investigate how the estimation task is approached, negotiated, and resolved as an unfolding process, we will focus on the interactional dimensions of estimation work. The questions we raise are as follows:

1. What characterizes software effort estimation as a collaborative activity in a multi-specialist team?
2. What types of communicative and explorative work are needed to accomplish the planning task and agree on an estimate?

This analytical focus on the communicative and collaborative aspects of software effort estimation has not been employed, to our knowledge, in previous research on estimation. Thus, the analysis in this paper will explicate dimensions of estimation practice that have not been investigated in detail before to enrich our understanding of the challenges practitioners face in this kind of work.

The paper is organised as follows: In section 3, we outline a theoretical perspective on estimation as a social practice taking form by collective accomplishment. In section 4, we describe the data material and analytical strategy. Section 5 presents the data analysis, which reveals how a multi-specialist team goes about accomplishing the task of arriving at an effort estimate. Section 6 summarizes and discusses the main findings in relation to our research questions and previous research. Finally, section 7 concludes by pointing to some implications for research on software effort estimation.

3 Theoretical framework: Software effort estimation as social practice

3.1 Emerging social practices

As a theoretical point of departure, we regard software effort estimation as social practice. This perspective highlights the dynamic interdependencies between humans, between humans and their material resources, and between individuals and collectives. Rather than understanding properties attached to individuals or artefacts as stable and independent of each other, a core premise in practice theories is that such properties are developed and given meaning only in relation to other subjects and objects in situated activities (Østerlund and Carlile 2005; Schatzki 2001). As a consequence, practice theorists typically take situated activities as their unit of analysis and attempt to reveal how ways of knowing and acting are collectively constructed in activity in specific ways. During recent decades, a differentiation of approaches has emerged, highlighting different dimensions of social practice. Some emphasise the regulative power of rules, habits, and embodied repertoires of action, and stress how the practice in question is historically constituted and reproduced. Others place their analytical interest in the productive aspects of practice, and focus on how activities and products emerge through ongoing interaction (Østerlund and Carlile 2005; Schatzki 2001). Researchers also differ in how they frame their units of analysis in space and time. Some contribute to theory by revealing the mechanisms by which practices are continued and sustained over time (e.g., Engeström 1999; Lee and Roth 2008). Others contribute by examining the interactional dynamics by which people make sense of and engage with their social and material worlds (e.g., Brown and Duguid 2001; Eklund et al. 2010).

This paper follows the latter approach to examine how estimation tasks are made sense of and accomplished in the context of teamwork. Although recognizing that a social practice like software effort estimation rests on institutionalised and established ways of doing certain types of work, established routines are not sufficient for solving specific tasks. Rather, practitioners need to develop shared understandings of specific problems (Schatzki 2001). As pointed out by Suchman and Trigg (1996), the maintenance of social practices over time includes ad hoc behaviours in which specific, non-routine problems are dealt with against the backdrop of established routines. In this perspective, the order of practice is not given, but rather produced in local, ongoing activities. Hence, this paper takes a micro-oriented position in the field of practice studies where practice is understood as situated accomplishments of people taking part in local activities (Lynch 2001). Our focus is on how estimation practice takes form when people act and make sense of concrete tasks by drawing on established rules and institutionally generated patterns of action.

3.2 Estimation practice as collective accomplishment

The practice of software effort estimation is typically organised as a collaborative activity in which software professionals with different specialties combine their forms of expertise to accomplish a given task (Haugen 2007). To understand and explore this practice as a collective and situated accomplishment, two notions are of special importance: the notion of distributed expertise and the notion of mediated action.

The notion of distributed expertise contests the traditional view of cognition and expertise as properties of individuals. Instead, these are understood as constructed in encounters and exchanges between humans and between humans and artefacts (Engeström 1992). As a consequence, expertise is not seen as stable or well-defined capabilities, but rather as emergent in activities and collectively accomplished. Previously achieved ways of knowing need to be negotiated, recontextualised, and recreated to be meaningful in specific settings; the ways in which expertise is performed are thus context-specific (Lemke 2000; Vygotsky 1978; Wertsch 1991).

Multi-specialist teams comprise specialists in different areas of software development, and hence somewhat different forms of expertise. Estimating the effort of a software development project is thus seen as a matter of collectively exploring, negotiating, and making relevant the different experiences and forms of knowing present in the team as resources for achieving a shared understanding of the given task. This may include resources that derive from other contexts of practice. For instance, visual models, concepts, or technologies generated in other activities may be introduced and used as resources in present problem solving (Engeström et al. 1995). Coordination of perspectives is needed even when practitioners do not move across contexts. Their framing of problems, as well as their exploration of possible solutions, may draw on resources from other contexts, which are introduced and made relevant in the given activity.

The notion of mediated action underscores that people always make sense of and engage in social practices by means of cultural artefacts and tools. Thus, ways of knowing and performing practical work are both embedded in and constituted by cultural tools. The tools we have at our disposal influence what kind of problems we can solve, and they may provide suggestions for how the problems can be approached. Sociocultural perspectives on practice highlight two types of tools: semiotic tools—such as language, sign systems, and professional vocabularies—and material tools, such as models, instruments, and physical devices (Orlikowski 2006; Østerlund 2008; Säljö 2005; Wertsch 1991). These interact in complex ways in social activities and serve to constitute the space for possible actions.

Furthermore, these tools are often open to different interpretations and uses (Knuuttila and Voutilainen 2003; Orlikowski 2006). Ways of understanding them are therefore not assumed but emergent in activities. Their actual use is a product of negotiations and sense making in practice. As noted by Barnes (2001, p. 25), social practices constantly have to be generated "by agents concerned all the time to retain coordination and alignment with each other in order to bring them about."

Communication is crucial for this work, not just for sharing knowledge but also as a joint dynamic engagement in which collective exploration of ideas may result in a new and shared understanding. The communicative actions may be marked by habits and conventions about how work is carried out in specific domains. It may also be marked by improvisation and imagination. To reveal how social practices emerge as situated accomplishment, we need to examine how actors communicate and interact with each other in specific settings through the use of mediating tools (Greeno et al. 1996; Wertsch 1998).

In the context of software effort estimation, achieving a shared interpretation of the requirement specification and how it can be used as a basis for inventive and future-oriented activities is a central task for estimation teams. Such an interpretation is most likely not established once and for all, but is rather an ongoing and recurrent challenge. Rönkkö's (2005) study shows how plans are revisited and explored in a continuous manner in software development. Eklund et al. (2010) make a similar point in their study of team shifts in an IT support unit, showing how achieving continuity in collaborative practices is a matter of coordinating and securing a minimum of shared understanding in different stages of the process to make it possible to continue. The present paper contributes to this strand of research by examining what it takes to proceed and accomplish effort estimation as a specific process in software development projects.

4 Data material and analytical strategies

The analysis presented in this paper is a re-reading of videotaped data from a large study of software effort estimation conducted within the Norwegian branch of an international IT-consultancy company in June 2002. The design of the study was quasi-experimental in character, in the sense that seven teams of software professionals were organised and asked to estimate two legitimate software projects by applying the two estimation approaches of bottom-up and top-down. Considerable efforts were made to ensure the authenticity of the estimation process. Actual requirement specifications of software projects received from the company's customers were used as estimation tasks, and software professionals working in the given firm were hired as participants. To resemble actual estimation teams as much as possible, each team comprised one experienced project manager and one or two developers.

The setup of the study allowed for focused examinations of the challenges faced by estimation teams in ways that could have been difficult to explore in software development projects that evolve over longer time frames. Even though we don't follow the estimation process as it plays out in a real development project, the data are suitable for investigating core processes in teamwork. More concretely, the design provided opportunities for examining the interactional activity by making negotiations and collaborative ways of reasoning within a focused and researchable timeframe accessible. It also provided access to how the requirement specification was used for estimation purposes. The entire corpus of data has previously been analysed and reported in Jørgensen (2004b). For a detailed description of the setup and data collection, we refer to page 4-5 in Jørgensen's article. The instructions for the estimation task and corresponding description can be found enclosed in Appendix A.

To gain insight into estimation as collaborative practice, we have chosen to analyze the interactional process of one team that applied the bottom-up estimation approach. This approach is the richest in interactional work and, to the participants in the study, it is the preferred and best-known approach. The selection of a team was based on the analysis of the entire corpus previously described by Jørgensen (2004b). In addition to choosing a team that corresponded to the typical characteristics identified in the quantitative analysis, we used the richness of the interaction among team members as a criterion for selection to ensure a solid dataset. The estimate provided by the chosen team was too low compared to the actual effort expended on the project. Such over-optimism is a known problem in effort estimation and the choice of team may explicate this issue.

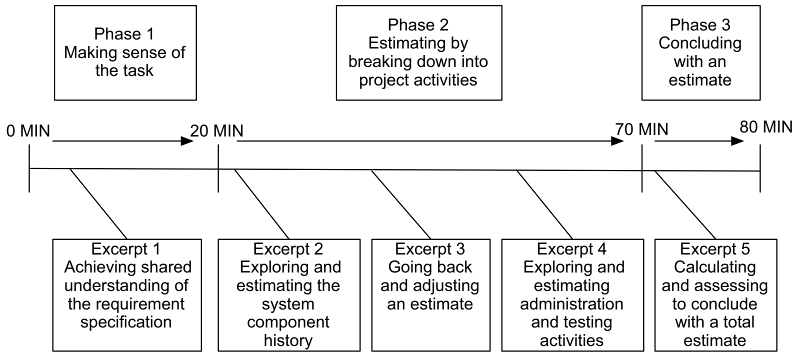
Our analytical interest is oriented towards the collaborative actions by which the team approaches, makes sense of, and produces collective knowledge as they engage with the problem (Orlikowski 2006). Thus, the analysis focuses on what kind of problems the team needs to explore and what they collectively achieve by the actions taken. The video recordings are analysed by means of interaction analysis, which is an interdisciplinary method for studying how people interact with each other and with the cultural tools that are available in their particular environment (Jordan and Henderson 1995; Lantz-Andersson et al. 2008). Interaction analysis rests on the socio-cultural notion that knowledge and actions are social in their character and situated in the interaction between members of a particular practice. The focus on the participants' collaborative achievements makes the method appropriate for studying social processes in estimation teams. It allows for an investigation of how participants make sense of each other's actions and aligns their expertise with each other to solve the estimation task (Jordan and Henderson 1995).

Interaction analysis can be performed in different ways for different purposes. Our approach is to follow the content dimension of the interactional process. In line with the theoretical perspective on practice as situated accomplishment, we focus on the moment-to-moment interactions of the team to reveal which problems they need to engage with, how different resources are made relevant in their task accomplishment, and how they go about achieving a collective understanding that allows them to proceed towards an estimate. To open up the details of the interactional practice and make these issues accessible for analysis, the following concepts are used as sensitizing means: *orientation*, *elaboration*, *clarification*, and *positioning*. These concepts are adopted from Furberg and Ludvigsen (2008) and Rasmussen (2005), and were fine-tuned for this specific analysis. Orientation refers to how the participants orient themselves towards the estimation task and the problem-solving activity. Elaboration describes how the team opens up and makes sense of problems by adding information and trying out different solutions and ways of thinking. Clarification allows the team to narrow down problems and to close gaps as a means to achieve shared understanding of the task. Positioning captures the dynamic aspects of who speaks from where in the interaction and how shifting positions enhance the alignment of different types of expertise.

The video-recorded data from the selected team add up to a total of 80 minutes. First, the whole team process was analysed inductively through repeated viewings of the data and a sorting of the discussion in relation to content. Then, excerpts were chosen for an in-depth analysis of the interactional process. These have been presented and discussed in research workshops to refine and validate the analysis. The excerpts presented below are chosen for their capacity to display the dialogue as it progresses and the role it plays in the process of accomplishing the estimation task. Some of the excerpts are condensed, which is illustrated by […] in the transcript[1].

5 The interactional process of estimating a software project

The team we follow in this analysis consists of one project manager, one developer, and one database specialist. Hence, they form a multi-specialist team that is temporarily put together to solve this specific estimation task. The main artefact the team has available for solving the estimation task is the requirement specification. This document is developed for software professionals, so it is marked by a professional terminology. Knowledge of this terminology is, therefore, crucial for relevant participation. Software estimation is characterised by interpretations and discussions concerning the requirement specification and the future-oriented aspect of relating to a project that has yet to be realised. After completing an inductive analysis through repeated viewings of the whole estimation process and sorting the discussion in relation to content, three main phases of work were identified. In Figure 1, we have visualised the content and type of work conducted in the team that followed after each team member had read the requirement specification and the estimation process instructions.



Phase 1 Making sense of the task
Phase 2 Estimating by breaking down into project activities
Phase 3 Concluding with an estimate
0 MIN
20 MIN
70 MIN
80 MIN
Excerpt 1 Achieving shared understanding of the requirement specification
Excerpt 2 Exploring and estimating the system component history
Excerpt 3 Going back and adjusting an estimate
Excerpt 4 Exploring and estimating administration and testing activities
Excerpt 5 Calculating and assessing to conclude with a total estimate

Figure 1: Visualization of the different phases of the estimation process, identified by an inductive analysis of the team's discussion

As Figure 1 shows, we identified three different main phases in the team's estimation process. From these phases, a total of five excerpts were selected to illustrate the types of work conducted and the main turns taken in the discourse, where some things are temporarily solved, making it possible to proceed in the work.

In the first phase, making sense of the task was the main objective. To achieve this objective, the team referred to the estimation instructions, which included the work breakdown structures (WBS) (Appendix A) and the requirement specification available on the table in front of them.

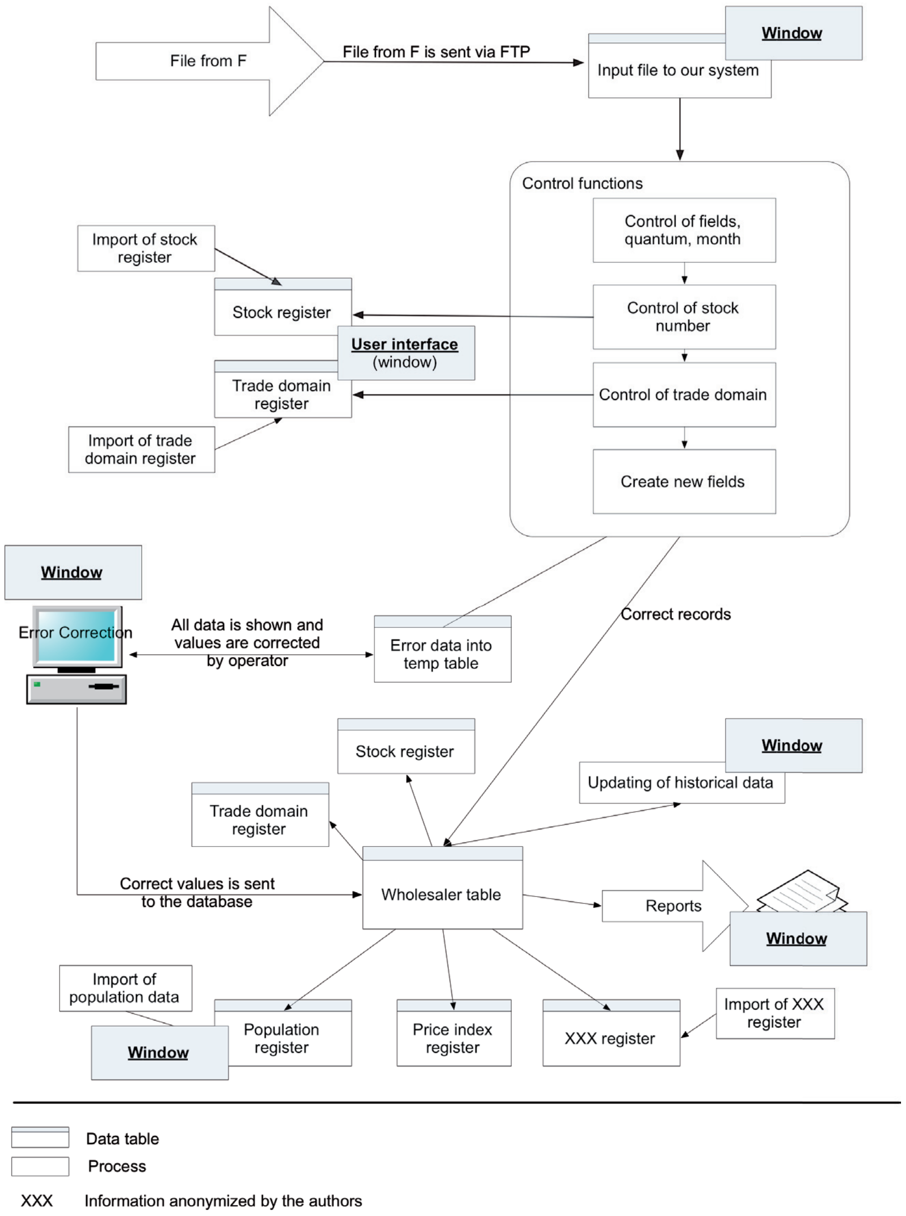


Figure 2: Diagram of the software system adapted from the requirement specification[2]

The team decided to begin their estimation work by addressing the project activities related to the design and programming of the software system. These project activities could be considered the most familiar and concrete parts of software development work and can therefore be perceived as the easiest to estimate. The team, therefore, did not choose to follow the sequence listed in the WBS. The requirement specification, which comprised a diagram of the software system together with text that explained the different elements of the diagram, was the object that in large part mediated the team's estimation work (see Figure 2). The WBS then served as a checklist for the project activities the team had to address. Excerpt 1 illustrates how the team achieved a shared understanding of the task at hand by actively making use of the requirement specification.

In the second phase of the discussion, the team explored the different project activities in depth and assigned estimates. Excerpts 2, 3, and 4 show the explorative and complex expert work needed in the estimation practice. In the third phase of the estimation discussion, estimates from the different project activities were added up to form the total effort estimate of the software development project. Excerpt 5 illustrates how the team reached a conclusion on a total effort estimate.

5.1 Achieving shared understanding of the requirement specification

As a starting point for solving the estimation task, the team needed to interpret and comprehend what the project to be estimated was about. The requirement specification serves as an important mediating tool and as an object of investigation in this process. The team used it in two different ways to achieve understanding, as shown in Excerpt 1. We entered the video-recorded process at the beginning of the estimation discussion where the sequencing of the work process was about to be set and an elaboration of the work needed for developing the software system will be initiated. The team consists of a database specialist (DB), a software developer (D), and a project manager (PM).

Turns	*Verbal communication*	*Description of actions*
1. DB:	The first thing we have to do at any rate is that we have to implement what it says here, then we'll see if it's right.	DB looks at the diagram. PM is picking up a pen, looking in DB`s requirement specification together with DB. D looks in her requirement specification.
2. PM:	Yeah	
3. DB:	And of course that takes some time, so we have a one-off job here. We'll convert what we have from Oracle to an SQL server. (2,5) I don't understand why they are going to do this, but it is (5,3). The challenge here is that. It says at the back here, that they don't have, they don't have any Oracle installations themselves. So the question is whether we can assume that they have access to an existing Oracle installation so that we can get it over, access the database directly, or whether we have to get it on files and define the file format.	
4. D:	Of course, there's not supposed to be any online interface [at all]	

5. DB:	[No, and] then we have to	
6. D:	Everything will just go on files.	PM looks in the requirement specification while talking.
7. PM:	I see it also quite clearly as the e::h e::h transfer of historic data. Then, someone whether or not it is a part of this, I am not quite sure at the moment, which then extracts it from the Oracle, but of course the most important part of the job here is to get the data put into the database. So that you. The format is different, so you can't just plop it in.	
8. DB:	Yeah, yeah, because it says that they are different formats and that it is a one-off job. It's something you do just once. (2,2) This is a bit difficult since I don't know how much data is supposed to be transferred. It doesn't say very much about it.	All three team members look in their requirement specification while talking.
9. PM:	In my opinion, it needs to be interpreted based on that we have defined, or we are now defining the database we want. Then we will need some data, and we have to take that from the old one, which we will actually have to use as a starting point. What we want to include and assume will be available in one form or another in the old one.	
10. D:	Mm	
11. DB:	Mm	DB looks in the requirement specification while talking. PM starts taking notes.
12. DB:	As I interpret the text here, there is at least wholesale data that we are retrieving from there. Getting a comma-separated file is not a big job. If we, that is, if we assume they can retrieve it for us, and if not, we will need people who know both Oracle and Outsider. No, excuse me, SQL Server. Because then you need to be able to export it from there, and it must be possible to make an import program in the other database.	

Excerpt 1.

The requirement specification first serves as a guide in sequencing the work process of the team, clearly shown in the language used and indicated by the repeated term "here" when the team follows the documentation (lines 1-3). This sequencing of the work process initiates an elaboration about the meaning the requirement specification has for the software development, which shifts the artefact's use to one of a means of discussion. This change in use happens when the database specialist questions a lack of information in the requirement specification (line 3). Through the elaborations that follow, the team employs different orientations where a dual focus on the technical details and the overall functionality is taken care of by the database specialist and the developer respectively. The elaborations conducted (lines 1–6) identify problems and ambiguities (line 3) that are resolved in lines 4–6.

Solving these problems is achieved in two ways in the interaction: by clarifications or by making assumptions. Clarification happens when the requirement specification is directly used to clear up overlooked features in the elaboration. For instance, the developer clarifies some intent in the requirement specification and thus rejects an assumption about it with the utterance in line 4. Assumptions are used to both delimit and resolve a problem that often takes the form of assigning properties to the context or, in this case, to the customer. This is what is happening with the database specialist's utterance in line 12.

In this interplay between elaborations and clarifications, assumptions drive the interaction forward and create possibilities to pinpoint and narrow down the elaborations. This narrowing down is achieved through an articulation of the main challenges at the close of the elaboration and establishes a shared understanding, making it possible for the team to continue in their work. This is what happens with the project manager's utterance in line 7. The articulation also leads to another aspect of the estimation discussion—namely, planning. Planning the development of the software system activates the future-oriented dimension in the estimation discussion, which is of vital importance for assigning the number of work hours needed for development. Moreover, this articulation creates a connection between the two aspects of problem solving and planning, making switching between them possible. The articulation also closes down the problem solving conducted through elaborations and establishes a shared understanding that facilitates the planning of the software development before new problems are identified and elaborated on. These connections thereby function as prime movers in the dialogue.

This form of elaborating, together with the contributions that emerge according to the different positions team members take on, allows the team to make their different forms of expertise relevant. The database specialist's expert knowledge on the technical part of programming and development is made relevant when he elaborates on the technical details connected to the issue at stake, positioning as a technical expert in the interaction in line 3. The developer's capacity to interpret and hold the overall focus by making clarifications becomes relevant because it allows her to elaborate in line 4, which positions her as a team facilitator. The project manager articulates the main challenges, demonstrating a capacity to ensure progress in the dialogue, which positions him as a lubricator, as illustrated in line 7. Collectively, the team is able to achieve a shared understanding of the requirement specification that enables them to continue the estimation work.

