JYVÄSKYLÄ STUDIES IN COMPUTING
140

Ville Isomöttönen

Theorizing a One-Semester Real Customer Student Software Project Course

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 140

Ville Isomöttönen

Theorizing a One-Semester Real Customer Student Software Project Course

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Delta-salissa
joulukuun 17. päivänä 2011 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in the building Agora, Delta-hall, on December 17, 2011 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2011

Theorizing a One-Semester Real Customer Student Software Project Course

JYVÄSKYLÄ STUDIES IN COMPUTING 140

Ville Isomöttönen

Theorizing a One-Semester Real Customer Student Software Project Course

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2011

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-4534-3
ISBN 978-951-39-4534-3 (PDF)

ISBN 978-951-39-4289-2 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2011

ABSTRACT

Isomöttönen, Ville
Theorizing a one-semester real customer student software project course
Jyväskylä: University of Jyväskylä, 2011, 189 p.
(Jyväskylä Studies in Computing
ISSN 1456-5390; 140)
ISBN 978-951-39-4289-2 (nid.)
ISBN 978-951-39-4534-3 (PDF)
Finnish summary
Diss.

Project courses in software engineering education have, roughly speaking, as long a history as the term software engineering itself — about 40 years. Several project course models have been described in the literature, including the research target of the dissertation, a one-semester course where students develop software in small groups for real customers. The long history of the research field constitutes a challenge for new research. The research approach of this dissertation is theorizing in the sense of classical grounded theory. The motivation behind this theorizing approach was the possibility of finding fresh viewpoints in a long-established research field and a need to give structure to a research field in which studies are often realized in the form of description. The dissertation consists of two parts. The first part focuses on the course experiences of students. Here, the theory that was developed proposes that students' positive course experiences are boosted when students are given a unique opportunity for a real-world experience. When realism is provided to students, who, as part of their professional development, are in need of such exposure, not only are the students' positive experiences boosted but also their tolerance of negative experiences is increased. As a result, the students' overall course experience tends to become positive whatever the problems that arise during the course. This mechanism enables an objective evaluation of student feedback, and implies that the curricular position of a course may play an important role in the nature of student feedback. The latter part of the dissertation focuses on the operational issues in the running of the course. The aim was to come to know how the course can be managed without annoying project delays. The theory that was developed emphasizes that implementation of the course must match the course context. The teacher is in a key role in seeking to understand the connection between the course context and the implementation of the course, as the teacher only can observe successive projects. The teacher must share these operational considerations with students to promote learning under intensive course work. Throughout the dissertation a lot of attention is paid to theorizing as a research approach.

Keywords: Computing Education, Software Engineering Education,
Capstone Project, Grounded Theory

Author	Ville Isomöttönen Department of Mathematical Information Technology University of Jyväskylä, Finland
Supervisor	Tommi Kärkkäinen Department of Mathematical Information Technology University of Jyväskylä, Finland
Reviewers	Dr. Anders Berglund Department of Information Technology Uppsala University, Sweden
	Professor Sally Fincher School of Computing University of Kent, United Kingdom
	Professor Victor J. Friedman Max Stern Jezreel Valley College Israel
Opponent	Professor Markku Tukiainen School of Computing University of Eastern Finland

ACKNOWLEDGEMENTS

This dissertation owes much to many people. First of all, I want to thank the Department of Mathematical Information Technology for allowing me to allocate a huge number of working hours to this dissertation. Special thanks go to my supervisor Tommi Kärkkäinen, my colleague Leena Hiltunen, and the Dean of the Faculty, Pekka Neittaanmäki. Tommi has been supportive throughout the dissertation process, with respect to both the importance of the issues foregrounded in the research and the approach followed. Leena's dissertation, finished at the end of 2010, was helpful to me in illuminating the multiple traditions of action research. Pekka's supportive framing of the research process was essential in being able to finally complete this work.

I thank all the project teachers who have worked at the department, particularly Jukka-Pekka Santanen for continuous discussions about project teaching, and my former colleagues Markus Inkeroinen, for pointing out many things about project teaching that were advanced in this dissertation, and Vesa Korhonen, for research collaboration. Importantly, I am honored to have worked with so many talented students during the research process.

I warmly thank my reviewers Dr. Anders Berglund from the Uppsala University, Professor Sally Fincher from the University of Kent, and Professor Victor J. Friedman from the Max Stern Jezreel Valley College. Their expertise in computing education research significantly helped in improving the writing of the dissertation during the final stages of the process. I also thank my roommate Paavo Nieminen for his helpful comments on my think-aloud writing and Michael Freeman for his proof reading.

I thank my family, my kids Onni and Alma, for their interest in their father's endeavors, and for tolerating a somewhat stressed father at times. Lastly, but no less importantly, I express my gratitude to my long-time friends Harri Parkkinen and Eero Siipola for their continuous support and unconditional friendship.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

This dissertation aims to contribute to the several decades-long tradition of student project studies in software engineering. Student project courses, often referred to as *capstone projects* when regarded as milestones in curricula, have been studied not only for a long time but also in great quantity. It is widely agreed that project courses are a means to impart a real-world basis to a curriculum. They represent learning by doing, which in the case of John Dewey's work dates back to the early 1900s (Grant, 2002).

1.1 On the research area

The long project course tradition encompasses many landmarks worth mentioning. An early comment on imparting realism to computing education is found in the ACM 1968 curriculum recommendation (Atchison et al., 1968), which proposed a real programming experience for computing students in order to increase students' competence. As the first "special project course", the historical review by Tomayko (1998) refers to a graduate course instructed by Douglas T. Ross at MIT in 1968 (Ross, 1989). According to Tomayko this was a practical project course tailored to fit the academic teaching tradition. Another pioneer is Peter Freeman who has reported experiences with project courses at the University of California, Irvine, in 1970s. In (Freeman, 1976) he reports on a quarter-long (10-week) course that allows students "to bring together several things they have learned". This integrative character of hands-on project work is now a well-known goal/option for a project course (Fincher et al., 2001, p. 57). A lot of work has also been done at the Software Engineering Institute (SEI), whose materials and the publication series *SEI CSEE* (which have evolved into the currently active CSEE&T conference) report teacher experiences and present some taxonomic frameworks on project course models. A real-world basis with real project customers is clearly preferred in these studies, e.g. (Tomayko, 1987; Knoke, 1991; Moore and Potts, 1994). Much later, the SE2004 recommendation (Press and Press,

2004) advocated the inclusion of a significant real-world basis in the curriculum. Project-based studies, capstone projects, and internships were given as a means to implement this.

Despite the long tradition, it can be observed that papers highlighting project courses and the real-world basis continue to appear in the literature. For example, Bridgeman's (2008) paper confirms that writing on student projects is active. His paper reports on an active community in New Zealand. Generally, it is not difficult to find a very recent paper in which a project course model is described (Mochol and Tolksdorf, 2009) or particular course design questions discussed (Richards, 2009). It seems that practically oriented project courses continue to attract researchers who often act on a "teacher as researcher" basis, as is also the case in this dissertation. Tomayko (1998) has criticized the situation by noting that the CSEE&T conference keeps receiving papers from isolated academics who submit project studies as if they were something new. Such a comment hints that the field may lack a mature research culture that properly addresses previous studies. In fact, the quality of software engineering education research has been criticized in this respect (Lethbridge et al., 2007). If the field is continuously studied, and in this manner, the literature is likely to become vast but also disconnected.

The paper by Bridgeman (2008) reports on a Google Scholar search (May 2008) with the key words "Capstone Projects"[1]: 1,110 responses were received. I repeated the search (June 2009) and received 1,610 responses. The keywords "Capstone Projects" + "Software Engineering" received 686 responses which seemed a bit more manageable, but the words "Team Project" + "Software Engineering" then received 1,960 responses (June 2009). While such searches are of course only rough approximations, it seems that the literature touching in someway on the subject is vast, and it thus becomes a somewhat burdensome task to capture and grasp all of it.

