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.

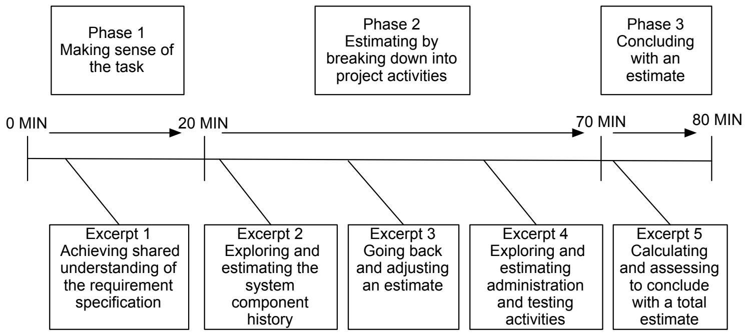
Phase 1
Making sense of the task
Phase 2
Estimating by breaking down into project activities
Phase 3
Concluding with an estimate
0 MIN
20 MIN
70 MIN
80 MIN
Excerpt 1
Achieving shared understanding of the requirement specification
Excerpt 2
Exploring and estimating the system component history
Excerpt 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.

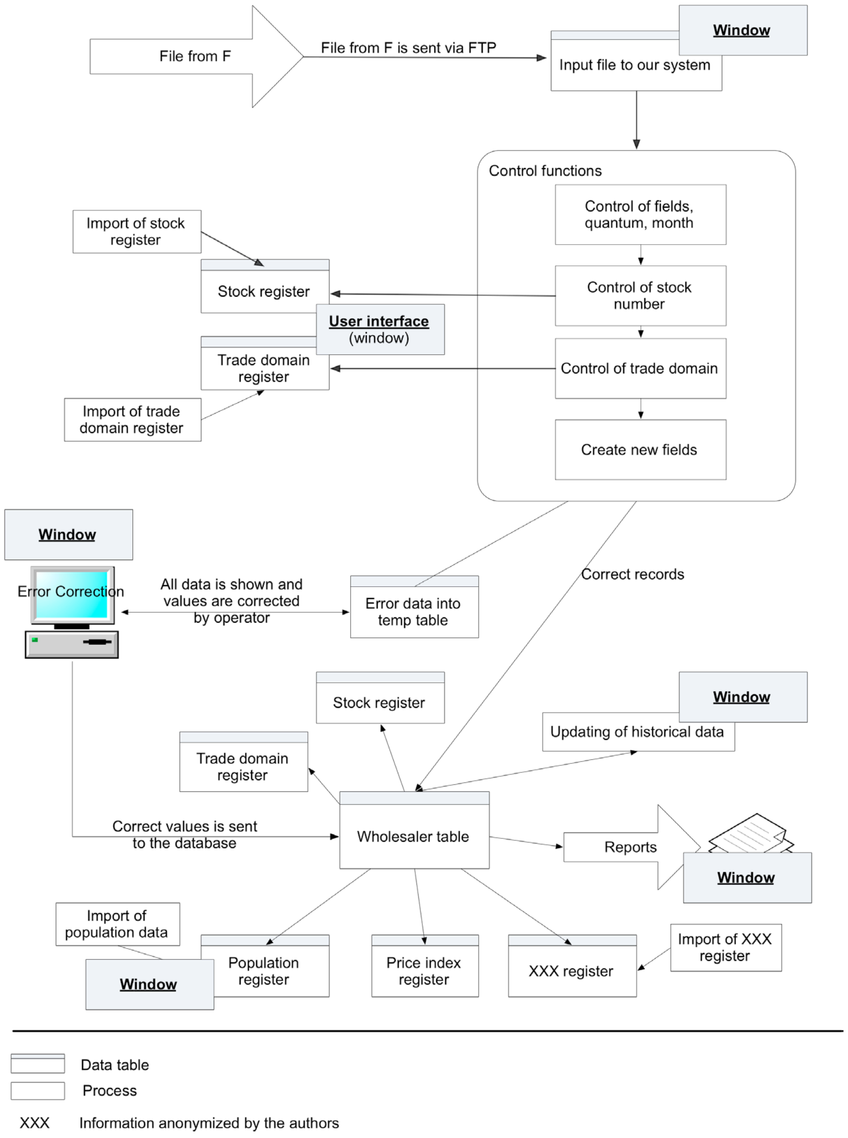

Figure 2: Diagram of the software system adapted from the requirement specification[2]

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)	PM takes notes. D uses the calculator.
	(16,0)	
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.

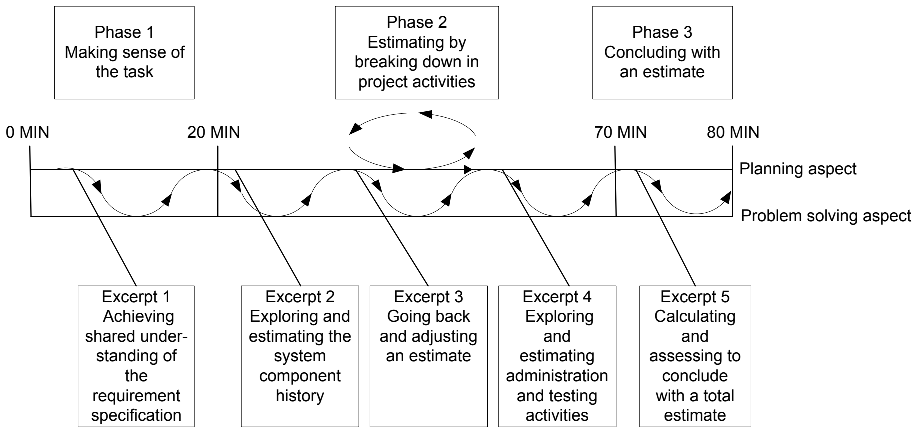
Phase 1 Making sense of the task
Phase 2 Estimating by breaking down in project activities
Phase 3 Concluding with an estimate
0 MIN
20 MIN
70 MIN
80 MIN
Planning aspect
Problem solving aspect
Excerpt 1 Achieving shared understanding of the requirement specification
Excerpt 2 Exploring and estimating 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 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