The analysis of Excerpt 1 shows that the team's work is oriented towards making sense of the task, both in terms of what issues the team members are trying to resolve and the sequential organization of these issues. The requirement specification shifted from being an organizing tool that sequenced the work process to serving as the means of discussion for achieving a shared understanding through elaborations. In the interaction, the continual switching between the problem solving aspect and the planning aspect served as a mover in the dialogue, ensuring that different tasks were completed. In addition, the team members aligned their different forms of expertise with each other by taking on different positions in the interaction.

5.2 Exploring and estimating the system component history

When a general understanding of the project to be developed was achieved, the team went on to explore the different project activities in depth, and assigned estimates to these, making the requirement specification the main object of investigation. In Excerpt 2, the team is exploring the system component history, illustrating how this type of exploration facilitates the assigning of work hours believed to be necessary for developing this component. We enter the process when members suggest a solution for how to handle the system component history.

Turns	*Verbal communication*	*Description of actions*
1. DB:	Only some of the fields in the stock register will follow a history. In other words, the very hard part here is that they don't tell me what fields should be history, when it should be transferred to history tables, and how long it should be kept.	DB is pointing in the requirement specification while talking.
2. D:	No, it looks like they flag or tag the fields that are traced in the history. So then you need some way of finding out which values he wants to track and then create the history of it, and just ignore all the others.	D looks in the requirement specification.
3. DB:	And then we, what we are actually saying is we have a, If you have one of these, stock reg	DB makes a drawing while talking.
4. D:	Mm (6,0)	
5. DB:	and then you have a table named "hist", for example, which is identical to it, and just copy it over, we can, for example, have a help table that tells us which fields should be copied over.	PM is turning pages in his documents and start taking notes.
6. D:	Mm	
7. DB:	Then you could just let the database do it itself. Then you have some time interval or another that you copy over when you.	
8. D:	But I envisage that this one will be fairly consistent in size all the time, except to be updated with the new values, while this one here will grow and grow and grow and grow.	D is pointing in the drawing made by DB.
9. DB:	grow grow grow grow. And what is usually done is that you have. That this one is partitioned, that you have, for example, every Q1, Q2, Q3 right?	DB continues drawing while talking.
10. D:	Mm	
11. DB:	Q4 or month or week or something like that in the history table. You build it up depending on what you want, then you delete it or transfer it to tape when a long time has passed.	
12. D:	Mm that depends on how much data we're talking about? They said it was a million lines a year.	D looks in the requirement specification and PM takes notes.
13. DB:	Mm. That's not a lot….	
14. D:	No, maybe it isn't.	
15. DB:	Worked at [name of company] and there it was a billion. That was fun. We didn't use Access then.	
16. D:	But the import itself of the inventory list, just the basic import, I don't think is very difficult.	
17. PM:	No, but then you need the history	
18. D:	We are talking about one	

19. DB:	But what we probably should do here because it's a question of history in many places. I think we should build a standard component for history, that will be the same everywhere. You take a table, you have a table and then we create a history table over it because they don't know which fields, so we have a help table that tells us which fields should include history and when they should be run, every 14 days, then you have a "cron job" that runs in the background that just copies it over continuously.	DB is pointing in his drawing while talking. D is looking in the requirement specification.
20. PM:	Mm	
	(4,0)	
21. DB:	And then, to save the history data you will spend, or I guess you will spend (2,5), 16 + 16, 16 hours for design, and 16 hours for development to create history material and then you get it for free for everyone (3,0). Then its automat - that is for all the tables for which you want history. I've made this before.	PM takes notes.
22. PM:	Yes	
	[.... Time leap, 30 seconds]	
23. PM:	Yeah, yeah, I've set that up. I noted 16+16 for general history. But for the two register lists, I haven't noted anything yet. You say they are simple, is it 8 for each of them or …	
24. DB:	8+8 .... The import of the stock register, it is 8 for each of them.	
25. PM:	Import stock plus trade, that makes 16 then.	
26. DB:	Yes, and then that brings us down to XXX	

Excerpt 2.

In Excerpt 2, the requirement specification serves as the main object of investigation. The team elaborates on the specifications of the system to be developed by identifying gaps and adding information. This is what happens with the utterance made by the database specialist in line 1. The elaboration then moves on to try solving the different problems that are uncovered by imaginary scenarios concerning design solutions that are supported by drawings and explanations (lines 2–11). The capacity to engage in such imaginary solutions is dependent on the orientation taken towards problem solving. The database specialist and the developer's orientation is solely toward technical details, but is held in two ways: by imagining being a part of the solution—as one who performs the actual programming work (line 1)—and by keeping an eye on the overall results (line 2).

The kind of elaboration that creates imaginary scenarios also makes a certain type of expert knowledge relevant, and by negotiating, problematising, and challenging suggested solutions, the different kinds of expertise are combined. This is what happens when the developer explains how the solution is envisaged (line 8), challenging the database specialist's scenario by seeking further explanations (lines 9-11). The explanations provided are supported by drawings. The negotiations and problematising that emerge show that collective achievement of understanding is necessary to be able to move on in the elaborations. It is not enough that only one member of the team understands; the understanding needs to be shared, as demonstrated in the need for clarifications seen in lines 2–11.

When a shared understanding of the imaginary scenario is achieved, a change occurs in the direction of the teams work. A funny remark allows the team to go on elaborating a new project activity, but this remark triggers a need for a leader to direct and ensure that the work regarding one project activity is completed and an estimate assigned. This is achieved through the project manager's utterance in line 17, where the team's focus shifts back to the original project activity, which makes possible a connection between the discussion around problem solving and the discussion about planning the time necessary for developing the suggested solution.

In the planning discussion that follows, elaborations are almost absent. Instead, the team employs culturally shared categories of numbers of work hours, which correspond to one or two days of work. This is what happens with the database specialist's utterance in line 21. The use of these shared categories is also evident in the further interaction in lines 22-26. The assigning of work hours for a project activity closes a gap in the teams understanding. After the work hours are determined, the discussion about importing the two registers is closed by an estimate (line 25), and the team is then able to continue in their estimation work by addressing the next problem to solve (line 26).

The imaginary scenarios created are made possible by the positioning as technical experts of both the developer and the database specialist. Through this positioning, different technical expertise is made relevant and, by supporting the elaborations with drawings and explanations, the team is able to align the members' technical expertise with each other to achieve a shared understanding. The developer attends to the overall functionality and ensures that what is required of the system due to the specification is followed. Her utterance in line 2 activates her capacity to keep an overall perspective on the task and pay attention to the guidelines relevant for the creation of imaginary scenarios. The database specialist's capacity to imagine carrying out the actual developmental work required of the technical solutions suggested in line 5 activates his technical know-how and makes it relevant in this particular practice. These different forms of expertise that are combined during the elaborations make it possible to achieve a shared understanding that, in turn, facilitates the assigning of estimates to the project activity.

The analysis of Excerpt 2 shows that the capacity to form imaginary scenarios and add information is central in achieving a shared understanding of a project activity. In this example, the team used drawings and explanations to try out solutions, activating different forms of technical expertise, which again allowed the team to close the gaps and conclude this project activity.

5.3 Going back and adjusting an estimate

Our analysis of the estimation practice revealed that this was not a linear process but rather a process marked by returns and digressions. Excerpt 3 illustrates some of this complexity. In an interactional loop in the team's work, the team needs to return to a previously solved problem to adjust the estimate. We enter the process when the team is just about to end their elaborations concerning a particular window about control functions and the project manager is summing up the elaborations made.

Turns	*Verbal communication*	*Description of actions*
1. PM:	That control moves on to these that have an error. It is entered into its own temporary table	Everyone is looking in the requirement specification.
2. DB:	Mm	
3. PM:	There you actually get a window where you have access to all the data. Based on the error table, you can correct it there. There you are supposed to have a screen so then, so then, so then you can bring anything up in the table and make corrections.	PM is taking notes
4. DB:	Mm. That's fairly straightforward	
5. D:	Yeah	
6. DB:	Because it's not, int's no mean feat to create a screen with several fields across, right? That is in facts an 8 as I see it.	
7. D:	It is fairly a simple read and copy from a list, or a	
8. DB:	It's just that you need a long screen	
9. D:	Yeah	
10. PM:	A very long screen	
11. DB:	A very long screen	
12. PM:	Expensive assumption	
13. DB:	Yeah, yeah expensive assumption you have to scroll a little. Okay, we've got it then	
14. PM:	Yeah, that also makes error recovery possible. And then it says that after the control functions the data will be, directly into the wholesaler table. That is that when you make a correction here, as well as the right ones, they will go into the wholesaler table. But that - that is the control isn't much. Is it a big deal to get it over? It has - there is so little data here, so it's nothing	

Excerpt 3a.

The articulation of the main points from the elaboration conducted prior to Excerpt 3a allows the team to achieve a shared understanding of the project activity and to make the switch to the planning aspect of assigning estimates. This is what happens with the utterance made by the project manager in lines 1-3.

In the ensuing dialogue, an assessment concerning the level of difficulty is conducted and a corresponding estimate is assigned, which serves as a gap closing of the elaborations on one project activity while opening up for exploration of the next. This is illustrated by the utterances in lines 4-14, where a closure of elaborations is made (line 13) and the team continues elaborating (line 14).

In the following 2 minutes, the team continues elaborating and assigns estimates on the next project activity. Through the elaborations they achieve a deeper understanding of the project activity that, the team realizes, affects a previous estimate of another project activity (shown in Excerpt 3a). We enter the process again after 2 minutes when a link to this project activity is discovered through the team's elaborations.

Turns	*Verbal communication*	*Description of actions*
1. DB:	And then there are the two others who go into error condition. They go down to the left. The one that is over there, there	DB is pointing at the diagram in the requirement specification while talking.
2. D:	Mm	
3. DB:	They are the ones who - that have failed. Then you go over here, where you make error corrections, then they go down into the wholesaler table. Then there's the window that you have here. It must be, really then, from a users's point of view you should be able to see the content of the other tables too.	
4. D:	Mm	
5. DB:	It will be much easier for them that is	
6. PM:	Okay, it says here that they have access to a lot. So maybe that window is a little more complex then?	
7. DB:	Yes, that's what I'm wondering about too	
8. D:	Yeah	
9. DB:	If we are going - build it into the screen already in that window	
10. D:	But have you already estimated it	
11. DB:	Then we'll increase it	
12. PM:	Is it up to 16, or will it be more?	
13. DB:	No there is more. Then, then, those two are the most complicated because the application's functionality lies in those two functions	
14. PM:	Yeah	
15. DB:	That screen and the import functionality are what are hardest to estimate	
16. D:	Mm	
	(5,0)	
17. DB:	Do you agree?	
18. PM:	Yes I agree because there is quite a bit of functionality here.	

Excerpt 3b.

The elaborations conducted in Excerpt 3b are characterised by clarifications and explanations of the sequencing of one elaborated project activity in Figure 2. The sequencing of the technical aspects performed opens up for taking a different perspective on the suggested solution to the problem. This change in perspective initiates the interactional loop in the team's work, and happens with the utterance made by the database specialist in line 3.

For the team to perform the interactional loop being initiated, the new argument put forward during the elaborations needs to be acknowledged and clarified by the team members before they can be considered acted upon. This clarification and acknowledgement is what takes place in the utterances in lines 6-9, before a decision to act upon the new argument is proposed in line 11.

Identifying the need for such an interactional loop calls for a combination of different kinds of expertise. In positioning as a user held by the database specialist, activating the capacity to form an argument from another perspective is made relevant. Speaking from the perspective of a user of the software system opens up for exploring other solutions than the one decided upon. However, the interactional loop cannot be performed without combining it with the knowledge of how to make connections and moving back and forth across time in the problem solving, a process maintained by the project manager who acts from a position of lubricator, ensuring progress. In the utterances made in lines 1-12, this interactional loop takes place.

The analysis of Excerpt 3 shows how an interactional loop is performed and how it rests upon the team's capacity both to take different perspectives during elaborations and problem solving and to preserve an overview of the entire work process.

5.4 Exploring and estimating administration and testing activities

In addition to the technical aspects, different project-related activities need to be explored to complete the estimation task. Excerpt 4 shows the explanatory work conducted by the team when they estimate activities related to administration and testing. We enter the process when the database specialist initiates the discussion of the first non-programming project activity, namely, administration.

Turns	*Verbal communication*	*Description of actions*
1. DB:	And administration and such, isn't that about 10% of the total time?	
2. PM:	Well, I want at least 15 then, + 5% meetings	
3. DB:	That much? Yes, you also have to attend.	
4. PM:	We still have testing left. What I've been looking at here is that the development work, including the database, will take about 490 hours (3,0)	PM looks in his notes.
5. D:	M[m]	
6. DB:	[Mm]	
7. PM:	How much testing should we have for it? (5,0)	
8. DB:	It's good we have [name of developer] here who is a testing expert, so can't you please tell us.	
9. D:	I'm not an expert. I think when you say 8 hours for a single window, then I think they say 1 or 2 of those hours are for testing. 1 hour.	
10. PM:	Yes, I'm not talking about self-testing. Now I'm thinking more of the tests that come afterwards for them. I see that as included in the test. Those 8 include the self-test	
11. D:	But, first of all, for instance, it says that it should be delivered within 10 weeks right? [... Time leap, 33 seconds]	D is looking in the requirement specification.

12. D:	A week for testing and error correction?	
13. DB:	Mm	
14. D:	Mm. Then you have 10%, I don't think that's so	
15. DB:	10 %	
16. D:	wrong.	
17. PM:	10% or approx. 50 hours are testing.	
18. DB:	testing and error correction?	
19. D:	or a week's work for testing and error correction	
20. PM:	a week's work?	
21. D:	no, or 40 hours then	
22. PM:	I think that's a bit optimistic	
23. DB:	I think that is too little, yes	
24. D:	too little?	
25. PM:	yes, I think 10% is too little, I'd like to increase it to at least 20	
26. D:	yes, then its 2 weeks until	PM looks in the requirement specification and takes notes
27. DB:	no, it's just twice as many people	
28. D:	80 hours	
29. DB:	twice the number of people	
30. D:	Mm, yes it is	

Excerpt 4.

In Excerpt 4, elaborations concerning the project activity administration are missing in the team's explanatory work; instead, the team employs "rules of thumb" to a large extent. These rule of thumb usages are products of collective ways of knowing that are culturally and historically preserved in practices involving similar types of work and are made relevant and utilised by our multi-specialist team in this particular practice.

The team use rule of thumb in lines 1 and 2 to come to an understanding of the scope of the project activities as well as an indicator for calculating the number of hours necessary. In the interaction, there seems to be a shared pre-understanding of what the administration activity consists of, because it is left unelaborated. The team's work here is mainly concerned with planning rather than with problem solving. Orientations are taken towards an overall assessment of the calculated estimate of the design and programming activities, which is held by both the project manager and the database specialist. The utterances in lines 1 and 2 show this employment of rule of thumb.

Apart from a clarification of one rule of thumb that is needed to achieve a shared understanding of the notion of testing (lines 9–10), the team's explanatory work—which at this point is mostly concerned with the planning aspect of the project—is oriented toward the future. The primary debated factors at stake here are time and work hours. This kind of discussion is opening up to a more extended debate on the suggested percentages for estimates than that which characterised the previous planning discussions of the interaction. The arguments used, however, are treated as given, leaving explicit elaborations out of the work process. The interaction from lines 11–25 illustrates this point well.

The different forms of expertise necessary to assign estimates to the non-programming activities are made relevant both explicitly and implicitly. An explicit request for the specialist competence of the developer to address testing is made. The developer activates her rule of thumb knowledge, though with a little hesitation, with the utterance in line 9. A more implicit activation of expert knowledge is provided through the project manager's positioning as an administrative leader when he both leads the interaction (lines 4–7) and comes up with what he thinks is the relevant rule of thumb to be employed (lines 2 and 25).

The analysis of Excerpt 4 shows that rule of thumb is employed to a large extent when estimating the non-programming project activities such as administration and testing. The team draws upon a shared body of knowledge, unlike their collective exploration of project activities related to programming and design. Different types of tasks seem to generate different interactional patterns.

5.5 Calculating and assessing to conclude with a total estimate

After the different project activities have been estimated, the discussion enters the third phase, where the team concludes with a total estimate of the software project. Excerpt 5 illustrates the work conducted when concluding. We enter the process when the team is starting to add up the different project activity estimates for one of the three total estimates they were asked to come up with in the estimation task (Appendix A).

Turns	*Verbal communication*	*Description of actions*
1. D:	What are we missing then? Documentation. Shall we summarise? That's fun.	PM is taking notes. D looks in PM`s notes.
2. PM:	yes, that's fun, mmmm	
	(7,0)	
3. D:	you have completed the estimates here too	
4. PM:	yes, yes, yes, I'm quick, you know. But what if you take mine? Have you started summarising here?	D reaching for the calculator. DB reaches for his, but put it back on the table.
5. D:	can start. If you say, no, if you just say the numbers as we move down, it's easier for me.	
6. PM	Yes, 8	
7. D:	Yes	
8. PM:	8 20 16 16 56 40 24 8 16 8 40 16 16 16 40 192 8 40 16 80	PM is reading the estimates from his notes out loud. DB is collecting his papers. DB pays attention to PM.
9. D:	80?	
10. PM:	yes, 40 and 40.That is what we have so far. What does it add up to?	
11. D:	764	
12. PM:	764, that looks OK. Then we need 20% of that altogether (3,0) 8, 12, 14, that is 153.	
	(10,0)	

13. PM:	Maybe it takes 764 times 15% of the (mumbles) (16,0)	PM takes notes. D uses the calculator.
14. DB:	No, 1.5	
15. PM:	yes, but I want 15%	
16. DB:	Ah, you are to have 15, excuse me	PM is taking notes.
17. PM:	That makes 15 hours for that, and I will get 5%, which works out to	
18. DB:	one-third of it	

Excerpt 5.

A call for summarizing in line 1 initiates the practical work of adding up the different project activity estimates. This makes the team change the main artefact from the requirement specification to the notes taken throughout the estimation discussion, which serve as the most important semiotic tool for the work conducted. A calculator is also introduced in this phase, serving as the material tool for helping the team perform the calculations of the different project activity estimates. In the work that follows, orientations taken towards the calculation are twofold. One orientation is directed towards the details and the act of calculating performed by the developer (lines 1 and 5), and the other—performed by the project manager—is directed towards the result of the calculation and the subsequent act of assessment (lines 12-13).

The team does not embark on any elaborations, but instead their work is characterised by being directed towards closing on a total estimate. The work it takes to reach a conclusion on a total effort estimate does not call for exploration in the same way that the work in the previous phases did; instead, the team follows a more direct path by calculating and adding numbers. The exchange in lines 4 and 5 between the project manager and the developer shows the linear strategy they agree to follow.

When the team summarizes the different project activity estimates, culturally shared categories are frequently repeated. The numbers for hours of work, 8, 16 and 40, which correspond to one work day, two work days, and one week of work, respectively, have been used extensively throughout the estimation work of the previous phase. After summarizing and closing in on an acceptable estimate, the team refines the estimate further by calculating the different rules of thumb that were applied as indicators for the non-programming estimates, which are then added to the total. The kinds of expertise activated to conduct this form of work are project management and prior experience with this type of administrative work, which is facilitated by the project manager's positioning as an administrative leader, shown in his utterance in line 12. Administrative expertise is combined here with a competence in calculation and the act of using the calculator to provide different answers, functions employed by the developer who is positioned as someone who completes. Together, the team is able to close on a total estimate.

The analysis of this last excerpt shows that even though elaborations are absent, the team's work still calls for different types of expertise, together with an extensive use of artefacts so that they can complete the estimation task. The team spent less time on this part, which was marked by a different interactional pattern characterised by few elaborations and a more straightforward form of work.

6 Findings and discussion

Our analytical interest in this paper has been to reveal the collaborative and communicative work that comprises software effort estimation in teams. The aim was to explicate the details of the work that needs to be done to arrive at an estimate and to enrich our understanding of the challenges that software professionals face in this work. In the introductory section, we formulated two related research questions to guide the analysis: (1) What characterizes software effort estimation as a collaborative activity in a multi-specialist team? (2) What types of communicative work are needed to accomplish the planning task and agree on an estimate?

The analysis of the social interaction in the selected estimation team shows that arriving at an estimate is a complex process, which requires considerable communicative and intellectual work among the team members. When we look at the different types of work in relation to the time frame of the activity, one outcome of our analysis is that the process of software effort estimation is not so much about quantification as it is about identifying and exploring problems to achieve a shared understanding of the issues at hand. With the exception of the final ten minutes, the different phases were characterised by extensive discussions and explorative work before the quantification of project activities was realised. Thus, we find the same overall pattern as Jørgensen (2004b). However, our analysis examines the process further and offers insights into what it takes to understand the tasks at hand and how the team collectively accomplishes the estimation task and finally agrees on an estimate.

One challenge in this regard was related to the interpretation of the requirement specification as a basis for achieving a shared understanding of the project to be estimated. As the analysis shows, the information provided in this document was ambiguous and difficult to use in a direct manner. The team members needed to add information and explore different aspects of the identified problems to be able to proceed. Excerpt 1 showed how assumptions in the form of assigning properties to the context or to the customer were used to solve problems of ambiguities and achieve a shared understanding of the requirement specification. Excerpts 2, 3, and 4 showed, in different ways, how the team needed to explore the information beyond what was given through elaboration and clarification. This process also involved adding information through visualization—for instance, with drawings—as a means of creating and testing imagined scenarios. In this work, the team members took different positions and oriented themselves differently towards the problems at hand, which generated resources for the collaborative process of producing an estimate. Although the quantification steps at first seem rather straight, the analysis shows how these steps rely on and require considerable explorative work so that the team members could know which rules of thumb to activate. Jørgensen (2005) found that software professionals were unable to explain how they reached the quantification step, and hence described this as the "magic step". However, the analysis presented in this paper reveals how this step emerges from extensive elaborations and clarifications in the preceding discussion in which the details of the software project to be developed are collectively explored by mobilizing and coordinating expertise through social interaction.

Moreover, our analysis shows that software effort estimation does not play out as a straightforward process in which one issue is dealt with and completed at a time. Rather, the interactional structure was characterised by shifts in times, topics, and roles. Several achievements had a temporary character and needed to be reworked later in the process as new information or ways of understanding emerged. The process was brought forward as the team members collaboratively resolved a series of dilemmas by moving back and forth between technical elaborations and integrative windups. Their sense-making process thus alternated between addressing the planning aspect of the estimation practice—that is, questions related to how the software project could be developed and organised—and the problem solving aspect of the practice, reflected in issues and questions that needed further elaboration and clarification to be resolved and (re) positioned in the planning strategy.