The vastness of the literature is probably explained by the fact that the *software project* itself encompasses a myriad of themes that are worth studying, and when this multifaceted concept/phenomenon is transferred into an educational context, the number of themes that must be considered becomes multiplied as well; see Clear (2001). The disconnectedness is demonstrated by the several different terms employed when referring to project studies. To mention but a few, the terms *team project* (Collofello and Ng, 1999), *capstone project* (Keefe and Dick, 2004), *classroom project* (Shukla and Williams, 2002), and *project-oriented work* (Melin and Cronholm, 2004) have been utilized. While these concepts have their own nuances, they often cover overlapping educational concerns. Where a particular course or program level arrangement is stressed, a more specific concept, such as *software studio* (Tomayko, 1996) and *software factory* (Tvedt et al., 2002), has also been coined.

Given that the field has a long history and that it continues to attract teach-

[1] Notice that the term capstone project particularly refers to a project course model that aims to integrate students' previous learning (Fincher et al., 2001) while it also appears to be generally employed when speaking of student projects.

ers, let us proceed to another aspect. With respect to project studies in software engineering, one could reasonably argue that the academy follows the industry. For example, in the late 1990s, project studies included recommendations such as "require and evaluate design documentation before allowing coding activities" (Robillard and Robillard, 1998). The study compared academic practices with those needed in the industry. Another example is the rise of Extreme Programming (XP), which soon extended to academic projects, e.g. (Kivi et al., 2000; Müller and Tichy, 2001; Dubinsky and Hazzan, 2002; Umphress et al., 2002; Hedin et al., 2005). Perhaps a more topical issue is that distributed software business gives rise to distributed student projects, e.g. (Last, 2003; Petkovic et al., 2006). Project courses in software engineering have thus reflected changes in the industry, and research has been activated along with such changes.

In a similar manner, it can be observed that project course studies follow topics in general education. That is to say, project courses are being described and analyzed in the context of educational concepts, of which problem-based learning provides a good example (Newman et al., 2003).

1.2 Research target

This dissertation studies a semester-long real customer project course, offered at the Department of Mathematical Information Technology, University of Jyväskylä. This two decades-old course principally exposes students to a department-located (local) software project as a substantial part of their master's level studies.

In the light of the observations in Section 1.1, in particular the long project course tradition and the recent changes in the software industry, the research target of the dissertation is a challenging one. There is the challenge of justifying a study on what some might call a thoroughly dissected educational innovation, which again implies the challenge of making a genuine contribution.

1.3 Motivation and aim

Research on student project courses in software engineering sometimes differentiates between course arrangements in a problematic manner, which is here referred to as "juxtaposition". In particular, I refer here to cases where a particular course arrangement or model is preferred (ranked) over some other arrangement: e.g., real projects over toy projects, real customers over academic customers, reverse engineering projects over from-scratch projects, large teams over small teams, etc. This might occur because teaching is a sensitive issue. Teachers defend what they do and deal with, and they tend to look at the brighter side of the picture. This may also be a tradition in the research field, as it is a mode I can retrospectively recognize in myself, having worked as a teacher-as-researcher during this dissertation process.

This dissertation does not argue for particular course arrangements. It attempts to reveal what is going on in a particular set of arrangements. The theorizing put forward in the dissertation does not imply that other course arrangements are worse. In general, when superiority is claimed, it probably holds only with respect to a specific viewpoint. I will give an example. I have previously argued for a small team size in student projects because this provides each student in a group with a meaningful responsibility, hence reducing free riding (Isomöttönen and Kärkkäinen, 2008). This is certainly not a new finding; see for example (Freeman, 1976). On the other hand, Tomayko has criticized small teams since one improperly attending student can spoil the project, and argues that large teams better reflect industry practice (Tomayko, 1987, 1996). One can immediately recognize at least two viewpoints (concepts) that explain the advocacy of a particular team size: *amount of free riding* and *correspondence to real software engineering work*. Both of these viewpoints are important and thus there is no need to compare on the grounds that one is superior to the other. Argumentation coupled with explanation (knowledge gained by teacher reflection) is what teaching and the field can benefit from. The action research community acknowledges the reflection of teachers as an important tool in educational improvement, e.g. (Klafki, 1988, pp. 236, 243). In this spirit, the present dissertation attempts to focus on (holistic) knowledge instead of juxtaposition.

To be more precise, this dissertation aims to lay out knowledge in the form of theory. It was proposed in Section 1.1 that the project course literature is vast and project courses are described with overlapping terms. By this I refer to the descriptive nature of the literature, and argue here that the field needs its own theory to complement the existing decades-long project course literature which contains lots of results in the form of description. That is to say, research needs to be able to generate analytic and reduced conceptualizations[2] on relevant problems. Carefully constructed and sufficiently reduced conceptualizations provide a rationale and a tool to design and improve course arrangements. They are easy to remember and thereby make the field more accessible for practitioners. Moreover, such a theory orientation would enable computing education to be better identified as a professional occupation, as opposed to a non-professional activity that does not build upon its own theory (Carr and Kemmis, 1986, pp. 7–8).

The local motivator of the dissertation is the fact that the projects undertaken during the history of the course have suffered from serious delays (Santanen, 2008), and that this dilemma needed to be resolved. Opinion on how to interpret and manage such problematic situations has varied among the teachers implementing the course. It is clear that, like other courses, the same task necessitates a different effort from different students. This has been the argumentation behind interpreting the time constraint loosely. Interestingly, this may be a widely held opinion. Todd et al. (1995) noticed in their survey on engineering

[2] *Analytic conceptualization* in this dissertation refers to the use of concepts that, in terms of grounded theory, characterize entities rather than constitute entities themselves; see Section 3.1.4.

capstone projects that respondents often stated that “students were expected to complete the job regardless of how much time it took”. In this dissertation, the one-semester period and a limit on maximum working hours were taken as fixed constraints. This strict interpretation of the time constraints was a part of the research strategy, as will be explained later, but it also relieves students from study design problems, and, most importantly, it removes the ethical concern of using inexperienced students as free labor. From this perspective one can identify an emancipatory character in this dissertation. It should be noted here that it is unlikely that project deadlines are walked over in the industry; project delays must be somehow discussed and resolved. This industrial realism is in line with the position taken here.

Yet another important motivating factor is that the industry needs people who are able to work with limited resources (Begel and Simon, 2008), and thus capable of working under stress (Vaughn, 2001). From this perspective, the one-semester model, which is known to be a tight timescale for an educational software project, is worth studying: real customer expectations and the one-semester time constraint entail the potential to enhance students’ tolerance of uncertainty and stress. However, short project courses with a real-world basis have been accused of leading to problems (Bothe, 2001; Pierce, 1997). Many institutions provide software project courses that extend over several terms and suggest that a realistic experience actually necessitates a project extending over several terms; see Suri (2007). Obviously, the reasoning here is to be able to cover the whole software life-cycle, but this kind of argumentation, if strongly put forward, looks like juxtaposition. The strategy followed in this dissertation was not to put aside the one-semester constraint but to come to know how this constraint can be satisfied.

In the light of these motivational aspects, the aim of this dissertation is to provide analytic conceptualizations, viewpoints on and derived from a one-semester real customer project model. This should help in implementing such a course model. This is argued to be important both locally (with respect to the emancipatory starting point, as discussed above) and in the context of project course research and education (with respect to theory building and the needs of the industry, as discussed above).

1.4 Research approach

The research approach of the dissertation is thus that of conceptualization which I also call *theorizing*. For me, the latter term reflects on-going activity with the aim of developing theories that are centered around specific themes.

There were no specific research questions. All the research in this dissertation started from my course development work principally carried out during 2005–2008. This development work was basically interested in anything that could mitigate the tension involved in terminating projects on time. This immediately led to a holistic research approach as opposed to a study of a single software project issue (e.g. requirements management, becoming a team, etc.). In this connection, the dissertation consists of the two research themes that were arrived at. A research process of this kind is recognized in the grounded theory literature which suggests that research starts from a general area of interest, more specific questions emerging from the data during the process of theory building (Adolph et al., 2008).