We previously referred to the study by Rönkkö et al. (2005), which showed how software development was characterised by recurrent processes of planning along an extended timeline. Our analysis shows that a similar pattern characterizes the micro-processes of estimation within the timeframe of a team discussion, and point to how planning is taken forward by continually opening and resolving new problems and questions for the team to engage in. Shifts in topics and roles have been described as typical for 'planning talks' (Asmuss and Svennevig 2009; Holmes and Stubbe 2003; Housley 1999), but until now this has not been explored in detail in the context of software effort estimation. As shown in the present study, planning in software effort estimation is partly a matter of specifying the technological consequences of the information provided in the requirement specification and partly a matter of creating imaginative scenarios through which different solutions are tested. The identified need to move back and forth in collaborative exploration, and for alternating between planning and problem solving, suggests that planning should be understood as a continual aspect of such processes, which forms an important basis for problem solving.

Figure 3 visualizes the interactional pattern of the activity performed by the team. The undulating arrows indicate how the discussion oscillated between the planning and problem solving aspects. The circular arrows indicate how the team was able to move back in time to problems already addressed when new levels of understanding were achieved in the interaction.

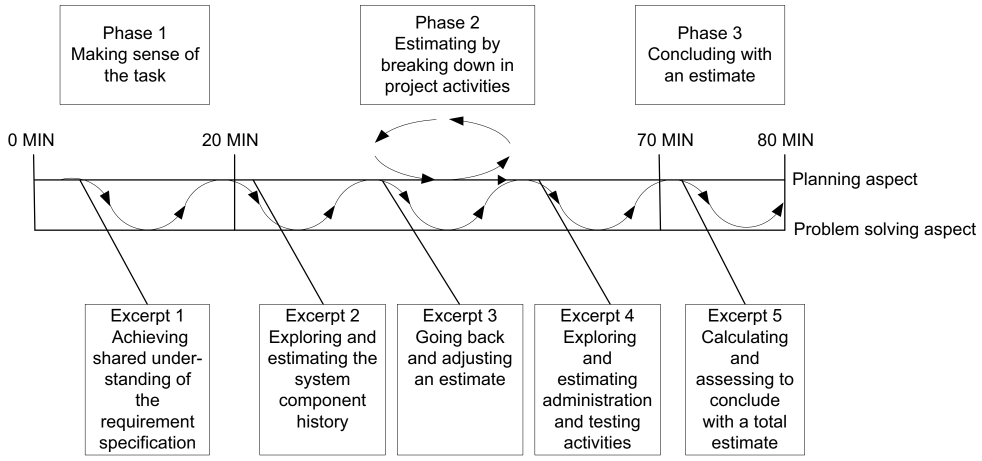

Phase 1
Making sense of the task
Phase 2
Estimating by breaking down in project activities
Phase 3
Concluding with an estimate
0 MIN
20 MIN
70 MIN
80 MIN
Planning aspect
Problem solving aspect
Excerpt 1
Achieving shared under-standing of the requirement specification
Excerpt 2
Exploring and estimating the system component history
Excerpt 3
Going back and adjusting an estimate
Excerpt 4
Exploring and estimating administration and testing activities
Excerpt 5
Calculating and assessing to conclude with a total estimate

Figure 3: Visualization of the team's interactional pattern and the oscillation between the planning and problem solving aspect

To understand the conditions for such collective accomplishment, we need to understand the role of the available cultural tools. Sharing a professional language allowed the team to collectively engage in elaborations and clarifications in a mutually constitutive process where these aspects of the sense-making process both inform and trigger each other. For instance, it allowed the team members to add important information necessary to solve the given tasks, as shown in the analysis of Excerpt 1. At the same time, the team members were experts in different areas of software development and displayed somewhat different interpretations of the technological concepts pertaining to this activity. To accomplish the tasks at hand they needed to align themselves constantly with each other's knowledge and perspectives and at the same time keep an overview of the whole estimation process. Our findings thus relate to Eklund et al. (2010), in that the capacity to coordinate information and articulate a minimum of shared understanding is crucial for the capacity to mobilize expertise and take the estimation process forward. While Eklund et al. (2010) investigated these aspects in the context of shift meetings in a 24/7 support organization, our analysis reveals that such "gap closing" is also a continual challenge in ongoing planning activities, and that its accomplishment rests on practitioners' capabilities to identify and explore problems by elaborating, specifying, and positioning themselves in flexible ways through the planning discussions.

The mechanisms of alignment cannot be understood, however, without taking into account the significant and shifting roles of the requirement specification. In general, material artefacts can serve several functions in communicative practices. They can be objects that evolve through elaborative and specifying practices; they can serve as an expressive medium through which other tasks and practices are realised; and they can be the indexical ground to which other problems and concepts are referred and described (Østerlund 2008). Our analysis confirms this multiple functionality and gives specificity to the argument by revealing how the shifting positions of the requirement specification in the activity both resulted from ongoing interactional accomplishment and served to take the interactional process forward. The requirement specification allowed team members to align their different forms of expertise and to collectively explore different facets of the activities. It also provided the team with issues to be resolved as well as with a sequential structure for engaging with these issues. Furthermore, it served as an area of exploration in itself and contributed to generating new questions as different problems were accomplished. In a wider perspective, a requirement specification incorporates established and historically generated knowledge, which were utilised as resources by the team in their concrete work. In this way, the team's engagement with the requirement specification both gives specificity to and contributes to the continuation of social practices in software development.

By revealing the details of the interactional and communicative practice of software teams in estimation work, this study also contributes insights for theorizing social practices more generally. First, practice theories still tend to emphasise the recurrent and routine-based elements of practice, which are understood with reference to the past, both in the context of historically constituted and emergent activities (Østerlund and Carlile 2005). In today's complex and knowledge-intensive society, however, we may argue that this approach is not sufficient. More social practices are geared towards a changing future and require models of performance other than those available from the past. As pointed out by, among others, Knorr Cetina (2001, p. 175), "creative and constructive practice—the kind of practice that obtains when we confront non routine problems—is internally more differentiated than current conceptions of practice as skill or habitual task performance suggest". As a planning activity oriented towards future scenarios, software effort estimation represents a form of creative and constructive practice. The findings of the present study suggest that practice theories need to account for movements back and forth between time frames, for instance, between making sense of given information and exploring imagined scenarios. Moreover, we suggest that a vocabulary that distinguishes between different forms of elaboration and clarification in explorative practices is helpful in this regard and should be tried out in other fields of practice.

Next, further advancement of practice theories depends on empirical contributions from studies conducted at a range of analytical levels. At the same time, however, practice-oriented researchers often fail to articulate the specific aspects of practice they highlight (Østerlund and Carlile 2005). This paper contributes to the understanding of communicative practices in teams within a specific domain of practice—namely, estimation of work efforts in software development. By applying an interactional perspective on team discussions within a short timeframe, the paper shows the complexity of such a practice and how it is accomplished by a series of tool-mediated, explorative, and sense-making actions, rather than by applying assumed information or routines. In this way, it shows how social practices are simultaneously historically constituted and emergent, and that these dimensions come together in productive ways at the level of social interaction.

7 Implications for research and practice

This paper contributes to the research on software effort estimation by revealing and specifying the details of estimation practice in multi-specialist teams. Moreover, by exploring the micro-dynamics of the practice in one team we have highlighted the distributed character of expertise in this field of work. Rather than residing within the cognitive structures of individuals, the expertise employed to resolve the estimation task is located and realised in the social interaction among team members and their material resources. Two issues emerge as important areas for further research and work. First, the significant role of the requirement specification calls for a greater attention towards this artefact in efforts at researching and supporting estimation practice. Although this has been addressed in the wider context of software development (e.g., Rönkkö et al. 2005), research on the use of requirement specifications for estimation purposes is sparse. Further research on this issue may reveal important aspects as to how estimation practice can be supported and improved. Second, the shifts between the planning aspect and the problem aspect of the activity described in this study shows that aligning different positions and perspectives in concrete problem solving is a critical issue. As a consequence, we will argue that relying upon selected team members' sharing of previously attained knowledge across distributed teams as described by (Boden and Avram 2009), or their abilities to take the perspective of other specialists in the work process (Martin et al. 2008), is not sufficient for solving estimation tasks. Instead, these processes require extensive communicative work to be opened up and dealt with in a collaborative manner. These dimensions need to be highlighted in future research and efforts to support the estimation work of software professionals.

To reveal the collaborative work of software effort estimation and make it accessible for analysis, a methodological approach is needed that accounts for the communicative and interactional dimensions of an activity and that is sensitive to how expertise is distributed and made relevant in ongoing practice. The method applied in this paper provides significant opportunities for improving our understanding of estimation practice by opening up the collective sense-making process as it emerges in moment-to-moment interaction. Moreover, it provides insights into how collective achievements in estimation work relate to available cultural tools and to infrastructures of knowledge. In our case, the interactional approach allowed us to recognize that software effort estimation cannot be regarded as a one-size-fits-all method. Rather, it is a complex and collective problem-solving activity that needs to be fine-tuned for each software project that is estimated.

Finally, the complexity of the practice examined in this paper calls for more context-specific studies of work and learning in different estimation teams. Our analysis focuses on only one team, selected because of the richness of their interaction and by their representation of the common problem of being over-optimistic when estimating. More studies of teams in different professional contexts are needed, and the increased use of group work in the industry (Haugen 2007) makes it even more important to uncover and investigate team processes in estimation as such. Since estimation is a core task in planning, monitoring, and controlling software processes, being able to provide realistic estimates is important. Examining this practice by means of a variety of types of studies and research approaches paves the way for a more comprehensive understanding of software effort estimation as a complex form of work. This, in turn, opens up possibilities for endeavours of estimating work efforts in software development projects.

8 Notes

1. The original material is in Norwegian, but for the purpose of this paper we have translated the interactions between the team members into English. Transcriptions were made consecutively and the conventions used can be found in appendix B. As emphasised by Jordan and Henderson (1995), there are several ways of transcribing verbal and non-verbal data; the level of elaboration and details to be included in the transcripts should correspond to the researchers' analytical interest. For our purpose, we have included information about non-verbal behaviour that we regarded as relevant for the way the interaction proceeded. In the excerpts, names of people, projects, and customers have been removed to ensure the anonymity of participants.
2. Figure 2 has been directly translated into English from Norwegian. No attempts have been made to correct or clarify issues beyond what was actually included in the requirement specification when translating and drawing Figure 2.

9 Acknowledgements

We would like to thank Professor Sten Ludvigsen, Professor Magne Jørgensen, the participants in the LiKE project at the University of Oslo, the anonymous reviewers, and the editor for their very helpful comments and suggestions on earlier versions of this paper.

10 References

Al-Ani, B., Ripley, R., Sarma, A., Hoek, A. v. d., and Redmiles, D., (2008). Continuous Coordination within the Context of Cooperative and Human Aspects of Software Engineering. In: *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, CHASE'08*, ACM, Leipzig.

Aranda, J., and Easterbrook, S., (2005). Anchoring and Adjustment in Software Estimation. *Software Engineering Notes*, (30:5): 346-355.

Asmuss, B., and Svennevig, J., (2009). Meeting Talk: an Introduction. *Journal of Business Communication*, (46:3): 3-22.

Barnes, B., (2001). Practice as collective action. In: *The Practice Turn in Contemporary Theory*, T. R. Schatzki, K. Knorr Cetina and E. v. Savigny, eds., Routledge, London, pp. 17-29.

Barthelmess, P., and Anderson, K. M., (2002). A View of Software Development Environments Based on Activity Theory. *Computer Supported Cooperative Work*, (11): 13-37.

Boden, A., and Avram, G. (2009). Bridging Knowledge Distribution—The Role of Knowledge Brokers in Distributed Software Development Teams. In: *Proceedings of the 2009 international workshop on Cooperative and human aspects of software engineering, CHASE'09*, IEEE, Vancouver, Canada.

Boehm, B., Abts, C., and Chulani, S., (2000). Software development cost estimation approaches—A survey. *Annals of Software Engineering*, (10): 177-205.

Bratthall, L., Arisholm, E., and Jørgensen, M. (2001). Program understanding behavior during estimation of enhancement effort on small Java programs. In *Product Focused Software Process Improvement,* F. Bomarius and S. Komi-Sirvio, eds., ABB Corporate Res., Kaiserslautern, pp. 356-370.

Brown, J. S., and Duguid, P., (2001). Knowledge and Organization: A Social-Practice Perspective. *Organization Science*, (12): 198-213.

Cherry, S., and Robillard, P. N. (2009). Audio-video recording of ad hoc software development team interactions. In: *Proceedings of the 2009 international workshop on Cooperative and human aspects of software engineering, CHASE'09*, IEEE, Vancouver, Canada.

Dittrich, Y., John, M., Singer, J., and Tessem, B., (2007). Editorial for the special issue on qualitative engineering research. *Information and software technology*, (49): 531-539.

Eklund, A.-C., Mäkitalo, Å., and Säljö, R., (2010). Noticing the past to manage the future: On the organization of shared knowing in IT helpdesks. In: *Learning across sites. New tools, infrastructures and practices.*, S. Ludvigsen, A. Lund, I. Rasmussen and R. Säljö, eds., Pergamon Press, Oxford.

Engeström, Y., (1992). *Interactive Expertise. Studies in distributed working intelligence*, Dept. of Education, University of Helsinki.

Engeström, Y., (1999). Expansive visibilization of work: an activity-theoretical perspective. *Computer Supported Cooperative Work*, (8): 63-93.

Engeström, Y., Engeström, R., and Kärkkäinen, M., (1995). Polycontextuality and boundary crossing in expert cognition: learning and problem solving in complex work activities. *Learning and Instruction*, (5:4): 319-336.

Faraj, S., and Sproull, L., (2000). Coordinating expertise in software development teams. *Management science*, (46:12): 1554-1568.

Fuggetta, A., (2000). Software process: A roadmap. In: *The future of software engineering*, A. Finkelstein, ed., ACM, New York, pp. 25-34.

Furberg, A., and Ludvigsen, S., (2008). Students' Meaning-making of Socio-scientific Issues in Computer mediated Settings: Exploring learning through interaction trajectories. *International Journal of Science Education*, (30:13): 1775-1799.

Greeno, J. G., Collins, A. M., and Resnick, L. B., (1996). Cognitions and learning. In: *Handbook of educational psychology*, D. C. Berliner and R. C. Calfee, eds., McMillian, New York, pp. 15-46.

Grimstad, S., (2006). *Software Effort Estimation Error*, PhD thesis, University of Oslo.

Grimstad, S., Jørgensen, M., and Moløkken-Østvold, K., (2006). Software effort estimation terminology: The tower of Babel. *Journal of Information and Software Technology*, (48:4): 302-310.

Happel, H. J., Maalej, W., and Stojanovic, L. (2008). Team: Towards a software engineering semantic web. In: *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, CHASE'08*, ACM, Leipzig.

Haugen, N. C. (2007). Moderne systemutvikling og estimering [Modern system development and estimation] *Presentation held at Estimation seminar 24 October.* Retrieved from http://simula.no/research/engineering/projects/best/seminars/Estimation%20Seminar%20 24.10.2007.

Heemstra, F. J., and Kusters, R. J., (1991). Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems*, (1:4): 223-237.

Herbsleb, J. D., and Moitra, D., (2001). Global Software Development. *IEEE Software*, (18:2): 16-20.

Hihn, J., and Habib-Agahi, H. (1991). Cost estimation of software intensive projects: A survey of current practices. In: *International Conference on Software Engineering*, IEEE Comput. Soc. Press, Los Alamitos, pp. 276-287.

Holmes, J., and Stubbe, M., (2003). *Power and politeness in the workplace*, Longman, London.

Housley, W. (1999). Role as interactional device and resource in multidisciplinary team meetings. *Sociological Research Online*, 4(3). Retrieved from www.socresonline.org.uk/4/3/housley.html.

Hughes, R. T., (1996). Expert judgement as an estimating method. *Information and Software Technology*, (38:2): 67-75.

Jordan, B., and Henderson, A., (1995). Interaction Analysis: Foundations and Practice. *The Journal of the Learning Sciences*, (4:1): 39-103.

Jørgensen, M., (2004a). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, (70:1-2): 37-60.

Jørgensen, M., (2004b). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology*, (46:1): 3-16.

Jørgensen, M. (2005). The "Magic Step" of Judgment-Based Software Effort Estimation. In: *International Conference on Cognitive Economics,* New Bulgarian University, Sofia, Bulgaria, pp. 105-113.

Jørgensen, M., (2007). Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models. *International Journal of Forecasting*, (23): 449-462.

Jørgensen, M., and Grimstad, S. (2005). Over-optimism in Software Development Projects: "The winner's curse". In: *Proceedings of IEEE CONIELECOMP*, IEEE Computer Society, Puebla, Mexico, pp. 280–285.

Jørgensen, M., and Grimstad, S., (2008). Avoiding Irrelevant and Misleading Information When Estimating Software Development Effort. *IEEE Software*, (May/June): 78-83.

Knorr Cetina, K., (2001). Objectual practice. In: *The Practice Turn in Contemporary Theory*, T. Schatzki, K. Knorr Cetina and E. v. Savigny, eds., Routledge, London, pp. 175-188.

Knuuttila, T., and Voutilainen, A., (2003). A parser as an epistemic artefact: a material view on models. *Philosophy of Science*, (70): 1484-1495.

Lantz-Andersson, A., Linderoth, J., and Säljö, R., (2008). What's the problem? Meaning making and learning to do mathematical word problems in the context of digital tools. *Instructional Science*.

Lee, Y. J., and Roth, W. M., (2008). How Activity Systems Evolve: Making Saving Salmon in British Columbia. *Mind, Culture and Activity*, (15:4): 296-321.

Lemke, J., (2000). Across the scales of time. Artifacts, activities, and meanings in ecosocial systems. *Mind, Culture and Activity* (7:4): 273-290.

Lynch, M., (2001). Ethnomethodology and the logic of practice. In: *The Practice Turn in Contemporary Theory*, T. Schatzki, K. Knorr Cetina and E. von Savigny, eds., Routledge, London, pp. 131-148.

Martin, D., Rooksby, J., Rouncefield, M., and Sommerville, I., (2008). Cooperative work in software testing. In: *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, ACM, Leipzig.

Mockus, A., and Herbsleb, J. D., (2001). Challenges of Global Software Development. In: *Seventh International Software Metrics Symposium (METRICS'01)*, IEEE Computer Society, London.

Moløkken-Østvold, K., and Jørgensen, M., (2003). A review of surveys on software effort estimation. In: *International Symposium on Empirical Software Engineering (ISESE 2003),* IEEE Computer Society, Rome, pp. 223-230.

Moløkken-Østvold, K., and Jørgensen, M., (2004). Group Processes in Software Effort Estimation. *Empirical Software Engineering*, (9:4): 315-334.

Orlikowski, W. J., (2006). Material Knowing: The Scaffolding of Human Knowledgeability. *European Journal of Information Systems*, (15): 460-466.

Østerlund, C., (2008). The materiality of communicative practices. *Scandinavian Journal of Information Systems*, (20:1): 7-40.

Østerlund, C., and Carlile, P., R., (2005). Relations in Practice: Sorting Through Practice Theories on Knowledge Sharing in Complex Organizations. *The Information Society*, (21:2): 91-107.

Panjer, L. D., Damian, D., and Storey, M.-A., (2008). Cooperation and coordination concerns in a distributed software development project. In: *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, CHASE'08,* ACM, Leipzig.

Rasmussen, I., (2005). *Project work and ICT : studying learning as participation trajectories*, PhD thesis, University of Oslo.

Robillard, P. N., (1999). The Role of Knowledge in Software Development. *Communications of the ACM*, (42:1): 87-92.

Rönkkö, K., Dittrich, Y., and Randall, D., (2005). When plans do not work out: How plans are used in software development projects. *Computer Supported Cooperative Work*, (14): 433-468.

Säljö, R., (2005). *Lärande och kulturella redskap. Om lärprocesser och det kollektiva minnet. [Learning and cultural tools. On processes of learning and collective remembering]*, Nordstedts Akademiska, Stockholm.

Sarkkinen, J., (2004). Examining a planning discourse: How a manager represents issues within a planning frame and how the others could do the same. In: *Participatory design conference*, ACM, Toronto, Canada.

Schatzki, T., (2001). Introduction: Practice theory. In: *The Practice Turn in Contemporary Theory*, T. Schatzki, K. Knorr Cetina and E. von Savigny, eds., London, Routledge, pp. 1-14.

Sommerville, I., (2001). *Software Engineering*, 6th ed., Addison-Wesley, Harlow.

Sommerville, I., (2007). *Software Engineering,* 8th ed., Pearson Education Limited.

Suchman, L., and Trigg, R. H., (1996). Artificial intelligence as craft work. In: *Understanding practice. Perspectives on activity and context*, S. Chaiklin and J. Lave, eds., Cambridge University Press, Cambridge, pp. 144-178.

Taff, L. M., Borchering, J. W., and Hudgins, J. W. R., (1991). Estimeetings: development estimates and a front-end process for a large project. *IEEE Transactions on Software Engineering*, (17:8): 839-849.

Vygotsky, L., (1978). *Mind in society. The development of higher psychological processes*, Harvard University Press, Cambridge, MA, .

Wertsch, J., (1991). *Vocies of the Mind. A Sociocultural Approach to Mediated Action*, Harvard University Press, Cambridge, MA.

Wertsch, J., (1998). *Mind as action*, New York, Oxford University Press.

Whitehead, J., (2007). Future of software engineering: A roadmap. In: *FOSE'07*, IEEE, pp. 214-225.

Ye, Y., (2006). Supporting software development as knowledge-intensive and collaborative activity. In: *WISER'06*, ACM, Shanghai, China.

11 Appendix A: The estimation process instructions for bottom-up estimation

Experimental setup: The participants sat in a meeting room at their own company's premises. Each team member had individually prepared for the estimation task by reading the estimation process instructions and the requirement specification. The teams then began the discussions in which they agreed upon an estimate of the software project.

The following instructions for how to employ a bottom-up estimation strategy, adapted from Jørgensen (2004b) were handed out in the beginning of the study to the different teams.

Estimation strategy. Use a 'bottom-up' estimation process, i.e. an estimation process based on a break-down of the project in activities and estimation of these activities individually. Use the work break-down structure described below. You are allowed to further detail the break-down structure.

Instruction for the estimation task. Description of the estimation context: Your company has already got the contract of developing the software described in, the requirement specification not included in this paper of confidentiality reasons. The task of your estimation team is to estimate the effort for the purpose of the planning of the project.

Most of the analysis phase is already completed and shall not be included in the estimate. In addition to the most likely effort, you are supposed to provide the minimum (best case) and maximum (worst case) effort, and the probability that the actual effort will be between the minimum and the maximum effort.

Example: You believe that the most likely use of effort for a project is X work-hours, that the minimum effort is as little as Y work-hours, and that the maximum effort is Z work-hours. You estimate that it is P percent likely that the actual effort is between Y and Z: You do not know who the project members will be. Assume that the participants are normally skilled employees of your company.

Work break-down structures. The following work breakdown structure was given out as a guide for dividing the project into different project activities. If necessary the teams were allowed to break the activities further down. The different project activities are listed sequentially as they would be performed if the project were developed; each matches items in the breakdown structure used in most of the company's projects.

1. Administration
2. Meetings
3. Analysis (not already completed)
4. Design
5. Programming
6. Data base work

7. Test
8. Documentation
9. Installation/system integration

12 Appendix B: Transcription conventions

The following transcription conventions were used in the data excerpts:

[…..] xx min, xx sec	Indicating that a timed part of the interaction is not included
(3,0)	Indicating timed pause
[ ]	Indicating overlapping talk

Article II

Børte, K. (submitted). Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation.

Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation

Kristin Børte [1&2]

[1]Simula Research Laboratory, P.O. Box 134, 1325 Lysaker; [2]University of Oslo, Department of Educational Research, P.O. Box 1092 Blindern, 0317 Oslo

Abstract: Making use of historical information from databases and previous experiences when solving present working tasks is a complex issue. Several work places develop databases and archives to make historical information accessible, however, many also experience difficulties in utilizing such information in productive ways. Software effort estimation is an interesting case in this respect. This paper examines how teams of software professionals go about to use historical information, and what challenges they face, when applying an analogy-based, top-down estimation approach. By employing a sociocultural perspective on knowledge and communication, the paper analyzes the collaborative process through which two teams achieved an effort estimate. The findings explicate the kind of challenges encountered by the teams, how they occurred and how they were dealt with. To be able to use historical information the teams needed to establish shared understanding for conducting the work. This was achieved by exploring details of both previous projects and the new project. Furthermore, the teams needed to articulate potential meanings of the historical information and create boundary concepts to be able to use knowledge across boundaries. The results indicate that the idea of analogy-based, top-down estimation does not take sufficiently into account that knowledge needs to be recontextualized to become meaningful in new situations. To understand and facilitate these processes in estimation work, more research is required that accounts for the communicative and collaborative dimensions of this activity.

Keywords: top-down estimation, collaborative work, recontextualization processes, collective remembering and boundary concepts.

1 Introduction

In today's working life, drawing on historical information when solving tasks is a way of making solid, well-grounded decisions. In many professions, much time is devoted to developing databases, repositories, and archives that store information so that it can be utilized in the future for solving related tasks. Making use of historical information is not only about using databases or archival information; it also concerns making use of previous knowledge and experiences that people have acquired over the years, through both working life and education. Moreover, when solving specific tasks at work people often collaborate and work together in teams to achieve a solution. To understand how teams make use of previous knowledge and experiences in their collaborative problem solving a focus on the work process as such is needed. The present paper is a contribution towards this, by investigating software effort estimation work, with specific attention given to the challenges software professionals face when utilizing historical information in present working tasks. A focus on understanding cooperative activities in software development have also been emphasized by researchers in the area of computer supported cooperative work (Dittrich, Randall, & Singer, 2009).

The empirical focus in this paper is on how teams of software professionals go about to estimate the work effort needed to develop a software system. This task is an important and recurrent activity in software development projects (Boehm, Abts, & Chulani, 2000; Kjærgaard, Nielsen, & Kautz, 2010; Sommerville, 2007). When planning and developing software systems, effort estimates are used for purposes such as budgeting, trade-off and risk analysis, project planning, control and software improvement investment analysis (Boehm et al., 2000). Moreover, software effort estimation is often organized as teamwork in the industry and some estimation approaches specifically requests the use of historical information to achieve an effort estimate of a new software project. This makes software effort estimation an interesting case to investigate with regards to how practitioners utilize historical information in present working tasks and the challenges they face.

Providing realistic effort estimates has proven to be a difficult task in the software industry. A review of surveys on software effort estimation reported that 70 to 80% of software projects overrun their estimates on average by spending 30 to 40% more effort than estimated (Moløkken-Østvold & Jørgensen, 2003). The consequences of using inaccurate estimates can be severe, leading to large financial losses, lost contracts, delays or low quality software for the involved company. Thus, insight in how estimation work is done and can be supported is important for professionals as well as project management in software development.

The most frequently used estimation method in the software industry is called judgment-based estimation (Bratthall, Arisholm, & Jørgensen, 2001; Heemstra & Kusters, 1991; Hihn & Habib-Agahi, 1991; Jørgensen, 2004a). Being judgment-based means that the quantification step is achieved by the decisions of experts rather than by a mathematical algorithm (Jørgensen, 2007). When a judgment-based estimation method is employed, those doing the estimation can either follow a top-down approach or a bottom-up approach. The main distinction between these two approaches is that when following a bottom-up approach the work is usually divided into different project activities before the effort of each activity is estimated. These estimates are then summed up to form a total estimate of the whole software development project. In a top-down approach, on the other hand, the total effort of a software project is estimated without breaking the system down into different project activities. Instead, the estimators look for similar previously completed software projects to compare with the current project and adjust for differences before a total effort estimate is agreed upon (Heemstra, 1992; Sommerville, 2007). This means that a top-down approach explicitly incorporates the use of historical information.

While there are advantages and disadvantages of both approaches, researchers have argued that a top-down approach is more efficient than a bottom-up approach because it is less time consuming and thereby cheaper to apply (Boehm, 1984). It can be applied without much knowledge of how to build software (Jørgensen, 2004b; Moløkken-Østvold & Jørgensen, 2005), and it may reduce the bias towards over-optimism, due to the greater use of historical data from previous projects (Moløkken-Østvold & Jørgensen, 2005). However, not much research has been done to investigate how software professionals actually use historical information in estimation work and the challenges they face.

This paper investigates in depth how teams of software professionals utilize historical information in present working tasks when employing an analogy-based top-down estimation approach. The aim is to reveal critical instances of the work process and details of the challenges that are faced in this work. The analytical focus is on the communicative and interactional work the teams need to do to collaboratively achieve an estimate.

The remainder of the paper is organized as follows. First is a background on how analogies have been utilized in software effort estimation provided. Then, in section 3, a theoretical perspective that takes social action and knowledge as situated in practice is presented, while section 4 describes the data and the analytical strategy that were used. This is followed by the data analysis, which reveals how challenges of using historical information occur and are dealt with in the collaborative work of achieving an effort estimate. In section 6 the findings are discussed in relation to previous research and their contribution to the field of research on software effort estimation. The paper concludes by pointing to some implications for research on software effort estimation.

2 Analogies and software effort estimation

Software effort estimation is a specific type of task that is conducted as part of planning and developing software systems. In the industry this work is often organised and conducted in teams of software professionals (Haugen, 2007). This is due to the multi-faceted character of the estimation task where input from specialists within different areas of software development, such as programming, databases, architecture, and project management are needed. The common point of departure for conducting the estimation work is usually a requirement specification describing the details of the project to be developed (Sommerville, 2001). This specification needs to be interpreted to achieve an understanding of the kind of system that is going to be developed. To estimate the work effort that is needed for developing software systems, a number of different estimation approaches and models have been developed. Some of these approaches incorporate the use of historical information by for example proposing the use of analogies.

The idea of using analogies is rooted in a cognitive tradition and is a widely researched phenomenon in cognitive science. According to the cognitive science perspective, analogical thinking involves the ‘ability to think about relational patterns’ (Gentner, Holyoak, & Kokinov, 2001, p. 2). Inferring such relational patterns relies on mapping the salient features of one situation (the source) to the situation for which one wishes to make inferences (the target) (Holyoak, 2005). Thus the cognitive science perspective assumes the existence of a shared complex representational structure in which people mentally represent knowledge in schemas that enable such mappings (Gentner, 1983). In addition to this mapping, analogical thinking comprises several basic constituent processes, such as accessing possible sources (analogues), making inferences about the target, adapting the inferences to the current situation and learning in terms of generating new knowledge (Gentner et al., 2001). Following the cognitive science perspective, when for example solving a problem by analogy, one first needs to access one or more instances in long-term memory that appear to be relevant analogues to the current problem. Thereafter, one has to map systematic correspondences between features belonging to the chosen analogue and the target problem at hand. This mapping provides a base for making inferences about the problem that may then be used to solve it in the current situation, leading to the creation of new knowledge (Gentner et al., 2001; Holyoak, 2005).

In the research field of software effort estimation this has taken form as a means to utilize historical data from completed software projects (Keung, 2009). The use of analogies has been incorporated in different estimation approaches, one of which is called estimation by analogy (Boehm, 1981). Researchers studying this approach have focused on formalising and developing software tools that can measure analogies between different software projects (Mair, Martincova, & Shepperd, 2009). The results from using such tools, ANGEL (Shepperd, Shofield, & Kitchenham, 1996) being one, have shown to lack consistency when it comes to accuracy in predicting estimates compared to other estimation methods (Mair & Shepperd, 2005).

Walkerden and Jeffery (1999) compared the performance of software tools and human judgment in finding analogies. They found that people outperformed software tools in identifying analogies between software projects. Also a recent literature review about expert problem solving related to estimation and the use of analogies, proposes that software tools do not incorporate the complex problem solving that human judgments involve (Mair et al., 2009). As these studies show, identifying analogies is not a straightforward process that can be easily automated and made algorithmic for software effort estimation.

Furthermore, the studies referred to above relate to formal estimation models. In judgment-based estimation, finding analogies depends upon people's capacity to remember. This can be done with or without the help of supporting tools (Jørgensen, Indahl, & Sjøberg, 2003). One judgment-based estimation approach that explicitly incorporates the use of historical data from earlier software projects is the analogy-based, top-down approach studied in this paper. This estimation approach proposes an idealized model of how to achieve an estimate by using historical data from previously completed projects that are identified by analogies (see Figure 1). The model proposes a structured line of work, where the software project is seen as a whole, in the sense that one does not investigate the requirement specification in detail or estimate activities of the project separately. Instead, an assessment of size and complexity, for instance, is formed after reaching an overall understanding of the requirement specification. Then the effort is estimated by taking the total effort expended on one or more similar previously completed projects as the point of departure. The previously completed projects have been identified based on analogies between the project to be estimated and completed ones. A comparison is then conducted for the purpose of making inferences about an estimate on grounds of analogies. Thereafter, differences between the projects are adjusted for, before a total effort estimate is agreed upon. Finally the total estimate is distributed across the different project activities (Heemstra, 1992; Jørgensen, 2004b). For further details of how to employ this top-down approach, see appendix A.

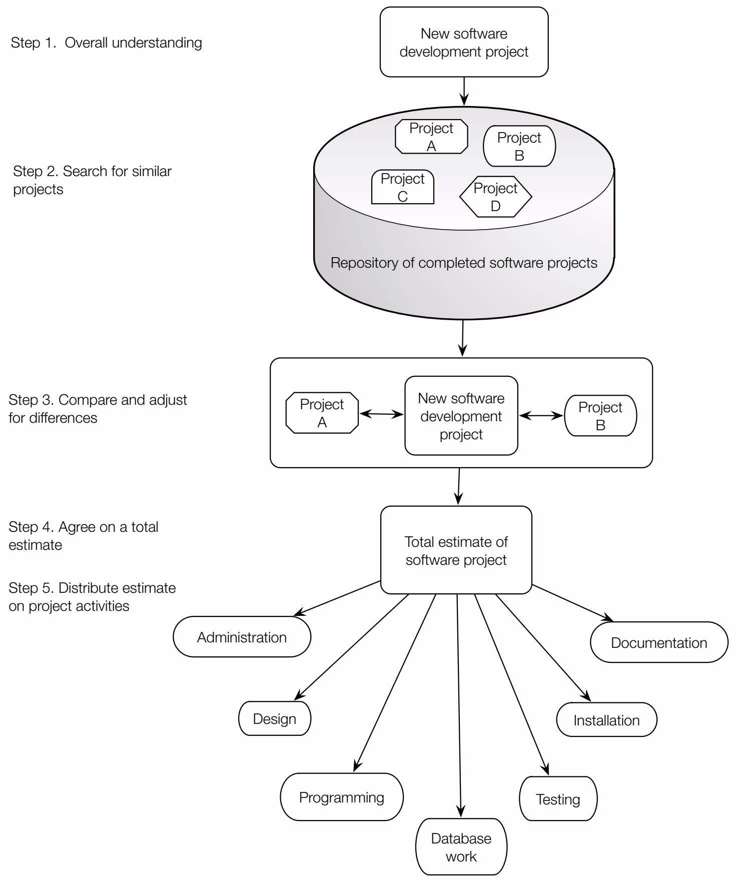

Step 1. Overall understanding
New software development project
Step 2. Search for similar projects
Project A
Project B
Project C
Project D
Repository of completed software projects
Step 3. Compare and adjust for differences
Project A
New software development project
Project B
Step 4. Agree on a total estimate
Total estimate of software project
Step 5. Distribute estimate on project activities
Administration
Documentation
Design
Installation
Programming
Testing
Database work

Figure 1. Visualization of the idealized estimation approach investigated in this paper.

Across the different estimation approaches, research has shown, that deciding on an estimate is difficult. One strand of research have found that factors such as anchoring, over-optimism, irrelevant information and wishful thinking can influence the decision software professionals take (Grimstad & Jørgensen, 2007; Jørgensen & Carelius, 2004; Jørgensen & Grimstad, 2005; Jørgensen & Grimstad, 2008). Lack of relevant experience, along with incomplete available information, also makes the task difficult. A study that compared the accuracy of two different estimation approaches—the analogy-based top-down approach studied in this paper and a bottom-up approach—found that the top-down approach only led to accurate estimates when very similar previously completed projects were recalled (Jørgensen, 2004b). Of the seven estimation teams in that study, only four were able to find similar previously completed projects and out of those four only two teams were able to benefit from the completed projects in their estimation work.

Another study that contributes to the understanding of how people make judgments when planning development projects was conducted by Busby and Payne (1999). They identified what was called 'suspect strategies,' i.e., practices that seemed unsuitable to the task in some way. Busby and Payne found that people relied too much on anchor projects when estimating, that tasks where decomposed instead of using past experiences on a general level and that, according to the authors view, the uniqueness of the tasks where overrated so that earlier experiences where not regarded as relevant.

Although the use of analogies in estimation has been subject to scrutiny in the literature, which has revealed that there are difficulties connected to both identifying and using historical data, few studies have looked at what causes these difficulties and how they are dealt with. Previous research on estimation has in large part focused on individual reasoning and decision-making, but the collaborative work involved in using historical information in estimation work is not sufficiently described. To investigate the collaborative processes and how they unfold, the communicative work conducted when estimating a software project needs to be opened up and explored (Cohn, Sim, & Lee, 2009). In other words, the details of what kind of work teams of software professionals do when using historical information from previously completed software projects needs to be examined. This paper addresses this gap by studying how teams of software professionals employ one specific top-down estimation approach. The following research questions are addressed:

1. What challenges do teams of software professionals face when applying an analogy-based top-down estimation approach and how do they occur?
2. What kind of collaborative work is needed to utilize knowledge from former software projects when estimating new ones?

This analytical focus on the collaborative aspects of estimation work has only been employed in estimation research to a minor extent. Investigating the collaborative work when teams employ an analogy-based top-down estimation approach might explicate important aspects of how teams of software professionals utilize historical information when estimating a new software project. In this paper the use of historical information is understood as a question of utilising previous knowledge and experiences.

3 A sociocultural perspective on knowledge and collaborative work

To open up and investigate the collaborative processes of the team's work in depth, a theoretical perspective that takes social action as point of departure and allows for studying interactional processes is needed. This paper therefore employs a sociocultural perspective on knowledge and collaboration.

In the sociocultural perspective, knowledge and collaboration is understood as embedded in the interaction between participants in a practice and the tools and artefacts they use (Greeno, Collins, & Resnick, 1996). In software effort estimation work, this means that the knowledge and experiences participants bring into the team, together with the available artefacts used, such as the requirement specification, form a context in which a particular problem is solved and knowledge is made relevant through communicative work. Furthermore, this perspective assumes a close link between knowledge and the social practice in which it is acquired (Säljö, 2001). Because of this close link, knowledge needs to be articulated, made relevant and adapted to be utilized in other contexts.

When articulating and making meaning of knowledge in different contexts, the distinction between meaning and meaning potential is important. Meanings are generated in communication and '…are properties of situations, utterances, contributions to interaction situated cognitive events, etc.' (Linell, 2009, p. 235). Sense-making is when people interpret something a certain way at a certain time. Hence, it concerns what is meant and made known in a particular setting. The notion of meaning potential has been explicated by Linell (2009) as semantic resources that together with contextual factors are used to achieve situated meaning. Meaning potential can thereby be conceived as that which moves between contexts and thus inhabits both history and structure. Furthermore, meaning potentials are multiple, in the sense that there is not one meaning for a concept, an artefact or a tool. It consists of a 'set of properties which together with contextual factors ...make possible all the usages and interpretations of the word or construction that language users find reasonably correct, or plainly reasonable in the actual situations of use' (Linell, 2007, p. 389). The meaning potential, therefore, always needs to be articulated in a specific context for it to provide a meaning that is relevant. In software effort estimation work, meaning potentials are articulated by a team of software professionals and are thus realized through various fields of expertise (Engeström, 1992).

In relation to meaning making, the process of making knowledge and experiences relevant and adapted to other contexts is important. This process is called recontextualization and is conducted through articulation and communicative work in which knowledge is shared, elaborated and clarified amongst the team members. Linell defines recontextualization as 'the dynamic transfer-and-transformation of something from one discourse to another' (1998b, p. 145). This means that some parts or aspects from one context are adapted or transformed to fit a different context. These parts or aspects can, for example, be knowledge, arguments, facts or different ways of seeing, thinking and acting. Through this recontextualization process, these parts are often subject to change because 'recontextualisation is never a pure transfer of a fixed meaning' (Linell, 1998a, p. 155). Moreover, the degree of adapting and adjusting knowledge through recontextualization might also vary. The three concepts that Carlile (2004) articulates — transfer, translation and transformation — provide an opportunity to distinguish between different levels of recontextualization that might be useful when investigating the use of historical information.

Another important aspect related to recontextualization is the act of remembering. Knowledge and experiences needs to be remembered in order to be recontextualized. Remembering is here understood as a social activity, an accomplishment which occurs through conversations with others in a specific context (Middleton & Brown, 2005; Middleton & Edwards, 1990). In the top-down estimation approach studied in this paper, the use of historical information from similar completed software projects constitutes an essential part of the work process. The finding of similar projects is proposed through either the act of remembering alone or by the support of databases containing distributed stored information about completed software projects. Such stored information will then function as a resource for remembering (Middleton, 1997). Even though information is standardised and stored in a decontextualized manner, it needs to be recontextualized before it can be used in another context (Ackerman & Halverson, 2004). A related term in this respect is boundary objects (Star & Griesemer, 1989) which contain shared information from intersecting worlds. Boundary objects work because they contain details that are understandable in both contexts, but also here ‘the information, if not supplied by the same individual, must be reunderstood for the user’s current purpose’ (Ackerman & Halverson, 2004, p. 176). A variant of boundary objects, boundary factors, also describe the elements that are meaningful across borders and can be represented and displayed the same way in, for example, different hospital information systems (Bjørn, Burgoyne, Crompton, MacDonald, Pickering, & Munro, 2009). Boundary objects can facilitate the use of knowledge from different contexts, but the knowledge still needs to be recontextualized.

When teams of software professionals employ the analogy-based, top-down estimation approach investigated in this paper, it is expected that the teams utilize knowledge and experiences from earlier projects on a general level. How this work is conducted in teams of software professionals is thus a core interest in this paper. In the following section the data material available for analysis is presented together with the analytical approach taken.

4 Data material and analytical approach

The analysis presented in this paper is an expanded reading of videotaped data from a quasi-experimental software estimation study conducted in a Norwegian branch of an international IT-consultancy company in 2002. The authors of the study’s design pursued authenticity of the estimation process, and therefore real requirement specifications from customers were used as estimation tasks and software professionals from the given firm were hired as participants. A total of seven teams were organised by a senior manager in the company where the study was conducted. The senior manager ensured that the team members had sufficient estimation competence to perform realistic estimates of the software development projects. In order to resemble actual estimation teams, each team consisted of one project manager and one or two developers. The teams solved two estimation tasks where two different estimation approaches, top-down and bottom-up, were employed. The entire corpus of data have previously been analysed and reported in Jørgensen (2004b) and an in-depth study of the bottom-up estimation approach has been reported in Børte and Nerland (2010).

The empirical setting was as follows: The participants sat in a meeting room on their own company's premises. To solve the estimation task, they were provided with a support structure consisting of a computer with access to the company's online database of previously completed projects, and a phone, so calling informants was possible. As preparation for the task, the participants each took 30 minutes to read and understand the requirement specification, the estimation task and the estimation process instructions describing the estimation approach to apply. The estimation task and the process instructions can be found in appendix A.

The requirement specification served as the main source of information about the task, while the support structure together with the knowledge and experience embedded in the team served as additional information sources for finding previously completed projects. After the preparations the teams began the discussions in which they would agree upon an effort estimate of the software project. These discussions were recorded on video, which constitutes the data material available for further analysis. The participants were asked to employ a specific variant of the top-down estimation approach, i.e., analogy-based, in which an estimate should be achieved through a comparison with similar previously completed projects. In addition they where told that it was not necessary to distribute the estimate on to the different project activities. Hence, the experimental setup was designed to investigate steps 1 to 4 in the top-down estimation approach, as visualised in Figure 1.

Even though the purpose of this estimation study was to investigate the accuracy of two different estimation approaches, the setup provided opportunities for focused examinations of the collaborative work in software effort estimation. Such collaboration is difficult to explore as software development projects and estimation work evolve over long periods of time. Moreover, this study also allowed for examining the interactional and communicative work conducted when particular estimation approaches were employed. This would otherwise have been difficult to research, because the top-down estimation approach is not the estimation approach most commonly employed amongst software professionals (Moløkken-Østvold & Jørgensen, 2005).

To gain insight in the employment of the top-down estimation approach and thus how historical information is used, the interactional process of two different teams of software professionals were chosen. The selection of teams was based on the analysis of the entire corpus of data conducted by Jørgensen (2004b), together with an overall analysis of the seven estimation teams' top-down work process. The overall analysis revealed considerable variation in how teams approached, made sense of and solved the estimation task at hand. Furthermore, Jørgensen (2004b) reported that only four out of seven teams were able to find project analogies. Thus, for the purpose of this article, two teams were selected, one that found analogies and on that did not, to reflect the variety of how challenges were faced in their collaborative work and how these were dealt with in different ways. Common criteria for selecting the two teams were the richness of the interaction and the order of the estimation tasks, i.e., the top-down was the second task in the estimation study for both teams. By conducting an event sampling of the data from the selected teams' interactional work, it is possible to demonstrate some of the key challenges of this top-down approach. The selected teams A and B correspond to the teams 3 and team 2, respectively, in Jørgensen's study (2004b).

To explicate the collaborative process of making use of knowledge embedded in previously completed software projects an analytical approach that emphasise communication must be taken. Interaction analysis rests on a sociocultural notion and finds its basic data for theorising about knowledge and practice in the details of social interaction (Jordan & Henderson, 1995). The focus on the collaborative processes makes this an appropriate method for studying social interaction in teams. The interaction, as such thereby constitutes the unit of analysis. This way of analysing video data is 'an interdisciplinary method for empirical investigation of the interaction of human beings with each other and with objects in their environment' (Jordan & Henderson, 1995, p. 3).