The first research theme characterizes the overall impact of the course. It proposes that the real-world basis inherent in the one-semester real customer course setting has an enduring value for inexperienced students — whatever problems arise during the course. I considered this a fresh viewpoint that needed to be explicated. For me, this demonstrated that it makes sense to conceptualize not only in the context of new research areas, but also within the ones that have long been studied. Having a personal relation to the research context and trying to make sense of the data at hand led me to this viewpoint which was then refined in order to be able to generate explicit knowledge. This research theme is based on the classical form of the grounded theory method (Glaser and Strauss, 1967).

The second research theme focuses on conceptual ideas concerning how to manage the course. Here, the present research aimed at change that can overcome the tension in project completion and thus avoid annoying project delays that risk student rights. The focus of the study changed several times (efficient project setup, communication issues, process issues etc.), and finally certain parts of the results that provided a holistic and sufficiently consistent operational view on the studied course model were included in the dissertation. This theme is based on action research (Lewin, 1946) complemented with conceptualization similar to grounded theory, and is directly connected to the course development work mentioned above. It must be noted that the present action research emphasizes theory development and theory as the research outcome.

Both of the research themes thus draw on theorizing and are influenced by classical grounded theory. In the educational ICT context, the classical grounded theory method has been adopted by van Niekerk (2009). In her work, theory development similarly takes place in the action-taking context where the researcher teaches and the theorizing concerns the teaching. Her principal research method reference is classical grounded theory. In my work I decided to refer to both grounded theory and action research, as the theorizing concerning the second research theme directly relates to action taking motivated by the need for a change.

It is particularly important to note that the theorizing approach adopted here has an *on-going* nature. It can be characterized with the grounded theory metaphors *theory as a process* and *ever-developing*. This has two major implications for the manner of presentation in this dissertation:

- Throughout the study the manner of presentation is *discussional*, which is in line with recommendations concerning theorizing-oriented research (Glaser and Strauss, 1967, p. 32): a discussional manner demonstrates the on-going nature of theorizing.

- Contextualization of the present research is done in close relation to the introduction of the research context, research methods used, and results. Hence, there are no separate chapters that would serve as theoretical frameworks at the beginning of the dissertation. In contrast, contextualization in this study takes place through *comparison*. Comparison creates knowledge and contextualization through comparison demonstrates the on-going nature of the research approach.

Examples of the latter implication are given below.

- The research context in Chapter 2: an approach where the research context is first described and then compared with the literature is chosen. It should also be noted that because of the theory orientation of the present study, taxonomy-like literature is preferred when discussing the research context: taxonomies foreground the contextual features of course projects in relation to educational goals.

- Research methods in Chapter 3: grounded theory concepts are first reviewed after which the adoption of grounded theory in computing education research and in this dissertation is considered in the light of the grounded theory concepts. The theory orientation of the present action research is first argued for in relation to the characteristics of the research setting and then rationalized by finding similar approaches in the action research literature.

- Results (theories) in Chapters 4 and 5: selected parts of the literature are reviewed at the end of the results chapters, in the context of the local theories presented. This arrangement allows for close comparison between the local theories and the literature, and this also corresponds to the temporal order in the research process: the local theories were developed first and then viewed in the context of the literature. Here, the literature is viewed as data and comparison between the local theory and the literature can advance the local theory in a discussional manner, demonstrating the on-going nature of the research approach. The inclusion of the literature in this way is thus found in both the results chapters of the dissertation, within both the theories that correspond to the research themes described above.

  Similarly, at the end of each of the results chapters, I return to methodological questions in a reflective manner. These reflectional sections advance the methodological discussion of the dissertation and also serve as discussion on validity. Having first read the theories developed (the results), I argue, should help the reader in considering validity in the context of research which is not based on controllability and repeatability but theoretical sensitivity; see particularly Section 4.4.

To argue for this research approach, I refer to the original formulation of grounded theory (Glaser and Strauss, 1967), which I see as legitimization of this kind of research. To argue for the manner of presentation, as described above, I also refer to

Piantanida et al. (2004). In pondering the essences of grounded theory, they suggest that the traditional format of a research report is atheoretical. With this I can easily agree from personal experience (Isomöttönen and Kärkkäinen, 2009). As I mention later, research conducted with classical grounded theory appears to be very implicit. Therefore, treating such a process in a mechanistic and a stepwise manner of presentation feels like approximation and forcing. In saying this, I am fully aware that this dissertation may appear "different" if viewed in the context of a more traditional qualitative research process. The literature examining the use of grounded theory suggests that theorizing is also a difficult approach (Adolph et al., 2008; Suddaby, 2006).

While the above considerations underline theorizing as the dominant research approach in this dissertation, the emancipatory characteristics of the dissertation must also be acknowledged here. Tension in project completion and the associated problem of project delays are a difficult state of affairs for students and also for the Department. In the course of the dissertation I come to the conclusion that students are not always aware of the causes of project delays or may live with such a problem due to obligation, in order to be able to complete a substantial mandatory course in the degree; there likely exists "silence" which should not be overlooked. The emancipatory characteristics of this dissertation refer to this issue and the associated necessity of change. However, my principal aim in this dissertation is to develop theory that can be utilized to resolve the problem. It is for this reason that I do not include much detailed discussion of the problem or ground the dissertation in the critical tradition; see for example (Habermas, 1979; Mezirow, 1981; Hart, 1990) — such a contextualization would suit the present task but is left for future research. The emancipatory characteristics of this dissertation are particularly present in the action research part (Chapter 5) — either implicitly as a motivating factor or directly illustrated with examples from the data. The emancipatory characteristics also have an impact on the manner of presentation. Namely, I find myself writing in the form of taking positions; this can be found, for example, in the concluding chapter of the dissertation.

Having stated all the above, I recognize that the approach taken and followed can be viewed as having connections with several paradigms. Compared to Guba's and Lincoln's (1994) taxonomy, the present aim of theorizing involving systematicity and generalization, while not adhering to fixed conceptualizations, savors of post-positivism. On the other hand, the need to provide an alternative to the current state of affairs, i.e., to demonstrate how the one-semester model can be managed without severe ethical drawbacks, has the value-loaded savor of critical theory. Furthermore, the project model conceptualizations presented in the dissertation have clearly evolved over time, and locally among the people involved. I have had frequent dialog with both students (actual teaching situations) and the teacher colleagues (reflective discussions). Such a continuous and gradual sophistication of knowledge savors of constructivism.

Above all, I found that I arrived at the position of a critical realist during this research, which, I argue, provides a stance in which all of the present research can be situated. The realist view of ontology differentiates between the domains of the actual, empirical, and real (Bhaskar, 1978), and research conducted on this ground attempts to reveal the real mechanisms when explaining social processes (Nash, 2005). In this connection, this dissertation is concerned with mechanisms that were arrived at through theorizing effort.

1.5 The term ’theory’ in this dissertation

I proposed above that the field needs its own theory and emphasized theorizing, on the grounds of critical realism, as the dominant research approach of this dissertation. It is necessary here to discuss what I principally mean by *theory* in the context of this dissertation. I speak of theory in the sense that is utilized when speaking of (classical) grounded theories (Glaser and Strauss, 1967). That is to say, a theory refers to a conceptualization which has a focus. It is positioned between every-day working hypotheses and grand theories. Such a theory already has a life apart from facts, which means that it is no longer a description of events but a more abstract construct. It is about concepts and their relationship. It may be associated with the constructs such as a model, a typology, a pattern, etc.