Jørgensen (2004b) has previously identified the typical top-down estimation process the teams conducted as a repeated discussion sequence between topics related to searching for similar projects and topics related to understanding the requirement specification, context or previous projects. In between there were also instances of discussions on how to go about and estimate the task. Furthermore, Jørgensen also pointed to some challenges of employing this analogy-based top-down approach, such as problems with finding similar previously completed projects and using historical information in the estimation work (Jørgensen, 2004b; Jørgensen, 2005).

The analytical focus in this paper is on explicating the challenges of utilising historical information that the teams face, and investigate what causes them and how they are dealt with in the collaborative work. This is investigated by following the turns and analysing the data in relation to both context and processes. First a chronological analysis was conducted with focus on the content dimension of the interactional process in the two teams. This analysis was conducted through repeated viewings of the whole collaborative process in both teams, before the discussions were sorted in relation to content. Through this first reading, the main challenges the teams faced in their work were identified and then event samplings of data were conducted that followed the identified challenges. Thereafter, by following the micro-chronology, in-depth analysis of the different data excerpts were conducted and presented thematically. In the following analysis, one excerpt from each team illustrating the identified challenges will be presented. However, the last challenge is illustrated by one excerpt from team B, because this was the most relevant one to analyse in order to understand the challenge.

The main analytical concept used in the analysis is recontextualization. This analytical term characterizes the ways in which the teams are able to adapt and make meaning of past experiences and knowledge embedded in previously completed projects and brings this to bear on the estimation task. To open up the details of the collaborative interaction, the concepts of elaboration and clarification are used to capture the communicative work the teams conduct to achieve recontextualization.

The chosen excerpts are organised in columns to document the participants' talk and actions that are regarded as relevant for how the interactions proceeded. The transcripts have been translated from Norwegian into English for the purpose of this paper. To ensure anonymity of participants, the names of people, projects and customers have been removed. The transcript conventions used can be found in appendix B.

5 Challenges that occur in the collaborative process of applying a top-down estimation approach

After completing a chronological analysis, the same challenges that Jørgensen (2004b) pointed to were identified. In addition, a search for a pattern between the types of main challenges that occurred across the two teams was conducted. The main challenges identified are then used as a point of departure to understand the work of utilising historical information in software effort estimation. Figure 2 visualises, these challenges together with corresponding excerpts displaying the events that will be analysed in depth.

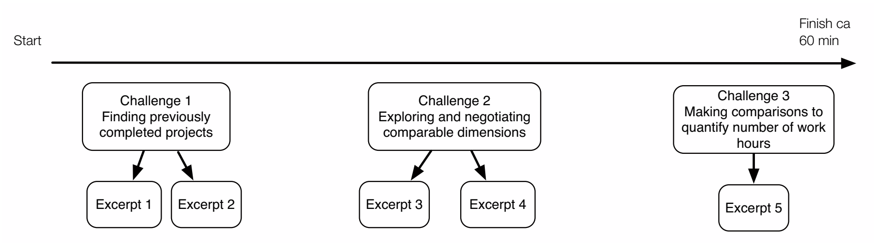

Start
Finish ca 60 min
Challenge 1 Finding previously completed projects
Challenge 2 Exploring and negotiating comparable dimensions
Challenge 3 Making comparisons to quantify number of work hours
Excerpt 1
Excerpt 2
Excerpt 3
Excerpt 4
Excerpt 5

Figure 2 Visualization of the challenges faced by the teams and the corresponding excerpts.

As Figure 2 shows, the first challenge the teams encountered was to find similar previously completed projects. The second challenge was to negotiate and explore comparable dimensions, and the third was to compare previous projects with the project to be estimated. Even though the teams faced the same types of challenges, rather different approaches were chosen to deal with these challenges and solve the estimation task.

In the following, episodes collected from two teams displaying the challenges of using historical information will be analysed in depth. The two teams consist of two and three team members, respectively. Team A consists of one experienced developer (D) and one project leader (PL). Team B consists of one developer (D), one database specialist (DB) and one project leader (PL). The main artefacts the teams had available were the requirement specification and a database containing distributed stored information about previously completed software projects.

5.1 Finding previously completed projects

The first challenge, to find similar previously completed projects for comparison, required the teams to open up and make relevant their own experiences so the searching could occur. In addition, the appropriate experiences from previously completed projects needed to be made relevant. Excerpts 1 and 2 illustrate how the two teams initiated the search for earlier completed projects. We enter the video-recorded process of team A in Excerpt 1 when the developer closes the discussion on how they should relate to the work process of prototyping and initiates searching for previously completed projects:

Excerpt 1

Turns	Verbal communication	Description of actions
1. D:	Yeah, that might well be. The point is just not to mix too much technology into the process, then. But we, I don't think we should discuss that. So what	
2. PL:	no	
3. D:	what I'm trying to get my head round is to try to find some projects that can be compared with this one, yeah, and that have actually been completed. (12,0)	
4. D:	They must exist, surely (12,2)	
5. PL:	Dunno, erm	
6. D:	Yeah (6,0)	*Both are looking in the requirement specification while talking.*
7. D:	Yeah, cos its relatively small then, so far as I can see, there's certainly a load of manual things here with (18,0)	
8. PL:	You need to have both import and also manual entry	
9. D:	Hmm	
10. PL:	And also you need reminder solutions, they shouldn't be automated so that makes it simple. So just one report on everything.	*Project leader is taking notes.*
11. D:	Hmm, 14 windows then?	
12. PL:	I wrote seven, it was 14	

A call for finding similar previously completed projects is initiated in line 3. Finding such projects appears to be limited to the act of remembering, which is shown in the utterance, 'What I'm trying to get my head round is to try to find some projects that can be compared with this one, yeah, and that have actually been completed' (line 3).

This utterance is followed by 12 seconds of silence, which is interrupted by a confirmatory utterance: ‘They must exist, surely’ (line 4). The confirmatory utterance functions as a suggestion in the interactional work and leaves the floor open for any of the team members to remember. Again a period of 12 seconds of silence follows. The silence is broken by the statement ‘dunno, erm’ (line 5), which indicates that finding previously completed projects by remembering alone is difficult. The utterance also closes down the initiative of remembering alone and shows that the team is unsuccessful at this point. The failure of not remembering previously completed projects initiate’s elaborations concerning the task. The elaboration that follows, in lines 7−12, is concerned with making meaning of the task at hand and opens up for engaging in joint remembering.

The object of the participants’ investigation is the requirement specification, where the elaborations take form as a walkthrough of the different components and functionality of the described system. During the elaborations, main features are articulated before being clarified (lines 10-12). The clarifications serve the function of grasping both the size and the complexity of the task, and in addition close down the elaborations, making it possible for the team to move on. The elaborations performed are mainly concerned with articulating and counting different parts of the system to be developed, thereby serving two purposes: first, achieving a shared understanding of the task and particularly the relevant parts for the purpose of remembering previously completed projects for comparison, and second, identifying and preparing possible points of connection between the task and previously completed projects. Possible connection points investigated were size (line 7), complexity (line 10) and number of windows (lines 11−12).

To decide upon which points of connection i.e., size, complexity, or number of windows, that can be utilized in the further work, meaning potentials need to be articulated. Articulating meaning potentials has to happen before the connection points can be created boundary concepts that can facilitate the bridging of knowledge between two projects. The meaning-making process, which is conducted through the elaborations, brings the team closer to jointly remembering appropriate experiences needed to solve this task. As this excerpt shows, the act of remembering is closely linked to team members' own experiences with former projects, where the relevance dimension serves as both a guide and as selection criteria.

Team B chose a different approach for finding similar, previously completed projects. We enter the interaction of team B in Excerpt 2 when the project leader closes the previous discussion by stating that it is a good idea to agree upon some assumptions:

Excerpt 2

Turns	Verbal communication	Description of actions
1. PL:	It's good to get some framework conditions in place like that, yeah.	
2. DB:	So what we then have to do, if we're going to work with, with lots and lots of prototyping now, so what I think now, based on the experiences I've had, is that then we have to create a data model, a rough data model. We just draw it up as we think it should be and then begin to, begin to look at which functions we feel are the most important for the customer. So we create the screens in it. It's a real shame that we have to use Visual Basic. If we were using something else, something that's quicker, and created the screens in it, then we could have just shown it to them and seen whether it was like that and then gone back.	*DB is looking in the requirement specifica-tion on a diagram.* *PL is taking notes. D is looking in her documents.*
3. D:	Hmm	
4. PL:	Now, there are certain customers who are a bit more difficult in that respect, right. If they have chosen a product then you can't then come in and say that...	
5. DB:	No, no, I agree with you, but in the prototyping it would have been much much quicker if we could have used products where you can just hack together the screens as you go. Erm, Another thing that's important, which we need to make a decision about, is how many iterations we should have with the customer.	
6. PL:	Hmm	
7. DB:	Three times is usual, or something like that.	
8. PL:	Yeah, three at most, yeah, or perhaps, many take up to five or something like that.	

In the elaborations and clarifications that follow the prompt to agree on some assumptions, imaginary scenarios are used as means to make experiences relevant (lines 2–8). This is conducted by imagining how to perform the development process, which is indicated by the utterance in line 2: '… so what I think now, based on the experiences I've had, is that then we have to create a data model, a rough data model'. Not only are team B imagined as the people who will conduct the work, indicated by the repeated term 'we', but also the creation of such scenarios makes it possible to articulate members' own expert knowledge. Through the process of creating imaginary scenarios, the team is able to identify those parts of the development process where they need to add information in the form of assumptions. 'Another thing that's important, which we need to make a decision about, is how many iterations we should have with the customer ' (line 5). The assumptions agreed upon (line 8) narrow down the task and establish a framework of the task for the team to work within as they continue.

For team A the main challenge at this point was connected to the act of remembering. As this could not be accomplished alone, the team elaborated on the task at hand to achieve a shared understanding so that they were able to engage in collective remembering. Team B, however, chose to narrow down and frame the task by agreeing upon a set of assumptions rather than engaging in remembering. By doing this the team contextualised the task so that imaginary scenarios could serve as a means to make meaning and achieve a shared understanding.

In these two excerpts, a recontextualization process is gradually taking form through the meaning-making performed and the shared understanding achieved. Because team A had to engage in collective remembering, they were not able, at this stage, to adapt and transfer knowledge from previous projects. As team B chose to contextualise the task by adding information in the form of assumptions, they were able to transfer experiences in the form of principles related to the development process. Hence, they were able to establish a shared point of departure for the conversation, making it possible to proceed.

5.2 Exploring and negotiating comparable dimensions

During the work process of the two teams, a second challenge arose: exploring comparable dimensions that could be used as bridges between the task and a previous project. Different approaches were taken once again, and we enter the data of team A in Excerpt 3 when the project leader suggests windows as a comparable dimension:

Excerpt 3

Turns	Verbal communication	Description of actions
1. PL:	We need to find a project with a, where the number of windows has been recorded, then.	
2. D:	Hmm.	
3. PL:	Well, it's a, it's probably the only measure we have here, then	
4. D:	Hmm	
5. PL:	Erm	
6. D:	It's probably good to try to have a certain, erm well, some sort of averagely simple one, if we're supposed to compare, cos are these projects, these windows complex.	
7. PL:	Question is whether we need to do that at this stage in the estimation. If we're only going to have one. But if you look at what, what it's about, yeah? You have a project initiation phase,	*Project leader is making a drawing.*
8. D:	hmm	
9. PL:	and then you have what we, let's call it a development phase, OK, then you have testing and installation. And if we have some fixed markups for the surrounding stuff,	
10. D:	hmm	
11. PL:	so if we had found a project, right, that had a lot of development then we could have taken the development part of that and scaled down. And run,	
12. D:	hmm	
13. PL:	run these. I think maybe some of these are a bit... You always need a test phase; you always need an installation phase. It's not necessarily, not necessarily dependent on the number of windows that get	
14. D:	no	
15. PL:	created, hmm.	
16. D:	Yeah, yeah, there's a certain connection there.	
17. PL:	Yeah, but if you're gonna, you're gonna set up an environment, whether you make it with 28 windows or whether you make 14	
18. D:	yeah	
19. PL:	it doesn't make any difference.	
20. D:	No, it doesn't	
21. PL:	The test phase it's probably partly dependent on	
22. D:	yeah	
23. PL:	the number of windows, cos then...	

In Excerpt 3 we see that team A has a clear vision of what comparison dimension to use (lines 1–3): 'We need to find a project with a, where the number of windows has been recorded then…. It's probably the only measure we have here then. ' Despite the clear statement of a comparison dimension, a large amount of elaborative work is still needed to make the dimension both valid and relevant to use for comparison purposes. This is what happens in lines 6 to 23. The concept of windows is here elaborated on in order to achieve a shared understanding. This shared understanding makes it possible for the team to use the concept of windows as a boundary concept that facilitates bridging of knowledge. To accomplish this, team A first agrees upon the windows complexity (lines 6–15), making rule of thumb knowledge relevant and suggesting it be applied to the windows. What is clearly shown here is how the meaning potential of the concept of windows moves between contexts, but also that meaning-making has to occur for it to be relevant in this specific setting.

The team thereafter elaborates on the concept of windows in a larger context in order to assess the relevance of windows in an overall picture of the development process (lines 7–23). Hence, the team goes out of the task frame to make sense of the concept of windows. To support the elaborations conducted, a drawing is made parallel to articulating different phases in the software development process (line 7–9): 'You have a project initiation phase, and then you have what we, let's call it a development phase, OK, then you have testing and installation.' Both the articulation and drawing of the different project phases allow an assessment of the relevance of windows in those different phases, which is what takes place in the elaborations that follow. To be able to elaborate and relate the concept of windows to a larger context on a general level, expert knowledge from project management is made relevant. When assessing the relevance of the concept of windows, the development phase is singled out as one possible comparison dimension. The other phases are then ruled out: 'You always need a test phase; you always need an installation phase. It's not necessarily, not necessarily dependent on the number of windows that get created' (lines 13−15). Ruling out these phases is problematised in line 16, which opens up for more thorough elaborations, where imaginary scenarios are created to function as support for the conclusion but also create distance from what is perceived as irrelevant (lines 17-19): 'You're gonna set up an environment, whether you make it with 28 windows or whether you make 14, it doesn't make any difference.' Creating imaginary scenarios helps clarify and support previous arguments, and also helps to convince the other team member. The team is here able to establish a shared framework, making it possible to proceed in their communicative work.

In Excerpt 4 a different approach to this challenge is taken. We enter the data of team B when the developer raises the issue of using similar projects as a basis:

Excerpt 4

Turns	Verbal communication	Description of actions
1. D:	But if we're going to use similar projects as a basis	
2. PL:	Yes	
3. DB:	Yes	
4. D:	Should we	
5. PL:	Haven't you got absolutely loads here?	
6. D:	Should we compare with, like, Gee(?) or compare by complexity, or the type of industry, the number of screens, what's the thing, where are the natural lines	
7. PL:	The screens are really made clear here, cos that's a number that means something anyway, even if it's also got something to do with breaking down.	
8. D:	Hmm	*DB is counting the screens in a diagram from the requirement specifica-tion.*
9. DB:	Hmm	
10. PL:	But I think it should be independent of industry, although of course I know from experience that some industries are more formal than others.	
11. D:	Hmm	
12. PL:	But with the framework conditions we have set here, I think we can free ourselves from that. After all, [name of DB] is trying to suggest here that this is something you've done many times, so you know exactly what you'll use.	
13. D:	The only	
14. PL:	Or how much time you have used.	
15. D:	So, yeah, no, I've never done that.	
16. PL:	You haven't?	
17. D:	No, the only one that I have to go by is that it, it says that it should be ready within ten weeks.	
18. PL:	Yeaaah	
19. D:	But	
20. DB:	Yes, so, you, what you can do, you can typically say that you are done within ten weeks, a bit like what you worked out a while ago.	
21. D:	Hmm	
22. DB:	with the last week being debugging, and the week before that as testing – that's eight weeks left for development.	
23. D:	Hmm	

Excerpt 4 shows that comparable dimensions are not given, nor is it obvious what to use. Several different possible comparison dimensions are articulated in line 6, which shows the variety of options the team can choose from. The articulation of possible comparison dimensions raises the question of what to use, thereby opening up further elaborations and clarifications to explore several dimensions. This is what happens in line 7, where the project leader makes his experience relevant and starts elaborating.

During the elaboration the project leader is not only making relevant his own experience, but is also aligning it to the assumptions the team established as a framework for the task (see Excerpt 2). When aligning with the assumptions, the elaboration takes the form of ruling out: 'But with the framework conditions we have set here, I think we can free ourselves from that'. An attempt to make relevant the experiences of other team members is made during the elaborations (line 12): 'After all [name of database specialist] is trying to suggest here that this is something you've done many times, so you know exactly what you'll use'. This attempt is unsuccessful and reveals what appears to be a lack of relevant experiences in the team (line 15): 'No, I've never done that'. This denial creates a tension in the interaction and closes down the elaborations. To be able to proceed, an alternative is suggested, where the problem of lack of experience is attempted solved by suggesting another point of departure (line 17). What happens is that the team is choosing an alternative approach than what was requested and makes use of information from the task as point of departure for reasoning. Using this alternative approach causes the team to violate the top-down instructions (line 20). In this alternative approach the team makes relevant the experiences from the previous task they just solved (line 20): 'What you can do, you can typically say that you are done within ten weeks, a bit like what you worked out a while ago'. As the experiences from the task just solved are the nearest ones in time, they form a natural part of the knowledge that can be recontextualized. The team does not know the outcome of their performance. Not knowing this should make the experiences less relevant and less valid for use. What the team does is not adapt the knowledge; instead, the reasoning process from the previous task is transferred as a method and applied in the same way in a new context.

In Excerpt 3, team A works quite purposefully to explore a comparison dimension and create a boundary concept that facilitates bridging of knowledge. This was conducted through an extensive recontextualization process, where knowledge embedded in a previously completed project was made relevant, transformed and aligned with the task at hand. A vast amount of elaborative work had to be conducted in order to make this happen. Excerpt 4 shows the explorative approach team B chose, and how this leads to violating the top-down instructions due to lack of experience.

5.3 Making comparisons to quantify number of work hours

The third challenge the teams faced was that of performing comparisons and using them as input for the quantification of number of work hours required for the task.

To understand why making comparisons can be perceived as a challenge, it was most relevant to look at team B, because they chose to explicate the details of the project to be estimated by breaking it down into project activities. We enter the data of team B in Excerpt 5 when the database specialist starts elaborating by breaking the task down into project activities:

Excerpt 5

Turns	Verbal communication	Description of actions
1. DB:	If you, if you have 14 windows, like here, and we use 16 hours on design and development if we had used the previous method. If you now say that you use half of that, but you do it three times instead so that in total you end up with 24 hours per screen, cos you've got three iterations and so you've got one iteration per screen in two hours, so you've got 30 hours per screen	
2. PL:	Hmm	
3. DB:	Then you've got 30 times 14, which is three hundred and	
4. PL:	420.	
5. DB:	420, yes, exactly, then you've got four so, 420 hours that is on the windows.	
6. D:	Hmm	
7. DB:	Then we've got three iterations at two hours each.	
8. D:	So, can you say how much is development out of the total budget? Is it 40, 50 per cent?	
9. DB:	Hmm	
10. PL:	Design and development is about 60 per cent.	
11. D:	Design and development.[Well, then this is just tiny]	
12. PL:	[That's including all the] database stuff	
13. DB:	Yeah, so we must of course have the database, that's got to be done too, and then you've got these import routines and stuff cos that is also some work.	
14. D:	Hmm	
15. DB:	Cos there's, how many imports was it again? Ah, there was quite a few wasn't there? You receive a file, manual loading (6,0) We have two import routines, there's either file or there's manual entry. But we weren't supposed to break it down.	*DB is looking in the requirement specification*
16. D:	No	
17. PL:	No, we weren't	
18. D:	But where	
19. DB:	But we have to	
20. D:	[You have to start with something.]	
21. DB:	[But we have to say something] or we have to say 400	
22. D:	Erm, so receive files, and get them into tables. The only basis for comparison we have is that you say you have been on similar projects that were ten-man, working 100 per, I mean nine-man, that worked 100 per cent during the project period.	

As a result of the lack of identifying a comparison dimension, team B performs incomplete bottom-up reasoning processes to arrive at estimates.

In the elaborations that take place in line 1, the expert knowledge of the database specialist is articulated through creating imaginary scenarios of the development process of windows. In the imaginary scenarios the experiences from the earlier completed task is made relevant and aligned with the knowledge concerning the development process of the task at hand. Through the use of imaginary scenarios the team is able to estimate the number of work hours on windows by employing an incomplete bottom-up approach (lines 3−5).

To complete the calculations of number of work hours on the task, the team makes rule of thumb knowledge relevant (lines 8–10). It is obvious that the rule of thumb knowledge applied here is not a knowledge that is shared by the whole team but rather a part of the project leader's expert knowledge: 'Design and development is about 60%'. Applying this rule of thumb knowledge leads to a quantification of an estimate, which in turn leads to an assessment of size (line 11): 'Well, then this is just tiny'.

This assessment opens up for further elaborations and clarifications, where the elaboration first takes form as explanations before it turns into an investigation of the different parts (line 15): 'How many imports was it again? ' The investigations come to a sudden halt when the database specialist realises that he is breaking the project down into activities: 'But we weren't supposed to break it down' (line 15). This utterance shows that during their own elaborations, a sudden awareness of the instructions from the task is made relevant and abruptly stops the elaborations (lines 16–17). Making the task instructions relevant is followed by a justification of why the project is broken down into activities (lines 19−22). The justification functions as a defence for not being able to remember and make relevant appropriate experiences to find comparison dimensions or finding previously completed projects: 'But we have to say something or we have to say 400 ' (line 20−21).

The justification also makes the team articulate what they know and can use as a point of departure for solving this task (line 22): 'The only basis for comparison we have is that you say you have been on similar projects that were ten-man'. This articulation shows that resources in a project, understood as number of people, is the knowledge team B is able to make relevant and transfer, so that it can function as a boundary concept between a former project and the task at hand.

Excerpt 5 shows how team B articulates their knowledge by breaking the project down into activities and using imaginary scenarios, instead of transforming knowledge from another project and making comparisons. The team faces a challenge when they realize that what they are doing is a bottom-up process. Due to the lack of a comparable dimension, the need to explore alternative reasoning processes to achieve a number of work hours to complete the task is needed. Hence team B relied upon what can be perceived as the first level of recontextualization, namely transfer (Carlile, 2004). This involved reusing the reasoning process from the first task in the estimation study, instead of adapting knowledge from completed projects.

6 Discussion

The aim of this paper has been to explicate the challenges of using historical information in present working tasks. The analytical focus has been on opening up and exploring the communicative and collaborative work of how the challenges emerged and were dealt with in the teams' interaction. These issues were explored by investigating the collaborative processes of employing an analogy-based, top-down estimation approach. The following two research questions guided the analysis: 1) what challenges do teams of software professionals face when applying an analogy-based top-down estimation approach and how do they occur, and 2) what kind of work is needed to utilize knowledge from former software projects when estimating new ones.