A taxonomy of project course models would already resemble theory in the sense described. A project course taxonomy is a conceptualization of a focused area; see, for example, the project course taxonomy by Fincher et al. (2001) which include entities such as “a project with a real client” and “a capstone project”. Having such a taxonomy in mind helps in situating a project course description found in the literature into a larger context. In the same way, patterns in students’ perceptions or patterns in teachers’ and students’ conventions can be foregrounded as theory, and these would likely help in managing teaching. Theory is thus not referred to as a grand theory, as a grand generalization, nor associated with the term “literature” or “related work”. As I speak of theorizing with an ongoing property above, I similarly associate the on-going property with the term theory; thus it is not viewed as a fixed generalization.

It should be noted that in Chapters 4 and 5, when I refer to the theory developed in the course of the dissertation, I frequently use the expression “the present theory”.

1.6 The structure of the dissertation

Chapter 2 presents the study context, the one-semester real customer course *Software project* `TIES405`, offered at the Department of Mathematical Information Technology, University of Jvyäskylä (JYU/MIT). Chapter 3 introduces the research methods used. Chapters 4 and 5 lay out the theories generated in the research. Chapter 6 provides both a discussion and conclusions and seeks to synthesize the conceptualizations offered in the results chapters.

2 STUDENT PROJECT COURSE AT JYU/MIT

This chapter describes the research context of the dissertation: the student project course *Software project* `TIES405` at JYU/MIT. The course code has changed during the course's history, but in the following the present code — `TIES405` — is utilized throughout. The chapter starts by outlining the background of `TIES405` in Section 2.1. The present course arrangements are then detailed, starting from Section 2.2. In order to position `TIES405` into a larger context, the literature on the taxonomy on project courses and project course arrangements is reviewed and discussed in relation to `TIES405` in Section 2.8. Summarizing comments follow in Section 2.9.

2.1 Background

Santanen (2008) outlined the two-decades long history of `TIES405`. He divided the course history into four periods and the present. The four periods are summarized in the following.

During the initial phase 1988–1989, the course produced software for research and teaching purposes. Project issues were sourced from the department's personnel and were oriented towards scientific computing needs. Customers or end users were not necessarily involved. Project planning was not required nor were there any regular project meetings. However, time logging was required from students who themselves formed the project groups. The outcome of a project consisted of a software product, a commented code, and a handbook documenting the software structure. The development process itself was not documented. The main learning goal was to enhance students' programming skills.

Later, during 1990–1994, software was mainly developed for university units, although some of the customers came outside the university. Software was developed for more real-world purposes, yet often to support teachers' and students' work. Not every project included programming but produced user instructions or some reference material for students. More attention was given to team work and project management, as it was noticed that project success relies on other than technical issues. The clientele became more diverse and it was noticed that projects required specific technical supervision. Some of the groups were provided with a specific workspace. As opposed to the first period, teachers started to form groups based on course registration information. The outcome of a project now also included a final report on both the project and the software (in the same document), accompanied with some reflection. It can be concluded from (Santanen, 2008) that the period was about taking steps towards a systematically and explicitly managed project course.

The latter two periods, 1995–2000 and 2000–2005, are more relevant to this dissertation because the data mostly originate from the years 2000 and later. Since these periods, the course was coordinated in an orderly manner and bills began to be sent to the customers who were now more carefully screened to have a real need for software. During 1995–2000, the inquiry of project topics became a systematic process. Each student group was provided with a specific workspace where they could work autonomously. Each group had both a corresponding teacher and a technical supervisor. External customers (firms) and university participants (teachers and students) agreed on responsibilities and constraints with a project contract. External customers were billed for the projects. The course started to pay attention to software processes and weekly project meetings were arranged. The outcome of a project was no longer software-centered but all student groups were expected to plan the project and prepare final reports. Along with the new study subjects in the department, heterogeneity was increased in the project participants.

Towards the present, during 2000–2005, project topics became more complex and critical, and systematicity in how to coordinate the course was further increased. Collaboration between project teachers was increased and systematized, and, as a result, teachers documented the management processes needed in the course coordination. Inspections of project deliverables were also increased. The outcome of a project included software, a project plan, a requirements specification, a software design document, final reports on both the project and software, and often test case specifications, user instructions, and installation notes. The students' work was supported with lectures on version management, usability, and immaterial rights. During this period, also customers from the public sector were increasingly billed for projects.

Santanen (2008) stated that from its very beginning, `TIES405` has been a one-semester course employing a small 3–4 person team size and active instruction. These permanent course arrangements were soon complemented with real customer collaboration, introducing real expectations into students' course work. Overall, roughly half of the project customers have been firms, one-tenth municipal and other public sector units, and the rest university units (Santanen, 2008). For this wide range of customers the nature of the project topics has varied considerably. Topics have been database-www applications, configuration software for highly specific equipment used in customers' processes, mobile phone applications, educational games, testing automation software, etc. The recessionary period cut down commercial customers at the beginning of the 2000, but the situation had improved within a year (Santanen, 2008). At the moment, the student decline in computer science is observable in the course. The number of the groups per semester has halved during the last five years. This decline is also due to the fact that since autumn 2005, the course has been mandatory only for software engineering majors. The following sections outline how the general `TIES405` course management arrangements stand at the moment.

2.2 Goals

`TIES405` exposes students to an "authentic software project with real customer" experience, and in this setting, attention is paid to both the end product and process. In the present form of the course, a software development component is always included. According to Santanen (2008), the educational goal is to enable students to accumulate experience and skills regarding project work, team work and leadership, project management and quality, specification and design, implementation and testing, written and oral communication, and documentation. The student data analyzed in this dissertation confirms that students develop and enhance the wide range of skills inherent in a software project. Straightforwardly, as the goal has been giving students a realistic software project experience, the projects have followed the acknowledged phases of software development work in an authentic team work context. It is acknowledged in the literature that project goals have a relation to the characteristics of the project. Fincher et al. (2001) note that "function drives the form" and Clear et al. (2001) that "given a lucidly enumerated set of course goals it should then be possible to characterize appropriate and inappropriate capstone projects". Burge and Gannod (2009) present relations between different project types (cf. goals) and project-, development-, and process -related dimensions. More generally, in the field of education, it has long been acknowledged that the whole curriculum is seen as a means to given ends (Tyler, 1949).

It is noted here that the collaboration with real customers introduces variation into the course context and the skills needed. Some groups work upon customers' existing software while some start from scratch. Some must conduct a feasibility study at the beginning of the project, which introduces a slight research character to the project. Some focus on software maintenance by importing an outdated system to a new platform and adding new functionalities to the system.

It is also noted that the present research introduced a change in the groups supervised by the dissertation author. Appropriate learning goals became a major concern in this dissertation, and a shift away from the emphasis on a realistic experience and teaching of skills will be outlined in Chapter 5.

2.3 Project stakeholders

Students. At JYU, students receiving a place at the university are directly accepted for both bachelor's and, the ensuing, master's studies. Thus there is no clear division between a bachelor's and master's program but in practice the students study a coherent, ideally five-year, "bachelor-master's" program. In our department, specialization in a particular subject (study line) takes place in the course of master's studies. Computer science students thus study a common bachelor program which is directly followed by master's level studies, with a choice of four study lines: simulation & optimization, mobile technology, software engineering, and teacher education. `TIES405` was first positioned in the advanced bachelor's level studies, but was moved to the master's level studies in autumn 2003. Project teachers noticed a slight effect on the student material, the subsequent project students tending to be a bit more experienced and mature.

In practice, the course enrolls students who are finishing computer science courses at the bachelor's level. These students have typically taken programming courses, algorithms and data structures, basics of databases, basics of web technologies, object-oriented analysis and design, an introductory course on software engineering, and some courses in math. At least half of the computer science bachelor courses, including key programming courses, have to be completed before acceptance into the course. While the students have often some experience in programming, majority of them have no or little previous software project experience. This inexperience is explicitly stated by the students in the course feedback data.

The student teams are local and mostly consist of the department's own students. Software engineering majors have started to dominate the course, after the course was made mandatory only for them. Some students still come from the other study lines, and the students majoring in science — math, chemistry, or physics — occasionally apply for the course. Some of these science students are considering whether they would like to major in physics, chemistry, math, or computer science, and attend the course to gain authentic experience in computer science and software engineering.

The student teams consist of three to four students. Four is the preferred team size for team experience and always used when possible. The team selects one of its members as a team leader, but the responsibilities are shared so that the team leader is not the only member in a responsible position. When the students plan their project they are prompted to assign roles and responsibilities to all team members.

Customers. The course solicits customers from both industry and the public sector. Customers are usually sourced from the local area but a few projects have been implemented in collaboration with a customer from another city. Customer organizations have been of different sizes, from small one-man companies to big software houses. The customers attend as real customers instead of merely issuing a problem statement. Typically one or two customer representatives attend the project actively. External customers pay for the projects and recently in-house customers are often billed as well. No technical knowledge or other experience in software customership is expected. Demands and constraints on customership are further described in Section 2.4 on project selection and contracting.

End users. Several projects have had direct contact with end users. Sometimes the end users attend the project actively, as a sub-group of customer representatives, and sometimes a student group interviews end users during the project to collect necessary information. Sometimes it is possible to utilize end users in system testing. On the other hand, end users are often not available or students work more as subcontractors basing their work on specifications that are provided by the customer.

Teachers. Several teachers have supervised projects during the history of the course. The course is coordinated by one corresponding teacher but each of the participating teachers is responsible for the supervision of the project he/she is assigned to. In this kind of authentic project work (cf. pre-packaged scheduled project courses), all the teachers inevitably have their own viewpoints on software engineering work and teaching in such a realistic context. A new teacher is familiarized with supervision work through a semester-long pair-supervision with more experienced project teachers. This arrangement familiarizes the new teacher with the course context and enables the other teachers to efficiently share tacit teaching knowledge with the newcomer. Such collaboration also allows for different views among the teachers to be reconciled.

Technical supervisors. The course employs senior students as technical supervisors. A technical supervisor is expected to know the tools and programming language used in the project he/she is assigned to. Besides tool and programming support, the technical supervisor is also responsible for code inspections. The technical supervisors are paid hourly wages and receive three credits if they return a reflective experience report on their own supervision process.

Technical consultants. Occasionally, specific technical expertise is needed in a project. Sometimes one of the customer representatives attends the project particularly for this purpose. In some projects, senior students or personnel consult on a project on specific technical issues or arrange an extra tutorial on request.

Computer support. Expertize in computer support is needed in the project selection process at the beginning of each semester. The computer support personnel share their views on the project topics and collect the information needed to prepare project computers with the appropriate software installations. Computer support is available to students on request.

Interactions. The project stakeholders' interactions are depicted in Figure 1. All the primary stakeholders, the student group, teacher, senior student (technical supervisor), and customer representatives meet at least every other week. Typically, a project meeting is arranged on a weekly basis although in the project contract the minimum is usually set at "a meeting at least every other week". Meetings are not the only communication medium but informal communication (face to face, email, phone) is also used. The students' interaction with the customer thus takes place in a realistic manner; it is not fixed or constrained. The

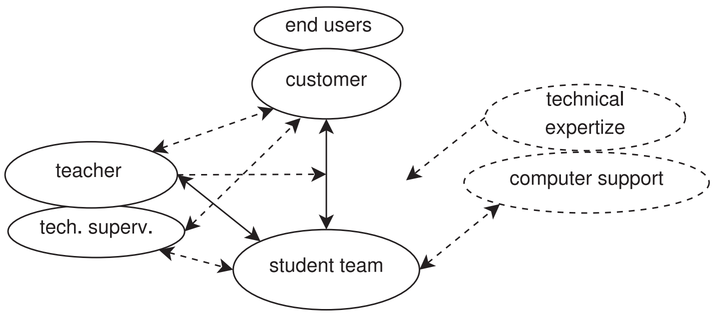

FIGURE 1 Interactions between the project stakeholders

students are similarly expected to autonomously interact with the computer support.

Outside this research, the interaction between teacher and students has largely consisted of review of the process deliverables. Along with the present research, more emphasis was placed on students' abilities when they enter the course and on monitoring and managing the collaboration between the students and the customer, as will be explored in Section 5.2.2 under the title "teaching as coaching".

2.4 Project selection and contracting

The project and student selection processes are depicted in Figure 2. The project selection process starts when the teacher in charge screens customers by contacting local area organizations and inquiring after potential student project topics, usually by email. The teacher then reviews the potential project proposals and leads the initial negotiations with the potential customers. The initial negotiations are typically an iterative process that aims to clarify and refine the customers' needs. The proposals are reviewed to see whether they encompass enough programming and involve a real purpose, whether the technologies involved are such that the university can support the students' work, and whether the customer can allocate sufficient time for the project. At this stage, the project contract template is introduced to the potential customers. Those proposals that are initially judged as applicable are delivered to a working group consisting of staff members.

The working group assembles for a project selection session at the beginning of each semester. The group selects the project topics by voting on the customers' proposals. In the selection session, the group further evaluates whether the proposals are applicable for team work (e.g. that the work can be split between team members), whether the proposals are interesting to students, whether a particular customer is known to bring about a risk (e.g. regarding commitment), and whether the proposal can benefit all the stakeholders (e.g. the students and the university can adopt up-to-date technologies). Taran et al. (2008) present a similar

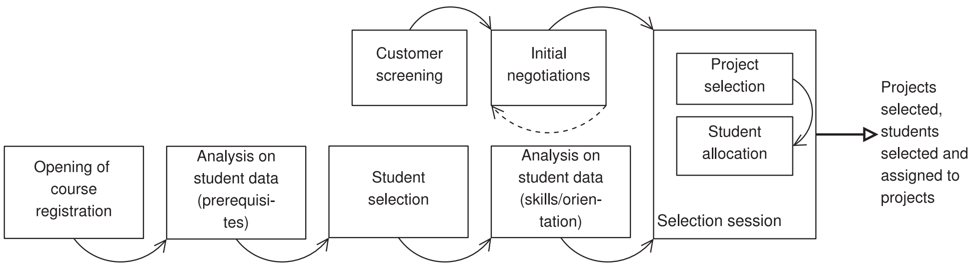
Customer screening
Initial negotiations
Project selection
Student allocation
Projects selected, students selected and assigned to projects
Opening of course registration
Analysis on student data (prerequisi-tes)
Student selection
Analysis on student data (skills/orien-tation)
Selection session

FIGURE 2 Project and student selection processes

project selection process, but with the addition that a project selection committee evaluates the customers and project topics based on a matrix of attributes. Such a visualization, a "matrix view", helps to make a systematic comparison between the proposals.

The student selection process starts with course registration (Figure 2). The students apply for the course by filling in a form that collects information such as study background, personal interests, and whether a student is going to be working or have other intensive courses during the semester. One or more of the project teachers analyzes the registration data to judge if all the registered students are eligible for the course, i.e., they have a sufficient study background and can allocate sufficient time for the project. The project teachers select the students and inform both those accepted and those not accepted. After this the registration data of the accepted students is further analyzed regarding their skills (based on study history and work experiences) and orientation (interests). Finally, the students are allocated to projects by the working group, which also selects the project topics in one and the same selection session (see Figure 2). When allocating students to the projects, the aim is build equally strong groups with plenty of diversity and motivation; this is done on the basis of the students' skills and study orientation and a possible match between the project topics and the students' interests. If possible, friends are not placed in the same group. This strategy aims to create a better communication challenge and experience of team development for the students. The opposite strategy, i.e., allowing students to form groups based on friendship, is also used in the field (Hagan et al., 1999).