The analysis of the social interaction in the selected teams revealed that employing an analogy-based, top-down estimation approach was far from the straightforward work process as proposed by the idealized model (see Figure 1). Not only were considerable amounts of communicative work needed, but also a number of challenges had to be resolved for the teams to complete the task and arrive at an estimate. The analysis revealed three concrete challenges that occurred at rather critical instances in the teams' collaborative work. The first challenge was finding similar completed software projects, the second was exploring and negotiating comparable dimensions, and the third was making comparisons to quantify number of work hours. The results from this analysis support Jørgensen's (2004b) findings: that it was difficult to find similar completed projects and use information from these projects. Furthermore, the analysis presented in this paper offers insight into why these challenges occur and how they are dealt with in the teams' collaborative work.

Finding similar completed projects and using information from these projects requires that the information not only is remembered but also is recognised as relevant at a general level. The analysis showed that this was difficult and the teams did not achieve this at an early stage in the work process. In Excerpt 1, the teams relied upon the capacity to remember alone, even though a database containing information about previous completed projects was available. To be able to find relevant software projects, the teams needed to interactively engage in collective remembering as a communicative practice (Middleton & Brown, 2005), because knowledge is distributed amongst team members. Finding similar projects requires that the team members talk to each other and thus share their knowledge and experiences, so they can jointly engage in remembering information relevant for the task at hand.

To be able to utilize information from former software projects in the estimation work, details of the different projects had to be investigated. First of all, the project to be estimated had to be opened up and explored in detail to establish a shared point of departure for conducting the communicative work. Achieving a shared understanding and point of departure required extensive elaborations and clarifications of the task at hand. In addition, comparable former software projects had to be elaborated in detail so they could be identified as relevant. A double sense-making process was therefore undergone, because both the new project to be estimated as well as the historical information available needed to be made sense of. Thus, these findings correspond with the results of Busby and Payne (1999), in that there is a strong reliance on decomposing a task, even though it is not suitable or asked for in a given context. The analysis in this paper expands this finding by explicating both the reason behind the need for decomposing and the focus on the details in teams' collaborative work. Busby and Payne (1999) suggested that the uniqueness of tasks was overrated and that this was perceived as a reason for earlier experiences not being regarded as relevant. With the theoretical perspective employed in this paper, a team context, and the analytical focus on collaborative work, establishing shared understanding for solving a task in teams is necessary for conducting the work. This cannot be established without exploring details to a certain extent, because a shared understanding of both the new project to be estimated and the historical information have to be achieved. Hence, task uniqueness can be understood as a reason for exploring details so that historical information can be utilized.

Furthermore, the focus on details is also a result of the recontextualization processes, which must take place in order for teams to utilize historical data. The reason for this is the close link between the knowledge acquired and the context it was acquired in (Säljö, 2001). Knowledge needs to be adapted and transformed to fit the context in which it is intended to be used. Hence, recontextualization processes are conducted through the communicative work of elaborating and clarifying information from similar software projects. The analysis showed that the recontextualization processes varied considerably in the teams' collaborative work. Some were quite extensive, in which boundary concepts were created to facilitate bridging of knowledge across contexts. An example is contained in Excerpt 3, which shows how meaning potentials of a possible boundary concept (the concept of windows) are recognised and how sense-making processes have to occur to give specificity and meaning in a given context. This illustrates the complexity of utilising knowledge across contexts and also the importance of articulating what Linell (2009) calls meaning potentials. Other recontextualization processes were easier. As illustrated by Excerpt 4, the concern was adapting a reasoning process, which required less elaboration and clarification to achieve. Through investigating the details, the teams first established shared frameworks that made it possible to proceed in their work and thereby perform the necessary recontextualization processes for adapting and adjusting historical data. Through this collaborative work, the teams were able to utilize knowledge and experiences from previously completed software projects, though not in the way proposed by the analogy-based top-down estimation approach.

Seen together, the collaborative work conducted was more complex than the search-find-compare-adjust sequence outlined by the idealized top-down model (Figure 1).

Through the analyses the work carried out to find similar projects and utilize historical information has been thoroughly unpacked and the results are striking regarding the extent of the collaborative work conducted. Figure 3 provides a visualisation of how historical information becomes recontextualized, when teams attempt to employ an analogy-based, top-down estimation approach. The curved arrows indicate how the interactional processes moved back and forth between addressing the project to be estimated and recontextualization processes and between addressing information about different completed projects recontextualisation processes. When connection points where found, boundary concepts were created to facilitate bridging of knowledge to the project to be estimated. These interactional processes are what constitute the recontextualization processes that have to occur for teams to adapt knowledge and experiences from former software projects. Thus, to utilize knowledge from former software projects to achieve a total effort estimate of a new software project, several recontextualisation processes of various degrees have to be performed.

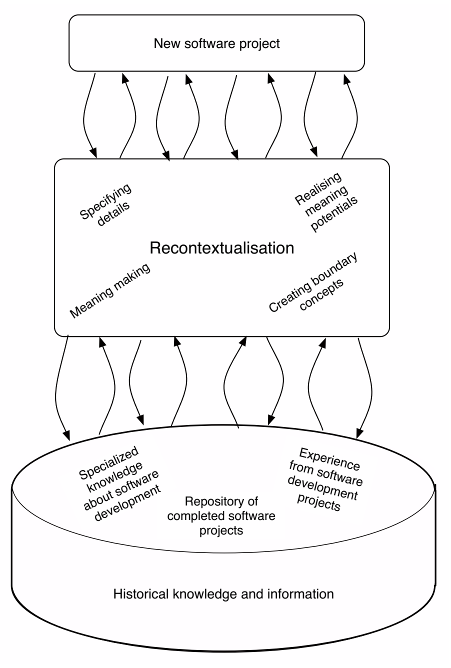

New software project
Specifying details
Realising meaning potentials
Recontextualisation
Meaning making
Creating boundary concepts
Specialized knowledge about software development
Experience from software development projects
Repository of completed software projects
Historical knowledge and information

Figure 3 Visualization of how historical data becomes recontextualized to be useful in new contexts.

Utilising knowledge and experiences embedded in teams of software professionals or historical information when estimating a new software project is not a one-to-one mapping process. The idea of analogical thinking, however, rests on the assumption that mapping is possible (Gentner et al., 2001). This mapping is done by first accessing one or more relevant analogues that are stored in long-term memory and then secondly this analogue is mapped to a target analogue so that similarities can be identified. When investigating what kind of work is conducted to find similar software projects and use historical information for solving a new task in teams collaborative work. Communication and articulation of knowledge are necessary because knowledge is situated. Mapping between projects is therefore difficult to obtain, as teams needs to achieve a shared understanding that requires knowledge to be articulated, adapted and adjusted. Hence, what happens in practice is that recontextualization processes is performed instead, which can be both extensive and difficult. As Figure 3 illustrates, practice does not occur in pure form: it tends to be much more complex than models, which are simplified and idealised versions of the world. Moreover, models always need to be contextualised to have meaning in a given context.

The analysis showed that a considerable amount of the teams' collaborative work consisted of such recontextualization processes, in which meaning potentials were recognized, articulated, negotiated and specified to make meaning in a new context. Hence, the use of recontextualization in this paper further exands Carlile's (2004), account of the concepts transfer, translation and transformation, by explicating the mechanisms and challenges that are involved in the context of using historical information. Thus, this expansion provides insight into how knowledge from former software projects can be used.

Moreover, the use of boundary concepts in this paper is related to the term boundary objects developed by Star and Griesmer (1989). However, in the current study the communication and language used are at the core of the team's collaborative work. Hence, it gives meaning to use the term boundary concepts instead as the time frame are short and the issues that are dealt with relates to the concepts the participants talk about and not objects in the sense Star and Griesmer outlines. As with boundary objects, boundary concepts contain common features from different contexts, but in addition, the process of creating boundary concepts reveals how the bridging of knowledge is facilitated in a short time scale in teams' collaborative work.

7 Conclusion

In estimation work, software professionals often collaborate in teams to achieve an estimate of a software development project. These professionals enter the team and the work assignment with different types of specialist knowledge and experiences that, in combination, is valuable for solving the specific task. A platform is thus needed whereby the communicative work can be conducted and the participants can share and explore their knowledge, experiences and understandings. The analysis presented in this paper shows how communicative work is crucial for utilising historical information across contexts. As solving problems is grounded in the communicative work conducted through elaborations, specifications and negotiations between team members, understanding the collaborative processes that unfolds reveals a dimension of estimation work that has not previously been investigated.

Most research on judgment-based effort estimation has been rooted in the cognitive tradition of empirical research that takes the individual expert as a unit of analysis. The results from these studies explicate one important dimension of software effort estimation—that of uncovering aspects that influence decisions experts make about estimates. In turn, this means that the communicative work to achieve an estimate has not been made visible to any extent. Nevertheless, Jørgensen (2004b) revealed that there are difficulties in employing an estimation approach that requires the use of historical data on a general level, i.e., an analogy-based, top-down estimation approach. This paper has investigated these difficulties further by employing a communicative approach called interaction analysis. With this analytical approach, in which the social interaction constitutes the unit of analysis, it was possible to explicate what kind of challenges teams of software professionals encountered, as well as how they occurred and were dealt with, when employing an analogy-based, top-down estimation approach. Hence, this paper contributes to the research field of software effort estimation in that it provides an understanding of the communicative work required for using historical information to achieve an estimate.

As estimation work is often performed in teams achieving an estimate can be conceived as a collaborative process. It is therefore a need for explorative studies with a communicative approach that can explicate the collaborative dimensions of estimation work. This paper is one contribution to such explorative work. Combining different types of studies employing different research methods and unit of analysis provides opportunities for fostering theory-building in estimation. A set of studies that examine these different dimensions by addressing both individual aspects as well as collaborative aspects of the estimation work will provide a solid understanding of what the phenomenon of software effort estimation is all about. More studies are thus needed, particularly context-specific studies, to explain and understand what actually happens in practice.

Acknowledgements
I would like to thank Professor Monika Nerland, Professor Sten Ludvigsen and Professor Magne Jørgensen for their very helpful comments and suggestions on earlier versions of this paper.

References

Ackerman, M., & Halverson, C. (2004). Organizational memory as objects, processes and trajectories: an examination of organizational memory in use. *Computer Supported Cooperative Work, 13*, 155-189.

Bjørn, P., Burgoyne, S., Crompton, V., MacDonald, T., Pickering, B., & Munro, S. (2009). Boundary factors and contextual contingencies: configuring electronic templates for healthcare professionals. *European Journal of Information Systems, 18*, 428-441.

Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches - A survey. *Annals of Software Engineering, 10*, 177-205.

Boehm, B. W. (1981). *Software engineering economics*. New Jersey: Prentice-Hall.

Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering, 10*(1), 4-21.

Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation practice in a multi-specialist team. *Scandinavian Journal of Information Systems, 22*(2), 65-98.

Bratthall, L., Arisholm, E., & Jørgensen, M. (2001). *Program understanding behavior during estimation of enhancement effort on small Java programs*. Paper presented at the Product Focused Software Process Improvement, Kaiserslautern, Germany.

Busby, J. S., & Payne, K. (1999). Issues of organisational behaviour in effort estimation for development projects. *International Journal of Project Management, 17*(5), 239-300.

Carlile, P., R. (2004). Transferring, translating and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science, 15*(5), 555-568.

Cohn, M. L., Sim, S. E., & Lee, C. P. (2009). What counts as software process? Negotiating the boundary of software work through artifacts and conversation. *Computer Supported Cooperative Work, 18*(5-6), 401-443.

Dittrich, Y., Randall, D. W., & Singer, J. (2009). Software engineering as cooperative work. Editorial. *Computer Supported Cooperative Work, 18*(5-6), 393-399.

Engeström, Y. (1992). Interactive expertise. Studies in distributed working intelligence. Helsinki: University of Helsinki, Dept of Education.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science: A Multidiciplinary Journal, 7*(2), 155-170.

Gentner, D., Holyoak, K. J., & Kokinov, B. N. (Eds.). (2001). *The analogical mind. Perspectives from cognitive science*: Massachusetts Institute of Technology.

Greeno, J. G., Collins, A. M., & Resnick, L. B. (1996). Cognitions and learning. In D. C. Berliner & R. C. Calfee (Eds.), *Handbook of educational psychology* (pp. 15-46). New York: McMillian.

Grimstad, S., & Jørgensen, M. (2007). *The impact of irrelevant information on estimates of software development effort*. Paper presented at the Australian Software Engineering Conference, Melbourne.

Haugen, N. C. (2007). Moderne systemutvikling og estimering [Modern system development and estimation] *Presentation held at Estimation seminar 24 October, 2007*. Retrieved 20 October, 2009, from http://simula.no/research/engineering/projects/best/seminars/Estimation%20Seminar%2024.10.2007

Heemstra, F. J. (1992). Software cost estimation. *Information and Software Technology, 34*(10), 627-639.

Heemstra, F. J., & Kusters, R. J. (1991). Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems, 1*(4), 223-237.

Hihn, J., & Habib-Agahi, H. (1991). *Cost estimation of software intensive projects: A survey of current practices*. Paper presented at the International Conference on Software Engineering, Austin, TX, USA.

Holyoak, K. J. (2005). Analogy. In K. J. Holyoak & R. G. Morrison (Eds.), *The Cambridge handbook of thinking and reasoning*. New York: Cambridge University Press.

Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences, 4*(1), 39-103.

Jørgensen, M. (2004a). A review of studies on expert estimation of software development effort. *Journal of Systems and Software, 70*(1-2), 37-60.

Jørgensen, M. (2004b). Top-down and bottom-up expert estimation of software development effort. *Information and Software Technology, 46*(1), 3-16.

Jørgensen, M. (2005). *The "magic step" of judgement-based software effort estimation*. Paper presented at the International Conference on Cognitive Economics. New Bulgarian University, Sofia, Bulgaria.

Jørgensen, M. (2007). Forecasting of software development work effort: Evidence on expert Judgement and formal models. *International Journal of Forecasting, 23*, 449-462.

Jørgensen, M., & Carelius, G. (2004). An empirical study of software project bidding. *IEEE Transactions of Software Engineering, 30*(12), 953-969.

Jørgensen, M., & Grimstad, S. (2005). *Over-optimism in Software Development Projects: "The winner's curse"*. Paper presented at the Proceedings of IEEE CONIELECOMP, Puebla, Mexico.

Jørgensen, M., & Grimstad, S. (2008). Avoiding irrelevant and misleading information when estimating software development effort. *IEEE Software*((May/June), 78-83.

Jørgensen, M., Indahl, U., & Sjøberg, D. I. K. (2003). Software effort estimation by analogy and "regression toward the mean". *Journal of Systems and Software, 68*(3), 253-262.

Keung, J. (2009). *Software development cost estimation using analogy: A review*. Paper presented at the 20th Australian Software Engineering Conference 2009 (ASWEC'2009), Gold Coast, Australia.

Kjærgaard, A., Nielsen, P. A., & Kautz, K. (2010). Making sense of project management: A case of knowledge sharing in software development. *Scandinavian Journal of Information Systems, 22*(1), 3-26.

Linell, P. (1998a). *Approaching dialogue: Talk, interaction and contexts in dialogical perspectives* (Vol. 3). Amsterdam: John Benjamins Publishing Company.

Linell, P. (1998b). Discourse across boundaries: on recontextualisation and the blending of voices in professional discourse. *Text: an interdisciplinary journal, 18*(2), 143-157.

Linell, P. (2007). Meaning potentials and the interaction between lexis and grammar. Some empirical substantiations. *Pragmatics, 17*.

Linell, P. (2009). *Rethinking language, mind, and the world dialogically*. Charlotte, NC: Information Age Publishing Inc.

Mair, C., Martincova, M., & Shepperd, M. (2009). *A literature review of expert problem solving using analogy*. Paper presented at the 13th International Conference on Evaluation and Assessment in Software Engineering (EASE), Durham University, UK.

Mair, C., & Shepperd, M. (2005). *The consistency of empirical comparisons of regression and analogy-based software project cost prediction*. Paper presented at the 4th International Symposium on Empirical Software Engineering (ISESE), Los Alamitos, CA, USA.

Middleton, D. (1997). The social organization of conversational remembering: Experience as individual and collective concerns. *Mind, Culture and Activity, 4*(2), 71-85.

Middleton, D., & Brown, S. D. (2005). *The social psychology of experience. Studies in remembering and forgetting*. London: SAGE publications Ltd.

Middleton, D., & Edwards, D. (Eds.). (1990). *Collective remembering*. London: Sage Publications.

Moløkken-Østvold, K., & Jørgensen, M. (2003). *A review of surveys on software effort estimation*. Paper presented at the International Symposium on Empirical Software Engineering (ISESE 2003), Rome, Italy.

Moløkken-Østvold, K., & Jørgensen, M. (2005). Expert estimation of web-development projects: Are software professionals in technical roles more optimistic than those in non-technical roles? *Empirical Software Engineering, 10*(1), 7-30.

Säljö, R. (2001). *Læring i praksis. Et sosiokulturelt perspektiv [Learning in practice. A sociocultural perspective]*. Oslo: Cappelen akademiske forlag.

Shepperd, M., Shofield, C., & Kitchenham, B. (1996). *Effort estimation using analogy*. Paper presented at the International Conference on Software Engineering, Berlin, Germany.

Sommerville, I. (2001). *Software Engineering* (6 ed.): Harlow: Addison-Wesley.

Sommerville, I. (2007). *Software Engineering* (8 ed.): Pearson Education Limited.

Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: amateurs and professionals in Berkeley's museum of vertebrate zoology, 1907-39. *Social Studies of Science, 19*, 387-420.

Walkerden, F., & Jeffery, R. (1999). An empirical study of analogy-based software effort estimation. *Empirical Software Engineering, 4*(2), 135-158.

Appendix A: Top-down instructions – adapted from Jørgensen (2004b)

Important: When estimating the project you are not allowed to break the project into activities. Instead, you are supposed to try to estimate the project through a comparison with previously completed (preferably similar) projects. In other words, you are supposed to use an analogy-based (top- down) estimation process. Discussions and all written notes shall reflect that process.

This estimation process may be unfamiliar to many of you, but do your best. The projects you compare with do not need to be very similar to the project to be estimated to be used. You may, for example, compare characteristics such as number of screens, number of tables of the current project with previously completed projects.

Description of the estimation context

Your company has already got the contract of developing the software described in, the requirement specification - not included in this paper of confidentiality reasons. The task of your estimation team is to estimate the effort for the purpose of the planning of the project.

Most of the analysis phase is already completed and shall not be included in the estimate. In addition to the most likely effort, you are supposed to provide the minimum (best case) and maximum (worst case) effort, and the probability that the actual effort will be in between the minimum and the maximum effort.

Example: You believe that the most likely use of effort for a project is X work-hours, that the minimum effort is as little as Y work-hours, and that the maximum effort is Z work-hours. You estimate that it is P percent likely that the actual effort is between Y and Z:

You do not know who the project members will be. Assume that the participants are normally skilled employees of your company.

Sequence of the estimation work

Step 1 (individual work): Read and understand the requirement specification.
Step 2 (teamwork): Discuss and agree on an estimate on the most likely effort, the minimum effort, the maximum effort and the probability that the actual effort will be inside the minimum – maximum interval). Describe important assumptions. Important: Emphasize to find earlier projects (preferably similar), and to use this in the estimation process.
Step 3 (individual work): Complete questionnaire.

Appendix B: Transcript conventions

[ ]	Overlapping talk
(8,0)	Timed pause

Article III

Børte, K., Ludvigsen, S., & Mørch, A. (submitted). The role of concepts in expert work: Unpacking ‘the magic step’ in software effort estimation.

The role of concepts in professional work: Unpacking the “magic step” in software effort estimation

Kristin Børte[1&2], Sten Ludvigsen[3], Anders Mørch[3]

[1]Simula Research Laboratory. P.O.Box 134, NO-1325 Lysaker; [2]University of Oslo, Department of Educational Research, P.O.Box 1092, Blindern, NO-0317 Oslo; [3]University of Oslo, Intermedia, P.O.Box 1161, Blindern, NO-0318 Oslo

Abstract: This paper examines the use of concepts in professional work by analysing the specialised work practice software effort estimation. The aim is to achieve an understanding of how software professionals invoke different types of knowledge when reasoning and reaching a decision on an effort estimate of a software development project. The step from reasoning to decision-making has been referred to as “the magic step” in software effort estimation. We propose that by taking a socio-genetic perspective on concepts in activities, which allow for a focus on three interrelated levels of understanding, institutionalized practice, individual knowledge and dialogue and activity, the ways in which software professionals reach a decision can be unpacked. The results from our empirical analysis showed that the user story mediates the types of resources and knowledge needed to solve the task. Concepts from the knowledge domain are used to frame the task and the participants develop a partially shared understanding, enough to take the next step in the problem solving activity. The paper argues that the magic step is found in the analysis of the social interaction in which the concepts used are anchored in the knowledge domain of software engineering and in the historical experiences of the participants and subsequently become activated.

Keywords: Professional work, concepts in activity, software effort estimation

Introduction

Professional work is based on long-term education and training. Competence is thereby developed in education for engagement in specific types of problem solving, and provides a basis for such work, which is shared by practitioners with a common educational background. At the same time, activating knowledge and making it relevant to solve specific problems in work is not straightforward (Konkola, Tuomi-Gröhn, Lambert, & Ludvigsen, 2007). Each problem setting has its own set of complex issues, which need to be attended to in locally specific ways. Moreover, work is often performed as collaborative processes in groups and teams. Hence, to understand how expert work is carried out, we need to focus on the work process as such, and on how problem solving is carried out in activity. The present paper is a contribution towards this, and focuses on the work of software professionals, with specific attention given to the role of concepts in collaborative problem solving.

The aspect of expert work that we address here, is how concepts are used and handled in the specificity of software effort estimation. We take as a point of departure, that when engaged in problem solving, professionals draw on resources that originate from research and developmental work in a particular knowledge domain as well as on experience-based knowledge. In both cases, however, the resources they draw upon have a discipline- or profession-specific character (Lauder, 2009; Young, 2009). When utilized in activities these different resources are intertwined in language. By way of education and training, professionals are socialized into specialised discourse communities in which language and artefacts are used to solve problems.

The empirical focus is on how a team of professionals with backgrounds in software development, creates estimates of the work effort needed for developing new components of a software system. This work, which is called software effort estimation, constitutes a specific and important activity in software development projects (Boehm, Abts, & Chulani, 2000; Kjærgaard, Nielsen, & Kautz, 2010; Sommerville, 2007). However, it has proven to be a difficult endeavour. A review of studies of software development projects shows that 70% to 80% of such projects overrun their estimates and spend on average 30% to 40% more effort than estimated (Moløkken-Østvold & Jørgensen, 2003) The consequences of providing inaccurate or over-optimistic effort estimates can be severe and may result in lost contracts, delays in implementation, or low quality software for the companies involved. Software effort estimation should therefore be perceived as a societal and organizational problem, and a better understanding of this work practice may yield important results for society.