The projects with external real customers necessitate a contract. It is necessary to address the transfer of immaterial rights, and agree on responsibilities and constraints in the educational setting. In `TIES405`, a formal and legally binding contract is used. The contract template is publicly available on the Web (in Finnish)[1]. According to the template, immaterial rights (proprietary right, copyright, right to use, modify and deliver etc.) are transferred to the customer. The customer directs payment to the department and can choose from two fixed-price classes. The lower price class is for cases where the results can be archived publicly and the higher for cases where the software and its related material is agreed as confidential, and is not archived by the department and the students. In the

[1] `http://www.mit.jyu.fi/palvelut/sovellusprojektit/sopimus.html`

higher price class, those parts of the results that report on the students' project work (not the product) are nevertheless kept by the department, but checked to ensure they do not include any confidential information. In both cases (price classes) the university and the students have parallel rights to use those parts of the project results that are not agreed as confidential. In addition to the fixed payment, the customer is also obliged to directly provide any specific software licenses needed in the project.

Several other issues are agreed in the contract, including software maintenance (no warranty is given on students' work and maintenance is left as the customer's responsibility), how to operate if disagreements are encountered, how to manage changes, etc. Important in the agreement is the active collaboration and problem domain specific supervision required from the customer. The contract template maintained by the teacher in charge, Jukka-Pekka Santanen, is the result of a continuous organizational learning process. It is principally shaped by the experiences of project teachers.

The students are asked in advance (with the course registration form) if they are willing to allow the transfer of immaterial rights. Open source topics are selected when the students do not want to transfer their immaterial rights to the customer. In these cases, the customers have typically been from within the university.

A change that took place with the course development work was that the students are now also asked in advance *whether* and *what kind of* nondisclosure agreement (NDA) they are willing to sign if they are to work in a project which necessitates NDAs. The students are advised to contact the teacher in charge if any questions arise in providing these answers. Another change is that students are liable for damage only if their malpractice is intentional. These issues are not included in the results chapters of this dissertation.

2.5 Schedule, workload, and events

Each student spends typically at least 10+ hours, usually 15–35 hours, per week on the project. The projects last around 16 to 17 weeks. In the autumn, the projects start in the middle of September and end at the end of January. Christmas provides a break of about two weeks. The spring projects start at the beginning of February and end at the end of May. According to the present contract model, one additional month is reserved for negotiations if serious unexpected problems occur. The fixed project termination deadlines are then the end of February (autumn semester) and end of June (spring semester). The projects supervised in the context of this research ended by the end of May (spring semester) and January (autumn semester).

Each student is required to spend at least 200 hours but no more than can be compensated with credits (400 hours). On top of this, compensation for the few lectures, tutorials, and presentation rehearsals included in the projects is given in

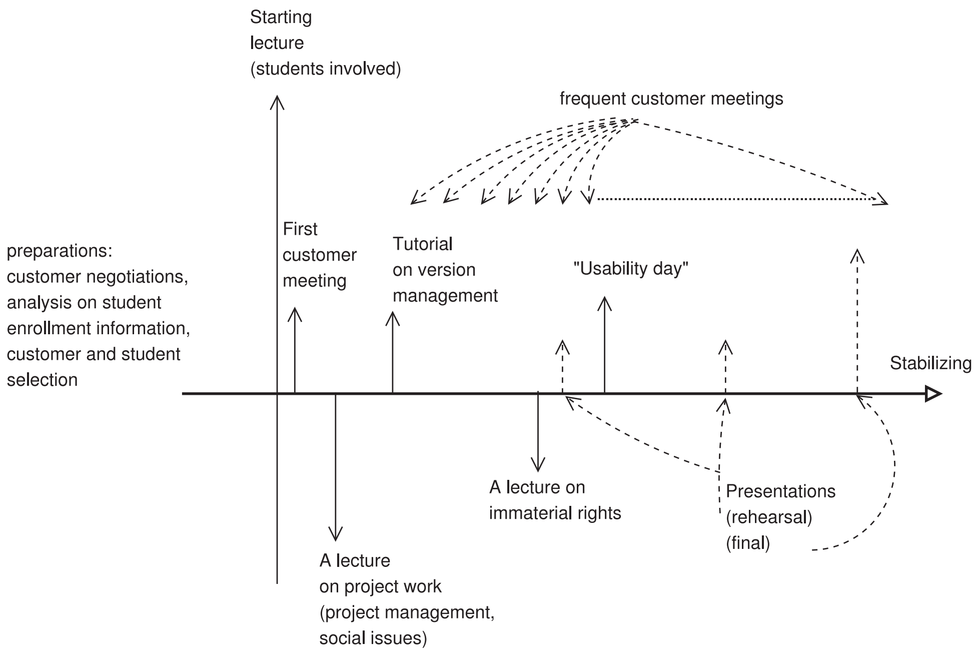

FIGURE 3 Events during one course offering

the form of a supporting course, entitled as Management, Communication, and Tools in Software Project. The credits given depend on the hours spent, varying from 10 to 15 for the project course, plus fixed three credits for the supporting course. Recently, some of the supporting course resources have been dedicated to a specific communication skills course. Overall, the workload arising from the project and the supporting courses can be about 500 hours in total.

Figure 3 summarizes the events occurring during a single course. Because the situation is real the actual dates of the meetings, lectures, tutorials, and presentations are not fixed but determined on the fly each semester. In addition to what is described in the figure, students work autonomously with the projects on daily basis. The lecture-like events are explained below.

Starting lecture The first event involving students is the starting lecture. In this lecture students are informed as to which group they belong to, the customer and the project topic they are to work with, and general information on carrying out the course. The issues that the students will encounter at the outset of the project are given specific attention while the students are also given documented information regarding the whole project. After the starting lecture, the students are given keys to the project rooms. They study the short one-to-three page description of the project topic, the same project proposal that was used in the project selection process, and prepare for the first meeting with the customer.

Project lecture The students attend a mandatory project lecture which is arranged within two weeks after course start-up. The lecture focuses on software processes, project management, and soft issues such as personality types. This lecture is part of the supporting course.

Version management tutorial The students are expected to use the version management system. Usually within two weeks of course start-up, a short tutorial is provided with hands-on support for adopting the system (SVN[2]).

Lecture on immaterial rights A lecture on immaterial rights is provided few weeks after course start-up. Because the students working with external customers sign a project contract and because the projects often deal with software licenses, information in these areas is given during the project course. This lecture also forms part of the supporting course.

Usability day One day of the project is dedicated to usability. A usability expert gives a lecture which is followed by group exercises. The lecturer also gives detailed feedback on the user interfaces being currently developed in the projects. This day is also part of the supporting course.

Presentations The students practice twice during the project how to give a project presentation. Only the teachers and project groups attend these sessions, which provide the students with a relaxed context in which to experiment with presentations. The groups receive feedback on their presentations from both the teachers and other groups. Often lively discussions follow. The aim is to develop presentation skills towards the final project presentation which takes place at the end of the semester. The final session is public, and it is advertised on the Web, in the university's announcement section. Project customers attend the affair. At the time of writing, the rehearsals of the presentations were moved to form part of the supporting communication skills course.

Despite the few lectures, the course can well be characterized as a "project only" model (see Section 2.8). Thus, the few lectures and tutorials provided in the mode of project education are meant to advance the projects.

2.6 Project outcome

This section describes a typical project outcome which at the same time describes the activities the students deal with during the project. Most projects are completed with the following deliverables which are handed out to the customer in a single folder (everything printed) and on CD:

- Software: installable software delivered on CD. Because of the time constraint and the use of real (read: large) project topics, the software is often a prototype.

[2] `http://subversion.apache.org`

- A source code listing: source code is reviewed several times during the project and the final version is printed, often with class documentation (using e.g. JavaDoc).
- User instructions: a specific document and/or a part of the software.
- Installation notes: brief instructions on how to deploy the product; sometimes included in the software report.
- Project plan: includes project background, problem and objectives, organization and resources, process, schedule with milestones, team members' responsibilities, management methods, and risks.
- Requirements specification: including use cases and/or functional and technical software requirements, and prioritization.
- Design plan: including user interface design, software architecture, classes, data structures, etc.
- Project report: a reflective report on the conducted project; the issues initially written in the project plan are analyzed and personal experiences written.
- Software report: a reflective report on the product. The product is described, the solutions made during the project analyzed, and potential issues for future development identified.
- Test plan, test case specification, and a test report: these may also be included in the design plan and software report.
- Agendas and minutes of the project meetings which document all the decisions and deviations during the project: students prepare the agendas and minutes.
- Time logging listings: each student is expected to log time usage throughout the project.
- Email listings: all email communication during the project — each project use their own email list for communication between the project stakeholders.
- A copy of the signed contract.
- Self evaluations. At the end of the projects the students return a self evaluation (usually from one to three pages) in which they comment at least on the group performance, their own performance, and performance of their supervisors. These can be returned confidentially in which case they are not included in the project results. The self-evaluations are read by a teacher after the teacher's initial round of assessment. This enables account to be taken of both the teacher's and the students' opinion in assessment.

- Other: weekly progress reports presented in the meetings, minutes from the usability session and presentation sessions, possibly end user interview data, any material that the customer delivered during the project, etc.

The following is an extract from the paper by Clear's et al. (2001), here given as a reference for the variety of alternatives regarding project course deliverables.

- **Project Deliverables**
  - *What types of documents might be generated by a project?*
    - Documentation of team processes and decision making for the purpose of internal management, e.g., schedules, testing plans.
    - Documents produced for the instructor for educational or assessment purposes.
    - Documentation of events for historical purposes, e.g., personal activity logs, meeting minutes.
    - Documents recording agreements among the parties.
    - Documents produced for formal submission to the project sponsor.
    - Reflective documents, e.g., personal journals, evaluations.
  - *What specific work products are going to be required?*
    - A bibliography of sources and resources for the project under consideration.
    - A project statement or proposal outlining the problem to be solved and the methodologies to be used.
    - Lists of ideas that might be relevant to a proposed research problem.
    - A timeline dividing the project into its constituent parts, showing dates when these parts are to be completed and when associated deliverables might be presented.
    - Design documents. These may differ depending on the type of project type. Examples include UML models, database schemas, flowcharts or data flow diagrams, etc.
    - Logs of administrative decisions, design changes, implementation decisions, etc.
    - Descriptions of attempted solutions with explanations of failure and assessments of efforts.
    - Individual participant logs and journals, possibly reflecting on process oriented issues.
    - The software or system developed by the project.
    - Software or system documentation.
    - Users' and administrators' tutorials, manuals, etc.
    - A final project report documenting all aspects of the project from inception to completion.
  - *What project deliverables should be archived for future reference?*

TIES405 basically involves everything included above in one form or another and during one semester. On top of this, customers attend the inspections of all major work products (software, plans, specifications, and reports) and everything is archived. The project semester is thus very demanding and *intensive* for the students. Clear et al. (2001) state that it is unlikely that everything needs to be archived. The rationale for the total archiving has here been that it lessens the risk of disagreements, since the viewpoints and decisions are carefully documented and maintained in projects that use a legally binding contract.

Because of the varying nature of real topics, the outcome is not totally fixed across projects. Sometimes, at the end of a project, the participants negotiate whether the customer prefers unused resources to be used to prototype new features or to test the ones already developed, and hence, make the software more stable. This affects whether, if any, testing-related documents are part of the outcome. In very large projects, the project participants may notice that the problem at hand necessitates two subsequent projects for completion. The first of such projects serves as a means to prototype and explicate complex business processes[3]. This of course affects the outcome. Recently, there has also been a trend towards the use of project management systems that enable the amount of traditional documentation to be reduced (Trac[4]).

The present research process introduced some change to the outcome of projects. Towards the end of the study, the projects adopted short iterations (cf. waterfall that is illustrated by the given outcome) and the requirements were managed in electronic form throughout the project instead of preparing a requirements specification document. Moreover, the iterative projects used a whiteboard for design so that the design plan then principally focused on initial views of the architecture and the dynamic view of the software. These changes in software process thus affected what deliverables, and particularly in what form, they are produced.

2.7 Physical facilities

The layout of the project premises is given in Figure 4. The premises encompass a shared area that was earlier accessible only to project students but is now open to all the students of the faculty (the large empty area in the figure). The project-specific rooms are found at the back of the premises (the lowest part in the figure). Each room is equipped with personal computers and a whiteboard and is lockable, as the projects deal with confidential customer material. The stu-

[3] Comparison can here be made to Royce's advice to "do it twice": *If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned* (Royce, 1970). Notice, however, that TIES405 project proposals are intended to be stand-alone projects as opposed to an arrangement where students attend to on-going project work.

[4] http://trac.edgewall.org

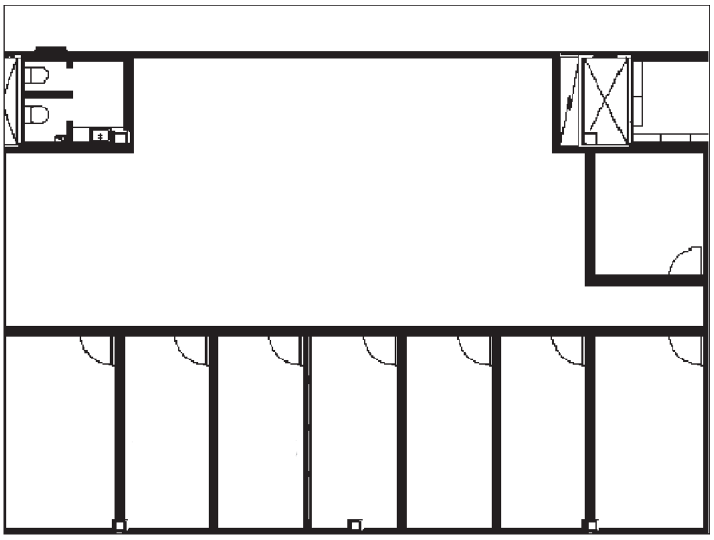

FIGURE 4 Project premises

dents attending the course are provided with the keys to enter the premises and their own room as they wish. The project rooms often become the students' home bases.

A meeting room is found near the project premises, and the items of equipment needed in customer meetings and end user interviews made available to the students. These include laptops, video projectors, digital speech recorders, and a digital camera.

These facilities aim to contribute to authentic team work and support the students' autonomy.

2.8 Comparison to project course models

This section compares `TIES405` with the course models found in the literature. `TIES405` is essentially a department-located supervised course project, for which the comparison will principally take place in the context of such a literature. This leaves out the models that are more directly connected to working life (students are paid and/or placed in real work places). Such models include cooperative education and student companies. Cooperative education models enable students to alternate between school and work, and they thus acknowledge that learning takes place outside traditional university teaching; see for example (Huggins, 2009). Education in the form of student companies refers to a curriculum design which allows students to run (or take part in running) service centers or business, and advance their studies by such undertakings; see for example (Chase et al., 2007; Neagle et al., 2010) and epiGenesys[5].

Within this restriction, attention is given to survey and taxonomy-like literature references, thus to those references that particularly concern structuring the research area. The section first illustrates how prevalent the described `TIES405` arrangements might be in engineering education and then situates `TIES405` in CSE/SEE-specific taxonomies.

Prevalence of `TIES405` arrangements

If software engineering education is regarded as a part of engineering education, a survey on North American capstone courses (senior projects) in engineering (Todd et al., 1995) serves as a relevant reference study. The survey studied how prevalent the variety of project course arrangements are, and it received answers from 360 departments. The results that were considered most relevant to this dissertation are summarized below. I have boldfaced the keywords that correspond to or are close to a `TIES405` arrangement in order to demonstrate how `TIES405` is positioned within the results. According to the survey,

- **one semester** was the most common timescale for a project **(45 %)**. A two-semester model was in second place (36 %), two quarters in third place (11 %), and one-quarter and three-quarter models shared fourth place (4 %). The majority of the respondents were in the semester system, which is here assumed to mean 15 weeks per semester, which is close to the `TIES405`'s four-month period.