As a type of problem solving, effort estimation takes the character of "wicked problems" (Rittel & Webber, 1973) or what is more commonly described as, ill-structured problems (Simon, 1996). Being wicked or ill-structured means that there is no best solution to the problem and that achievement goes by way of finding solutions that are good enough within a defined set of constraints, such as time and money. In doing this, knowledge needs to be made relevant and useful in this activity. Moreover, software effort estimation can be conceived as a specific type of planning activity that usually involves more than one person. Being able to plan and control for future incidents is seen as an important part of participating in estimation work. However, planning and controlling or predicting work effort is a complex issue. Planning can also be understood as a meaning-making activity (Linell, 2009), in which prediction must be seen as part of social and cognitive processes within social practice. This makes the estimation practice an interesting example to investigate when looking into the role of concepts in professional work.

Language consist of concepts that do not, in general, have any set of fixed meanings, but rather need to be understood in relation to what people are trying to achieve in collaborative activities. Concepts have two primary layers: The first layer is that a concept incorporates information that could be relevant to the here and now, while the second layer is the historical accepted use of the concepts (Linell, 2009). Concepts in the workplace are often loaded with history as a result of being part of the long-term historical development of inscriptions, artefacts and conceptual tools. The actual meaning making with concepts is contingent in an activity. One of their functions is that they provide opportunities to classify phenomena and through this, cope with a high degree of complexity.

This article looks into one particular aspect of this issue, namely how software professionals in teams reason and reach consensus on an effort estimate through their use of language and concepts. The study adopts a socio-genetic perspective, which implies that knowledge is associated with the concepts that are invoked in talk. To understand what counts as knowledge in estimation practices, we study the work of expert practitioners in the field. Raising the question of what counts as knowledge makes us aware of how and what kind of knowledge is seen as valuable, or, alternatively, what becomes invisible in the practice. This challenge makes us sensitive to changes in how knowledge is negotiated and treated in professional work.

The research questions raised in this paper are:

- What concepts direct participants' talk and guide their decision-making process?
- What kinds of knowledge are invoked in the work of achieving an effort estimate, and where does this knowledge come from?

The questions are examined in the context of a team of software professionals who employ a specific estimation technique, known as planning poker, for estimating the effort of one release of a large software system that handles public pensions and loans.

The structure of the paper is as follows. We begin by describing software effort estimation as a type of work as it is revealed in research literature. Next, we present the theoretical perspective that forms the premises for the empirical analysis. Then we present our methods and analytical strategies before we conduct an empirical analysis. In the final section, we compare our results with the results reported in the literature using our theoretical perspective, and summarize our findings.

Software effort estimation as a specific type of work

A specialised work practice such as software effort estimation provides a good case for examining professional work because both a specialised technical knowledge of software development and the capability of dealing with a software system that is yet to be developed are needed.

The large number of overruns in the IT-industry has been a known problem for years and thus software effort estimation has been addressed in research literature since at least the 1970s. The dominating focus in this research field has been on developing and improving formal estimation models in which software cost estimation has been investigated from a technical point of view (Jørgensen & Shepperd, 2007). Since 1990, there has been an increase in research papers addressing expert judgment-based estimation, which is the most common estimation approach used in the industry (Heemstra & Kusters, 1991; Hihn & Habib-Agahi, 1991; Jørgensen, 2004). The focus of attention has turned from the technical details towards the individual expert performing the estimation work. A large part of this research looked into what kind of aspects influence the decisions that software professionals make when estimating the effort of a software development project. For instance, in a review article by Halkjelsvik and Jørgensen (submitted) on studies of judgment-based predictions of performance time, it is pointed to that in software development, aspects such as anchor information, wishful thinking, request format, and irrelevant information influence the judgments software professionals make on an effort estimate.

Many studies of expert judgment-based estimation use the individual as the unit of analysis (methodological individualism), but the social aspects are rarely taken into account. This also applies to the few studies of group estimation that have been conducted (Halkjelsvik & Jørgensen, submitted; Moløkken-Østvold, Haugen, & Benestad, 2008). However, in recent research, a stronger focus on the communicative and social aspects of estimation work has been addressed (Børte, submitted; Børte & Nerland, 2010) and new estimation techniques have been developed that facilitate social interaction to a larger extent.

Planning poker (Grenning, 2002) is one such technique that facilitates social interaction. As this is a rather new technique, not much research has been done to investigate this way of achieving an estimate in teams. Haugen (2006) compared the use of planning poker with unstructured group estimation to investigate whether introducing planning poker improved the estimation performance. The results of the study showed that the use of planning poker estimates on familiar tasks was more accurate than the unstructured group discussion. However, for unfamiliar tasks the use of planning poker increased the inaccuracy of estimates.

A second study that investigated the planning poker technique was conducted by Moløkken-Østvold et al. (2008). The purpose of this study was to explore the group process of using planning poker and to compare the accuracy of estimates provided by the use of planning poker with that of the estimates achieved individually. The results showed that the tasks that were estimated with planning poker were less optimistic than the results from a statistical combination of the individual estimates. In addition, the tasks estimated with planning poker were more accurate than the same tasks estimated individually and statistically combined.

The planning poker technique provides opportunities to study the use of concepts in professionals' work as it facilitates social interaction during the construction of an estimate. What makes this technique interesting, is, firstly, that it is organised in a way that ensures the participation of all team members, regardless of position or experience. Secondly, that it requests justification or explanations from the team members who provide the highest and lowest estimate as part of the work process. Thirdly, that it relies on the use of playing cards as a material artefact to perform the estimation. It is in social interaction that the individual participant's knowledge gets connected to the concepts and artefacts that are part of the collective resources needed by the participants to activate and reason with when achieving an estimate. In order to understand this as phenomena, we need a theoretical perspective.

Concepts in activities: A socio-genetic perspective

In social interaction people use language, concepts and activate resources that constitute what they achieve in interaction. The concepts that we use come from our everyday experiences and some are appropriated through education and specialised practices (Mäkitalo & Säljö, 2002; Vygotsky, 1978). In practical activities, concepts, whether they come from experience or a particular knowledge domain such as software engineering, create the direction for talk and thus how problems are solved. Concepts and their relationship to each other do not include a fixed set of meanings, but have meaning potential relative to the activities performed. The meaning making processes and the realisation of meaning potentials that occur through social interaction becomes an important part of the interactional work in problem solving. These activities have their basis in semiotic mediation. Different concepts are developed as part of an institutional practice, while they may also "travel" to participants in ways that require considerable interactional work. Recontextualisation and decontextualisation thereby becomes part of productive interactions (Linell, 2009).

In this article, we take a socio-cultural position, or what is called a socio-genetic perspective, on social interaction. We emphasize that the socio-genetic perspective gives us a set of powerful analytic tools for understanding social interaction at multiple levels (De Graaf & Maier, 1994; Valsiner, 1994; Valsiner & Van der Veer, 2000; Van der Veer, 1994). The multiple levels involve three interrelated aspects of human activities: 1) Institutional practices, 2) individual knowledge, and 3) dialogue and activity. *Firstly*, institutional practices are understood as historically driven events in which concrete artefacts and language represent the collective knowledge. This collective knowledge becomes a cultural resource and can reveal potential ways of organizing activities. *Secondly*, the individual who takes part in the institutional practice holds a unique and personal history (experience) and knowledge. *Thirdly*, the dialogues and activities in an institution develop in the intersection between the collective knowledge of the participants, and the cultural resources that are enacted. Thus, what becomes part of each participant's own trajectory, also becomes part of the institutional trajectory.

The socio-genetic perspective therefore implies that learning and knowledge construction cannot be understood only by studying individual achievements, or be seen as determined solely by a local situation or social structure. These need to be seen as relational and situated.

It is important to thoroughly study these socio-genetic processes in software effort estimation if we want to understand how participants collaboratively solve complex problems and co-construct knowledge. Whereas institutional practices can be described and analysed at different levels, it is important to recognize that these levels should be understood as interdependent. The socio-genetic perspective employed in this paper makes it possible to focus both on collective aspects of human practices and on how actors orient and position themselves to take part in moment-by-moment interaction.

Data material and analytic strategy

The data material that is analysed in this paper consists of video recordings from real life estimation meetings in the Norwegian software industry where planning poker was used as an estimation technique. The software development project that is estimated in the meetings is an on-going (re)development of a large software system that was initiated by the government for administrating pensions and loans. The software project has three software suppliers that are involved in the development, one in-house team from the customer whose project is (re)developed, and two suppliers from external consultancy companies. The project has its own premises where all the project members from the different software suppliers are located and conduct their daily work. A total of 88 software developers participate in the work and they are organised into 11 different teams. The actual development of this software system is planned to last for three years and is divided into a series of about three releases per year.

As part of the work to realise this large project, approximately 300 high-level requirement specification documents describing the software system to be developed has been made (Hannay & Benestad, 2010). Each of these requirement specifications comprise a *user story*, which is a description written in the customers' language of what a user should do from the users' perspective in any given situation. For example: "As a caseworker I should be able to calculate the pensions for all members". These requirement specifications represent development work to be done and are placed in queues called “backlogs”. For a specific release, more detailed specifications, user stories, and design specifications are developed together with specific release backlogs comprising the more specific tasks that are going to be implemented in the particular release. All the different documents that are developed and changes to these documents are registered in the project management tool JIRA (http://www.atlassian.com/software/jira/).

Prior to the start-up of the work on a release the key stakeholders meet to discuss the release backlog i.e., the tasks that are going to be implemented. During this meeting, changes, refinements and specifications are made. Then the release backlog is split into three different specific backlogs, one for each subcontractor. Each of the subcontractors, here, two consultancy firms and the customer, estimate their designated specific backlog with their preferred estimation method. In this case, two of the subcontractors, the in-house team and one of the external suppliers, used the planning poker estimation technique, while the third supplier, which was external, used a bottom-up approach supported by the company’s own model, developed on grounds of historical data from the company. The work of estimating the effort of the release was conducted in meetings in which representatives from the different teams that were going to develop the task were participating. The number of participants in the estimation meetings ranged from 3 to 12 persons depending on the estimation approach or technique that was employed. The estimation work of one release was spread out on different meetings that were held during one week in March 2010. One estimation meeting at each subcontractor, three in total, has been videotaped. The videotaped meetings lasted 2.04, 3.41 and 2.43 hours respectively.

After viewing the video recordings of the three meetings, we narrowed down the data material to the teams that used planning poker as an estimation technique. Thereafter, we decided to analyse in depth the interactional work conducted by one of the teams to get a close look at the collaborative problem solving in this work. The selection of this team was based on the richness of the interaction, which provided good opportunities for exploring the role of concepts in professional work and should be seen as theoretical sampling since the data are selected based on a specific problem that is considered a key issue in the estimation research. As the in-house team was already familiar with the system to be (re)developed and thus a lot of the communicative work and understandings were implicit and shared in the team and not made explicit through the work process, we chose to follow the team from the external supplier.

The video-recordings were the first step in the analysis, and were used as an ethnographic frame, which means that they provided an overview of the material and the setting for estimation. The next step was to understand the "logic" of the participants' reasoning. In this step, we, as analysts, looked at particular episodes several times. After this process, we chose a few episodes that demonstrated typical actions and sequences of talk. The selection of these episodes is based on two premises. The first is what seems to be important to the data and the activity itself, which in these data is the goal of reaching consensus. The second is a theoretical problem that we will later address, based on the literature review of estimation and how estimation practice is conceptualized in software development.

In the analysis of data, we used principles from interaction analysis (Derry et al., 2010; Furberg & Ludvigsen, 2008; Jordan & Henderson, 1995; Silverman, 1998) meaning that we used the ethnographic video-data to frame and analyse the selected episodes, considering them to be demonstrations of sense-making and participation structures. Social interaction analysis combines aspects of conversation content with how the talk is performed and accomplished through participant interaction. Considered together, it provided insight into how knowledge is constructed by the participants in estimation activities. This knowledge is seen as a set of shared resources that participants use to reach consensus about how much work will be involved in the estimated task.

In order to analyse the sequences of actions and utterances, we used the categories *clarification*, *elaboration*, *justification*, and *specification* as sensitizing means. These categories have been adopted from our previous work and fine-tuned for this specific analysis (Børte & Nerland, 2010; Furberg & Ludvigsen, 2008). Through these analytic categories we interpreted the participants' interactional accomplishments. The analysis of the data was conducted in two steps (first and second order analysis). In the first order analysis (Linell, 2009), we analysed the data from the participants' point of view, i.e. what they were trying to achieve. Based on theory and the review, we created a second order analysis, which placed the analysis in the field in an attempt to understand estimation as a specific type of social practice (Linell, 2009). In the second order analysis, we make analytic generalizations based on a synthesis of the data, the review, and the analytic concepts based on the theoretical socio-genetic perspective. In the following we provide a description of the estimation setting before we present the empirical analysis.

The estimation setting

The estimation work session took place in a meeting room in which eleven participants from the different development teams participated. The majority of the participants were programmers. In the estimation meeting, several artefacts were used, in particular, design specifications that consisted of flow diagrams, showing the suggested flow of logic among the components of the planned software system. This was presented on a whiteboard with a projector. The participants were familiar with this way of representing programs as well as the symbols used in the chart. In addition, the team members each had a deck of planning poker cards that they used when estimating the different tasks in the release, see Figure 1. The number on the cards represents what is named relative story points of the work effort involved in solving a task. For example the difference between card number 1 and 2 is that number 2 represent twice the amount of work effort needed to solve the task.

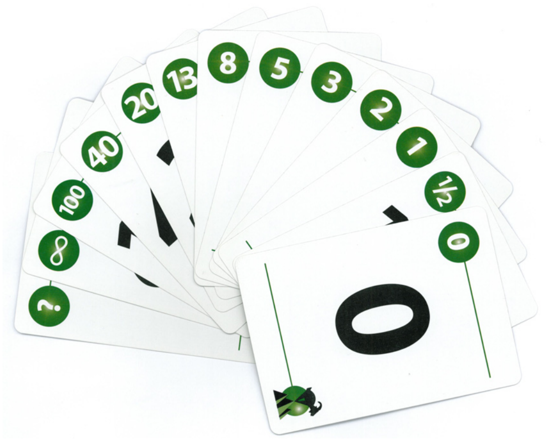


Figure 1. Illustration of a deck of planning poker cards

The instructions for playing planning poker that are included in a deck of planning poker cards follows these steps: 1) The estimation task is presented, 2) The task is discussed in the team, 3) Each participant privately selects a card representing his/her estimate, 4) Once all the participants have selected a card, the cards are flipped over simultaneously, 5) If all the cards show the same number, then that is the estimate, 6) If the cards are not the same, the group discusses focusing on the outlying values, 7) Step 3-6 are repeated until the estimates converge (Cohn, 2006; Grenning, 2002). These instructions are often adapted to fit the context in which it is used.

To investigate how software professionals decide upon an effort estimate, we have chosen to analyse a sequence of work in which a specific task illustrated in a flow diagram is followed from presentation to achieving an estimate. The selected sequence illustrates how the team reason and use concepts to explore, justify and clarify issues when employing the planning poker technique to achieve a consensus on an estimate. In the work process, the team uses several artefacts. The main material artefacts in the beginning of the work process are the design solution and its flow diagrams. These are used so that the team can achieve a shared understanding of the specific estimation task also referred to as a user story. When a shared understanding is achieved, the team's use of material artefacts shifts to the deck of planning poker cards. These cards are used as a representation of the different team members' estimates on the specific task/user story and thereby guide what kind of activity happens next. That is, whether further justification, clarification or negotiations are needed or consensus is achieved. There is one designated group leader who leads this estimation session and decides when it is time to move on in the work process.

Analysing the sequence of estimation work

In the following, an in-depth analysis of a sequence of excerpts will be presented. The sequence, called "stopping the pension", starts when the team attempts to achieve a shared understanding of how a particular user story can be implemented. The different participants in the meeting are identified by the abbreviations P1-7 and GL, which identifies the group leader. The utterances below are given numbers so that we can refer back to them in the analysis. The excerpts start when the team member in charge of the user story presents it to the other team members.

Excerpt 1: Clarifying the user story

1. P1: Two more, anyway. Next thing is to stop pension for case flow in a death case. So we're at the point in the logic where we have to check if he has anything. But yeah, that was a bit of a confusing case, but if he has some payments from us they have to be stopped. And that has to be as a separate task. So a task will come up... erm... if he has any payments from before, they should be stopped too.
2. GL: Haven't we?
3. P2: Stop[ped]
4. GL: [Have]n't we estimated it from before?
5. P1: Not in the case flow.
6. P2: But what's involved in getting it stopped? Is it...
7. P1: It's a specific code that's entered... erm...
8. *<many speaking>*
9. P3: It's ( )
10. GL: Yes, we [must (follow)]
11. P2: [it's just] a stop code in your history.
12. P4: Yeah, and I'm thinking of linking it there to the one called "Stop pension" that's on... 35.1, that one too, but with a different supplier, [so here we're a bit dependent on]
13. P5: [yes, but stop, stop, stop]
14. P5: ( ) stop pension on... in... erm... which we're doing in desktop.
15. P1: Yes, exactly, it's in desktop. There's got to be a task here in case flow. Which stops the pension logic. It's created by a different story, that's the one you have *<points to P5>*.
16. P5: Yeah.
17. P1: But there we can have a task that ( )

When introducing a particular user story, the importance of clarification as a means to both achieve shared understanding in the team and to develop the user story as a narrative becomes important. Immediately after the user story is introduced (line 1) questions are raised, which indicates that the issue that needs to be clarified is what "stopped" means in this context (lines 2-6). The request for clarification in line 6 creates a set of actions that specifically address what "stop" means in this particular case. The important interactive work here is the combination of what it means for the programmers, and for the users of the system (lines 7-17). The resources involved here are all connected to the technical specification of the system, but of equal importance is the issue of which functions the stop code will have for the users of the system (lines 7-12). Creating a stop code is closely linked to a set of actions for the employee working on the specific case. The specification that follows in line 15 about "case flow" indicates the specific line of actions that will be conducted by the employee working on the case. Thus, the specification also allows for a narrative to be created. Furthermore, the distinction between the concepts "desktop" and "case flow" becomes an important issue not only as a conceptual distinction but also as a way of classifying different sets of working with the system. Making this conceptual distinction involves both technical knowledge and knowledge of how the pension system operates. It is the integration of different sources of knowledge that makes it possible for the participants to clarify and specify the user story to achieve a shared understanding. This means that the user story should be seen as part of the institutional practice/socio-genesis of the planning aspect in software effort estimation in which the sequential aspects become a narrative that makes it possible to connect the more generic aspects and the specific aspect of a particular problem.

Excerpt 2: Specifying leading to sequencing actions

1. GL: Shall I come in and help then? What I'm worried about is that you will have one table for all... erm... all the types of services you have. So old-age pension... erm...
2. P2: Now you're talking about a GUI table, that is, one of those <*makes signs with hands*>
3. GL: A GUI table, so it'll work as a... as a... as a filter.
4. P2: Yeah, but you... yeah.
5. GL: And to begin with the way it is, is that in the draft you'll presumably be able to choose one of them and specifically stop each of them. And what will presumably, I think, be in this user story, and that... then the question is whether... erm. The easiest way you can visualise this user story is to say that in a case we decide to just show the one table. And you can do it there, maybe it could be that you want to do it using a result instead of doing something up there.
6. P2: Who's gonna... erm... who... there's a sequence of things here. If you, in... if there's a sharp variant of that, ready created in a table, then it's...
7. GL: No, yes but he <*points to P5*>... we'll get to that there.
8. P2: Yes, but when should it happen?
9. GL: It should happen before we create that task there.
10. P4: Yes, it should happen quite quickly.
11. P2: Yeah, but the stuff that's happening before that, then it'll be...[you'll have to handle that]
12. GL: [Yes, ideally] we should have estimated that before today.
13. (P5:) We should probably have done it.
14. P2: Yeah, but, I've just got to say that we're depending on there being a table in the desktop where you have these things in stop.

The meaning of the stop code is not a trivial issue. An attempt to specify this concept is conducted in line 1 where the group leader suggests a solution for how the stop code can be solved. When P2 asks for a clarification (line 2) and the group leader, in addition to confirming that he is talking about a GUI table (line 3), elaborates on his suggested solution (line 5), the framing of the task becomes clarified. The elaboration of a solution is taking form as an imaginary scenario where different ways of solving the "stopping" issue is tested by way of language use. To create such imaginary scenarios, the resources used are technical knowledge and experience, which together with the information about the specific user story create a frame for the participants and also a sequencing of the tasks. In addition, the imaginary scenario also functions as a way of developing the narrative of the user story further.

What happens next, is that the solution that has been elaborated on is challenged because it needs to fit a particular sequence of actions. The sequencing is an important aspect, which is confirmed by the actions taking place in lines 6-12. When the group leader says that under ideal conditions they should have estimated this set of tasks before, we interpret this as a pragmatic statement. Our interpretation is that this statement is important because it underlines the fact that it is through specification and the process of understanding what is involved in creating and accounting for the work needed that the new order is constructed. The solution to the problem of sequencing is to make an assumption regarding the "table" that is missing (line 14). Making this assumption by drawing on technical knowledge makes it possible for the team to move on in their interactional work. This technical knowledge works on two levels, it's the connection between the more generic and the situated specifics that makes the assumptions clear.

Excerpt 3: Summing up and playing poker

1. GL: <*Coughs*> Now there's a bit of chaos. Shall we try to muster the troops a bit.
2. P2: Yes, we should probably sum up.
3. GL: Summing up, that is a task that needs to be done... that turns up under certain circumstances. Actually, we've already pretty well estimated these circumstances. Erm... because they are some of the same circumstances as last time. Erm... what should happen is that you should take a GUI element that already exists and put it in a task. Create the task... it's a task we've already created before. Erm... presumably to be able to stop it here in an easy, reasonable manner, either by using a result, or by going in and stopping it just like you do in draft mode. Erm... not necessarily both. Ready?
4. P2: And... to sum up, to stop it there is a server function that can do it... it's not something we... You just have to go and call it, either five times or once or...
5. <*Many participants say* "Yes" *at the same time*>
6. GL: Yes. Ready, steady, go.
7. <*Showing cards*>
8. P5: Blimey. Yeah that was [different, that] is something that we should tidy up...
9. P6: [Go down a step.]
10. <*laughter*>
11. P2: Talked it down from a five.
12. GL: Did I?
13. P2: No, I did that.
14. P5: No, I only have a one.
15. P6: I've got a one as well.
16. P1: There are ones and twos. Shall we say two then, or what?

The group leader uses his authority as a leader and puts forward a request about gathering the troops. When P2 ask for a summary, the group leader picks up the task and describes the whole process, solutions and assumptions agreed upon from Excerpts 1 to 3. The user story has, through the team's talk, been clarified as being different from other previous tasks. The GUI element, which they have assumed exist, creates a framing of the operations involved in solving this task. After the summary, it appears that the team has reached a satisficing shared understanding that is enough to continue the work process and to start quantifying the number of work hours involved to complete the task (line 5).

Each participant has chosen a card with a number that represents their estimate, and when asked to by the group leader (line 6), all the participants show their cards to each other at the same time. The next line of actions (lines 7-16) show that the participants did not reach a consensus about a number, as cards with numbers 1 and number 2 on them are both chosen. This disagreement shows that even though the participants appear to have a shared understanding of the task as indicated in line 5, this is not the case. Furthermore, it is clear that the previous specification has made some of the participants change their opinion, which is indicated in the utterances in line 11-13. In order for the team to reach consensus, the rules of planning poker say that the participants with the highest and lowest numbers should justify their choice of card. This justification is shown in the next excerpt.

Excerpt 4: Justifying and reaching consensus

1. GL: We can... let's just go through it one more time. Should I justify a two then, suddenly?
2. P2: Mmm
3. GL: No, it's a new task and I thought: yeah, there's bound to be a bit of a discussion, about how this here should be stopped like that to begin with. Yes? *<reacts to P4 putting hand up>*
4. P4: Erm... I'd almost done it in a (half)... I... cos I remember we were discussing the previous round.
5. GL: Yes.
6. P4: To bring in something that already exists and create a simple task without any more ( ).
7. GL: Yeah, yeah.
8. P4: That was a half weight you said then. So I thought, OK, I'll take [one]
9. P2: [You] shouldn't trust GL.
10. *<laughter>*
11. P2: Look at me.
12. P7: When it can [sit]
13. P4: [So there] will be two now for the same one, so it sounds much more comprehensive.
14. P7: When so many programmers can sit here and discuss it here. It says to me then that it's not a one, like that, wow, it was [( )]
15. P4: [But] that's got something to do with the clarifications, that has.
16. GL: Yes, no, but I agree with... yes, so... I... yes, no, well, the main problem here is after all that, no, it's probably functionality that's not here, in place yet. So to... yes? *<Response to many participants raising hands>*
17. P3: Take P6 first.
18. P6: Yes, we have after all actually done that check A, whether he has pension from before or not. It's a new task that should be added and we've got the functionality in the task from before.
19. GL: Yes.
20. P1: So there must be... there should be a task that comes in the case flow then.
21. GL: OK?
22.
23. *<... 8 secs>*
24.
25. GL: Ready, steady, go!
26. *<Showing cards – laughter>*
27. P3: Can't you talk about the previous...
28. P5: Has there been a weight here now? A one is very little, after all. It is...
29. P1: Put a two then we're agreed that it's a two, we can't spend any more time on...
30. P6: OK.
31. P5: Two.

The group leader starts to provide a justification for why he chose the card number 2 (line 1), but instead of following through his argument, he responds to a raised hand amongst the participants. A disagreement about the card number 2 is then revealed between the group leader and the participant. The disagreement is followed by a justification referring to the rules that seem to have been agreed upon earlier in the team's work process (lines 3-8). This justification functions as an argument against the card number 2 that the group leader has chosen. What happens next is that this justification is then questioned in a funny tone (lines 9-11), before one of the participants provides a social justification (line 14). This justification makes use of a social statement in order to problematize that this is a simple task. The argument put forward has a rhetorical character since the team at an earlier stage appeared to have a shared understanding. Even though this social justification is not clearly picked up in the interactional work, the utterance from the group leader can be interpreted as an agreement to this position. In the following line of action (lines 17-21), one of the participants reformulates what the task is about (line 18). In this reformulation, the participants single out three important issues that are highly relevant for the team's understanding. It is the task of checking that the pension is done, that is a new task that needs to be added and that the functionality of this task already exists. Through this clarification, together with the specification provided in line 20, the participants seem again to have reached a satisficing shared understanding of the task and are ready to play a second round of poker.

After the second round of playing, the participants have not yet reached a complete consensus and make utterances about the card number 1 being too low. At this stage, the participant that was in charge of presenting the user story makes a decision that it should be the card number 2. As a justification for this decision, other considerations come into play, such as the limited amount of time they have available for conducting the estimation work.

The resources the participants draw on to reach a consensus are intertwined in the interactional accomplishment. Firstly, the social aspects of working in a team, where several participants have to come to a shared understanding of the task through elaborating, clarifying and specifying certain issues. This is a time-consuming activity, which also becomes apparent in the end of the excerpt where time is used as part of the justification. The second aspect is the technical knowledge in which the team is able to specify this particular task as new, and here the generic and specific knowledge are connected to reach a consensus that is based on a temporal shared understanding.

General discussion

In this section, we discuss the empirical questions raised in the beginning of the article. Through the analysis of the estimation sequence it became clear that the specification of the user story had been decided in order to achieve the final estimate.

The user story mediates between the historical practices on the one hand and between the use of generic and specific knowledge on the other, in order to make the assumptions clear and specify what the problem is about. The sequential and narrative form of the user story represents a specific aspect in the institutional genesis of the software development field in which the creation of future scenarios is framing the task. The user story as a mediated resource makes it possible for the participants to invoke the different types of knowledge involved.

The concept of a "GUI element" becomes the framing element for how the task should be approached and accomplished. It's through the participants' conversation that the knowledge becomes partially shared and the requirements become specified (Børte & Nerland, 2010). The contribution of the team is both problematization and consensus-seeking talk in order to complete the specifications. Our interpretation of the conversation is that the participants reached a level of intersubjective understanding, so that it became reasonable to continue. We assume that this type of intersubjectivity does not mean that the participants share mental models of the estimation task, but that they, through interactional work, share enough information that serves as common ground to take the next step in the problem solving activity (Fugelli, 2010; Matusov, 1998; Rommetveit, 1992). This is an interesting finding since it seems that the participants do not use concepts in a systematic way. Another key observation is that in the conversation, the participants make use of concepts such as GUI (Graphical User Interface) in order to frame the task. This is one of the few explicit references to systematic knowledge and concepts from the domain of information systems (user interface design). Most of the other talk makes use of historical experience, without being explicit with regards to domain knowledge. We would like to argue that concepts such as GUI in this dataset makes an important difference when it comes to direction, but that personal experiences seem to be the more dominating orientation when it comes to specifying the work needed for solving the tasks. The participants seem to invoke concepts of systematic character from the knowledge domain only when there is a problem that needs to be framed or reframed.

The socio-genetic perspective we have outlined in this paper allows us to interpret the phenomena at hand at the different levels: 1) institutional practice, 2) personal experiences, and 3) dialogue and activity (Ludvigsen, Rasmussen, Krange, Moen, & Middleton, 2010; Valsiner & Van der Veer, 2000). The analysis shows that the participants bring different experiences and knowledge from all three levels when negotiating about and framing how the task should be solved. The institutional practice of reaching a consensus is achieved through the justification by connecting the social aspects and the use of different types of technical knowledge. At the level of the social interaction, these differences become visible through the different functions in the talk, such as clarification, justification and elaboration, which lead to a higher degree of specification (Børte & Nerland, 2010). The participants' individual contributions should be understood partly as a situated response to previous utterances, but also to the overall practice of reaching a consensus.

This consensus is mediated by different types of concepts and experiences, but also by the procedures provided with the planning poker game. This combined approach makes it possible for socio-cognitive processes to occur, such as problematization, disagreement, confirmation and elaboration. This allows for incremental formalization, supported by a gradual transition from informal to formal processes. The numbers on the cards together with the roles of the participants create a procedural practice in which language is used to construct a justification for the estimation value (final number) decided.

Our interpretation of the communication pattern, characterized by few domain specific concepts, is that the estimation task is embedded in a project organized by a socio-technical structure in which the set of tasks are taken for granted. The communicative work that is needed is connected to achieving a satisficing understanding of the particular estimation task, or what we call microelements of the overall problem solving. The communication becomes more effective when what is taken for granted is not brought up for discussion except when disagreements occur. In this case, the design of a software system stands at the core of the estimation practice. The systematic knowledge thereby becomes partly invisible and not articulated in the data presented.

Figure 2 illustrates how the communicative and interactional work oscillates between the three layers: institutional practice, individual knowledge and dialogical activity. Furthermore, it shows how the communication goes deeper and deeper into the task during their work process through higher degrees of specification. The analytical categories elaboration, clarification and specification illustrate this by processes of narrowing down and achieving a shared, satisficing understanding of the task and a decision on an estimate, whereas the domain specific concepts serve to frame the task.

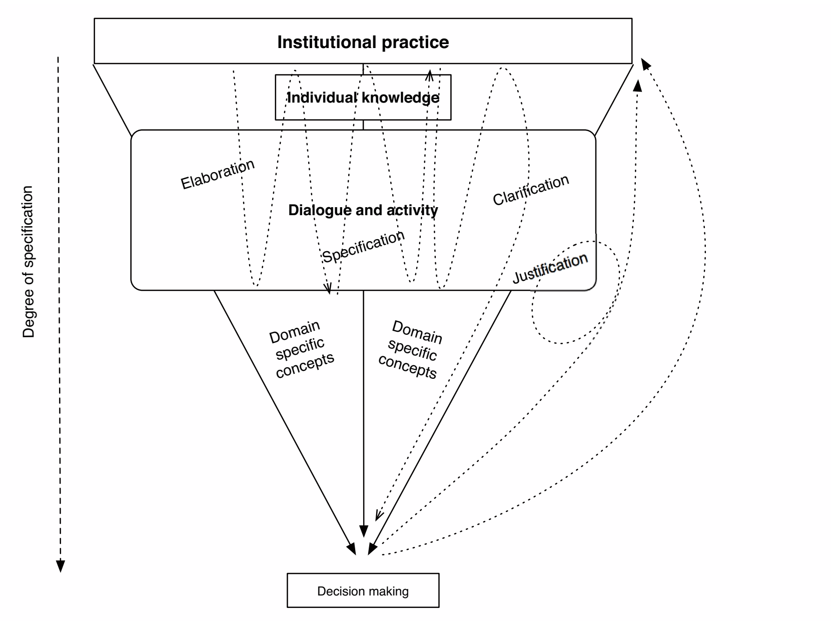


Figure 2. Visualisation of the communicative pattern in the team's work process and how the analytical categories make it possible to obtain higher degrees of specification and reach a decision.

The analytical categories clarification, elaboration, and specification can be compared to design rationale vocabulary, in particular to the concepts issue, position and argument, originally proposed by Rittel and Webber (1973) as part of a method for early stage design and to tackle wicked problems. The design rationale concepts have inspired the development of analytic categories for studying how groups of software professionals work in design meetings (Olson et al., 1996). Their analytic schema was based on content analysis in which discussion categories were identified together with a quantification of the number of transitions between these discussion categories. The different design rationale inspired categories were useful for understanding what types of discussion categories were raised and mainly attended to in the design meeting. However, our aim was to investigate the role of concepts in professional work and thus we needed to go beyond counting and tracing interactions among instances of various discussion categories. Hence, the analytic categories used in our empirical analysis of estimation practice provide insight into how the interactional work is conducted by way of elaboration, clarification and specification, which means that software estimation, is seen as an interactional accomplishment.

Conclusion – what can we say about the 'magic step'?

The question we ask here is do theoretical assumptions of the researchers matter? When seeking to understand the concepts and procedures in reasoning and quantification in software effort estimation in order to account for how decisions are made during program construction. The judgment-based approach to estimation relies on a set of assumptions that from the socio-genetic perspective, provide the analysis with a reductionist account.

In previous studies, it has been argued that quantitative studies need to be supplemented by qualitative studies for understanding the mechanisms behind the factors influencing software design processes (Curtis, Krasner, & Iscoe, 1988). Recent developments in software engineering methodology (e.g., open source, agile development) have increasingly emphasized communication and collaboration as key to assuring good results, favouring qualitative methods. This calls for rich data to provide a sound basis for understanding these practices (Dittrich, Randall, & Singer, 2009). The analysis in this paper contributes to this.

From a socio-genetic perspective knowledge is constructed with a basis in social interaction, drawing on concepts from the knowledge domain of software systems in their efforts to frame and guide the talk among participants. The more generalized knowledge resides in larger knowledge structures and becomes activated by the participants in social interaction (Bowker & Star, 1999; Ludvigsen et al., 2010). Planning and estimation should be understood as a social practice and be conceptualized as dialectical processes among social interaction, individual contributions and the institutional practice.

To understand how participants can combine their personal history with the systematic and more generalized knowledge from the knowledge domains involved, the idea of boundary concepts (Børte, submitted), inspired by (Star & Griesemer, 1989) boundary object serves a purpose of facilitating the use of knowledge across contexts. In our data we found that the concept of "GUI" serves this function of being a boundary concept, since it mediates between different aspects of the specification through concepts, and it stabilizes the estimation talk.

We claim that the "magic steps" to which Jørgensen (2005) refers are to be found in the analysis of social interaction, in which the concepts used are both anchored in the knowledge domain and in the historical experiences of the participants and subsequently become activated. There is a "cognitive leap" between describing and analysing work to creating an estimate for the number of work hours. While the talk about the estimation is grounded in knowledge on how the development can be done, the number for the actual estimate is an abstraction that leaves out the foundation for the estimation. The two reasoning processes are of different character. To clarify, the work implies a number of steps that involves concretization with the frames of user stories, while the estimate involves a meta-step that is categorically different from trying to understand what kind of work that is involved. Thus, this step is not magical but a step that is categorically different and thus differentiates two conceptual schemata. We claim that the specification work and the work to create an estimate should be seen together in order to understand the phenomenon called estimation practice.

The theoretical and methodological implication of this claim is that in order to advance the understanding of software effort estimation, one must study how teams under different conditions reach consensus. Furthermore, the unit of analysis needs to move outside individual heads. It is the social interaction mediated by artefacts that creates the adequate unit of analysis. With such a research strategy, detailed studies of how individuals' talk together combined with how different types of knowledge are activated and made into conceptual building blocks are necessary. It is through detailed multi-level analysis of how estimation practice is framed within larger social and knowledge structures that we can improve practical procedures for estimation work.

Techniques like planning poker create conditions for a richer and more knowledge driven type of conversation to make possible a realistic account of the work involved. Procedures for conversation in terms of rules for the planning poker game, the activation of resources given by the specifications of the project at different levels, and the concepts in use should be given priority in further empirical analysis of software effort estimation.

Acknowledgements

We would like to thank Professor Monika Nerland for her very helpful comments and suggestions on earlier versions of this paper. We would also like to thank the members of the PLASMA project at Simula Research Laboratory and the project management of the PERFORM project at SPK for giving us access to the data material.

References

Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches - A survey. *Annals of Software Engineering, 10*, 177-205.

Børte, K. (submitted). Challenges when utilizing historical information in present working tasks: An analysis of the use of analogies in team-based software effort estimation.

Børte, K., & Nerland, M. (2010). Software effort estimation as collective accomplishment: An analysis of estimation practice in a multi-specialist team. *Scandinavian Journal of Information Systems, 22*(2), 65-98.

Bowker, G. C., & Star, S. L. (1999). *Sorting things out*. Cambridge: MIT Press.

Cohn, M. (2006). *Agile estimating and planning*. Massachusetts: Pearson Education Inc.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31*(11), 1268-1287.

De Graaf, W., & Maier, R. (1994). Sociogenesis reexamined: An introduction. In W. De Graaf & R. Maier (Eds.), *Sociogenesis reexamined* (pp. 1-16). New York: Springer-Verlag.

Derry, S. J., Pea, R. D., Barron, B., Engle, R. A., Erickson, F., Goldman, R., Hall, R., Koschmann, T., Lemke, J. L., Sherin, M. G., & Sherin, B. L. (2010). Conducting video research in the learning sciences: Guidance on selection, analysis, technology and ethics. *Journal of the Learning Sciences, 19*(1), 3-53.

Dittrich, Y., Randall, D. W., & Singer, J. (2009). Software engineering as cooperative work. Editorial. *Computer Supported Cooperative Work, 18*(5-6), 393-399.

Fugelli, P. (2010). *Intersubjectivity and objects of knowledge: Making sense across sites in software development.* PhD, University of Oslo, Oslo.

Furberg, A., & Ludvigsen, S. (2008). Students' meaning-making of socio-scientific issues in computer mediated settings: Exploring learning through interaction trajectories. *International Journal of Science Education, 30*(13), 1775-1799.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. Retrieved 28.05, 2010, from http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

Halkjelsvik, T., & Jørgensen, M. (submitted). From origami to software development: A review of studies on judgment-based predictions of performance time.

Hannay, J. E., & Benestad, H. C. (2010). *Perceived productivity threats in large agile development projects.* Paper presented at the International Symposium on Empirical Software Engineering and Measurement (ESEM 2010), Bolzano-Bozen, Italy.

Haugen, N. C. (2006). *An empirical study of using planning poker for user story estimation.* Paper presented at the AGILE 2006, Minnesota.

Heemstra, F. J., & Kusters, R. J. (1991). Function point analysis: Evaluation of a software cost estimation model. *European Journal of Information Systems, 1*(4), 223-237.

Hihn, J., & Habib-Agahi, H. (1991). *Cost estimation of software intensive projects: A survey of current practices*. Paper presented at the International Conference on Software Engineering, Austin, TX, USA.

Jordan, B., & Henderson, A. (1995). Interaction analysis: Foundations and practice. *The Journal of the Learning Sciences, 4*(1), 39-103.

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software, 70*(1-2), 37-60.

Jørgensen, M. (2005). *The "magic step" of judgement-based software effort estimation*. Paper presented at the International Conference on Cognitive Economics. New Bulgarian University, Sofia, Bulgaria.

Jørgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions of software engineering, 33*(1), 33-53.

Kjærgaard, A., Nielsen, P. A., & Kautz, K. (2010). Making sense of project management: A case of knowledge sharing in software development. *Scandinavian Journal of Information Systems, 22*(1), 3-26.

Konkola, R., Tuomi-Gröhn, T., Lambert, P., & Ludvigsen, S. R. (2007). Promoting learning and transfer between school and workplace. *Journal of Education and Work, 20*(3), 211-228.

Lauder, H. (2009). On knowledge and work. *Journal of Education and Work, 22*(3), 157-162.

Linell, P. (2009). *Rethinking language, mind, and the world dialogically*. Charlotte, NC: Information Age Publishing Inc.

Ludvigsen, S., Rasmussen, I., Krange, I., Moen, A., & Middleton, D. (2010). Multiplicity and intersecting trajectories of participation: temporality and learning. In S. Ludvigsen, A. Lund, I. Rasmussen & R. Säljö (Eds.), *Learning across sites: new tools, infrastructures and practices* (pp. 105-121). London: Routledge.

Mäkitalo, Å., & Säljö, R. (2002). Talk in institutional context and institutional context in talk: Categories as situated practices. *Text, 22*(1), 57-82.

Matusov, E. (1998). When solo activity is not privileged: The participation and internalization models of development. *Human Development, 41*(5-6), 326-349.

Moløkken-Østvold, K., & Jørgensen, M. (2003). *A review of surveys on software effort estimation*. Paper presented at the International Symposium on Empirical Software Engineering (ISESE 2003), Rome, Italy.

Moløkken-Østvold, K. J., Haugen, N. C., & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *The journal of systems and software, 81*, 2106-2117.

Olson, G. M., Olson, J. S., Storrøsten, M., Carter, M., Herbsleb, J., & Rueter, H. (1996). The structure of activity during design meetings. In T. P. Moran & J. M. Carroll (Eds.), *Design Rationale* (pp. 217-239). Hillsdale, NJ, USA: Laurence Erlbaum.

Rittel, H., & Webber, M. (1973). *Dilemmas in a General Theory of Planning. Policy Sciences 4*. Amsterdam: Elsevier Scientific Publishing.

Rommetveit, R. (1992). Outlines of a dialogically based social-cognitive approach to human cognition and communication. In A. Wold (Ed.), *The dialogical alternative: Towards a theory of language and mind* (pp. 19-45).

Silverman, D. (1998). *Harvey Sacks - Social science & conversation analysis*. Cambridge: Polity Press.

Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge: MIT Press.

Sommerville, I. (2007). *Software Engineering* (8 ed.): Pearson Education Limited.

Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: amateurs and professionals in Berkeley's museum of vertebrate zoology, 1907-39. *Social Studies of Science, 19*, 387-420.

Valsiner, J. (1994). Bidirectional cultural transmission and constructive sociogenesis. In W. De Graaf & R. Maier (Eds.), *Sociogenesis reexamined* (pp. 47-70). New York: Springer-Verlag.

Valsiner, J., & Van der Veer, R. (2000). *The social mind: Construction of the idea*. New York: Cambridge University Press.

Van der Veer, R. (1994). The Concept of Sociogenesis in cultural-Historical Theory. In W. De Graaf & R. Maier (Eds.), *Sociogenesis Reexamined* (pp. 117-131). New York, NJ: Springer-Verlag.

Vygotsky, L. (1978). *Mind in society. The development of higher psychological processes*. Cambridge MA: Harvard University Press.

Young, M. (2009). Education, globalisation and the 'voice of knowledge'. *Journal of Education and Work, 22*(3), 193-204.

Attachment

Required enclosure when requesting that a dissertation be considered for a doctors degree

Declaration
describing the independent research contribution of the candidate

In addition to the dissertation, there should be enclosed a declaration describing the independent research contribution of the candidate for each paper constituting the dissertation.

The declaration should be filled in and signed by candidate and co-authors. Use the back of the page if necessary.

The declaration will show the contribution to conception and design, or development and analysis of a theoretical model, or acquisition of data, or analysis and interpretation of data, contribution to drafting the article or revising it critically for important intellectual content etc.

Article no. 1

Title: Børte, K., and Nerland M. (2010) Software effort estimation as collective accomplishment: An analysis of estimation practice in a multi-specialist team. *Scandinavian Journal of Information Systems*, 22(2):65-98.

The independent contribution of the candidate:

I am the first author of this article. My independent contribution consists of having the major responsibility for the final version of the article. In more detail this means that:

- I have been responsible for the data material in terms of narrowing it down and selecting the parts that have been analysed in depth
- I have analysed the data
- I have written up the final version
- I have had contact with the journal during review and followed up editing issues when the final version has been set up for publication.

My co-author has contributed in the following manner:

- She has been a discussion partner and also commented on drafts throughout the work with this article
- She has contributed to the choice of theory that has been employed in this article
- She has written up different parts, such as the theoretical framework and discussion section.

[illegible]
Signature of candidate

Monika Nerland
Signature of co-authors

Article no. 3

Title: Børte, K., Ludvigsen, S., and Mørch, A. (Submitted) The role of concepts in professional work: Unpacking the 'magic step' in software effort estimation

The independent contribution of the candidate:

I am the first author of this article. My independent contribution consists of having the major responsibility for the final version of the article. In more detail this means that:

- I have taken part in collecting parts of the data
- I have been responsible for the data material in terms of narrowing it down and selecting the parts that have been analysed in depth
- I have analysed the data material
- I have written up the final version
- I have managed the contact with the journal during the submission and review process.

My co-authors have contributed in the following manner:

- They have been discussion partners and also commented on drafts throughout the work with this article
- They have contributed to the choice of theory that has been employed in this article
- They have written up different parts, such as the theoretical framework and discussion sections.

[illegible]
....
Signature of candidate

[illegible] [illegible]
....
Signature of co-authors