- The **project-only** model was **in third place (26 %)** after "class and project in parallel" (48 %) and "class followed by project" (28 %). The "class only" model scored 5 % and the category other 12 %. The category other refers here to models where some classes are provided in parallel with the project at the beginning of the course and the rest of the course is conducted as "project only".

- most often a capstone course was **a team experience located in the single department (83 %)**. The alternatives were the individual (32 %), inter-department teams (21 %), and other (2 %).

- 59 % of the projects were obtained from external parties and 58 % from the department, meaning that **in many cases both sources were used**. The category other (e.g. scientific journals and sponsored research) scored 15 %.

- a **small team** was the **most common team size** (1–3 students: 38 % and 4–6 students: 49 %). A single-student model scored 6 %, 7–9 students per team 5 %, and 10+ students per team 2 %.

- in **48 %** of the courses, students spent **more than 7 hours per week on the project**. The other alternatives were 1–2 hrs (4 %), 3–4 hrs 22 %, and 5–6 hrs

[5] `http://www.genesys-solutions.co.uk/`

25 % *...Many comments were received on this question indicating that students were expected to complete the job regardless of how much time it took.*

- in **33 %** of the courses **sponsor collaboration** took place **on weekly basis**. Monthly collaboration scored 27 %, beginning & final 31 %, and other (variety of patterns) 17 %.
- in projects where funding came from outside the university, **18 %** indicated that funding was based on **direct payment to the university by the sponsor company**, 29 % that the sponsor directly funded project expenses, 28 % that both of these were used, and 26 % that some other funding system had been devised.
- those using industry-sponsored projects indicated that the industry **sponsor retains intellectual property ownership in 40 % of their departments**. The alternatives were that the university owned the intellectual property (32 %), and that the students owned it (33 %). Some other arrangement also existed, or the issue had not been addressed at all (20 %). Ownership was shared in many cases.

Some departments were grouped in the category of computer engineering but whether a software development component was included in the project courses in this category is not given. Thus, while these numbers from engineering education are only indicative regarding computer science and software engineering education, they are given here to demonstrate that the basic arrangements used in `TIES405` are not unique. Notice, however, that the prevalence of such arrangements indicates wide acceptance.

`TIES405` vs. CS/SE project models

Computer science and software engineering education research encompasses several taxonomy-like studies. Shaw and Tomayko (1991) present undergraduate project models in software engineering. Knoke (1991) adds a medium size project in between the small and large scale project models given by Shaw and Tomayko. Later, Fincher et al. (2001) identified eleven project forms in computer science in their book dedicated to project course issues. Clear et al. (2001) provide a comprehensive list of issues that need to be addressed when devising a project course. Bothe (2001) provides attributes to assess the degree of realism in the project arrangements, and Burge and Gannod (2009) present dimensions for categorizing capstone projects — also including realism-related attributes. The models and dimensions given in these studies overlap. In the following, I therefore focus, first on one source, the project models given by Fincher et al., and then complement this by discussing the degree of realism with reference to the other studies.

The models of Fincher et al. are summarized in Table 1. They actually start with a smaller set of more abstract project forms which they call "project models"; however, for simplicity, I prefer to present all these eleven forms as "project models". Fincher et al. describe these models through so called composite case studies, i.e., they derived typical project course arrangements from their analysis and from actual practices at over 50 institutions, and identified eleven models. As the authors note, the list is not meant to be comprehensive, and each of the project models involves a variety of ways in which the model can be implemented in practice. Here, they demonstrate the variety of possibilities for devising a project course in CSE/SEE. The models are reviewed and compared to `TIES405`. Note that in the following the academic term is equal to 12 weeks.

TABLE 1 Project course models according to Fincher et al. (2001)

Model	**Individual/ group?**	**Year**	**Assessment**	**Pedagogic focus**	**Specialities**	**Key problem**
final-year individual	individual	final (3 or 4)	usually focused on deliverables	substantial, independent software development	student-supervisor relationship	supervising and assessing many different topics
second-year group	group (4–5 to 7–8)	2	usually individual, based on deliverables	introduction to group working and a "complete" system	group work	assessing individual performance within groups
taught M.Sc.	individual	post-graduate	based on the dissertation	demonstration of subject mastery, academic skills, and the potential for individual contribution	original work is expected	getting the students to come to terms with what's required
project with handover	usually groups of 3–6	2 or 4	individual and peer	learning about working with other people's code	reliance of students on others' work and, in competitive version, the economic environment	finding and dividing a suitable topic
research-type	usually individual	2,3, or 4	based on deliverables	developing critical thinking and research skills	possible involvement in research program	topic scope: identifying interesting topics that are small, well defined, and "off the critical path"
design and build	can be individual, more often groups of 3–5, or 4–7	any	product oriented	analysis, design, and implementation of a software system	"practicing the craft"	getting the right level of difficulty of the topic
project with industrial involvement	can be individual, more often groups of 3–6	2,3, or 4	industry partner and supervisor, based on deliverables	relevant exposure to industry practice and concerns	direct input from industry	industry properties must be balanced with academic objectives
project with client	groups of 3–5 (less often individual)	2,3, or 4	group and individual, based on deliverables	developing the professional skills to work as a contractor for a real client within realistic constraints	external clients who are not computing specialists	screening the clients
professional bodies' view	individual	3	usually product based, perhaps with attention to professional process and issues	demonstrated of sufficient professional technical skill	accreditation is based on practical, problem-solving projects	identifying projects of sufficient depth and breadth to meet the accreditation criteria

(Continues)

TABLE 1 (Continues)

process-based	group (typically 3–6, up to 20 possible	2 or 3	cumulative, involving interim, non-technical products	the process of project work and professional activity	emphasis on process	getting the students to see the point rather than just of through the motions
integrative/capstone	individual	final	summative, final-product-oriented	to integrate and demonstrate the skills and knowledge acquired previously	not just using information learned in disparate courses/modules, but also pulling it together into a coherent body	needs a problem that allows demonstration of skills; and needs the students to have those skills to demonstrate

A *final-year individual* project is typically a part-time effort extending over 2–3 terms, or alternatively a full-time one-term effort. Hours per week range from 6–35. The projects are research based, culminating and integrating, or apprenticeship-like build projects. They serve as a demonstration of competence, being evidence for potential employers. The course model is often substantial, e.g., 10–33 % of the degree, and failing may mean failing the degree. The students typically follow some acknowledged development methodology (waterfall, PSP), and the main deliverables are a final report and software product. Other deliverables include presentations, software demonstration, plans and specifications, progress reports, and surveys of the literature. Supervision of such individual project requires an abundance of resources.

The viewpoint that a course serves as an evidence of competence is shared with `TIES405`, which has been the course in the JYU/MIT curriculum that testifies to an authentic software project experience for subsequent employers. The students are motivated to work and demonstrate their skills and commitment to an external partner — the potential recruiter. This, however, is not a strong similarity because `TIES405` is more about completing an authentic customer project, and, due to the course's position as the only project course in the curriculum, i.e., the students have not previously taken project courses, it is more about accumulating experience and developing and reinforcing the variety of skills needed in a software project (particularly so outside the present research, see Section 2.2).

A *second-year group* project varies from 1–3 hours per week for 12–15 weeks, to 12 hours per week for 24 weeks. The justification for the second year is that first-year students lack programming and software engineering experience, while the third year is reserved for more technically challenging individual projects. The projects provide early exposure to team work and communication, i.e., the courses serve as exposure to large-scale team work. They approximate the industry practice, involving issues such as planning, scheduling, time management, progress monitoring, client negotiations, and documentation. Supervision typically takes place on a weekly meeting basis, and sometimes supervisors act as customers. Real industry partners may enhance the experience. Deliverables include both individual (e.g. report, diary) and group deliverables. The latter consist of software project products (software, plans, specifications, test documentation etc.). The projects are usually conducted in the scheduled laboratory or project classes.

`TIES405` shares most of these characteristics. The difference is that the course is not conducted in a scheduled pre-packaged manner, but in a realistic situational manner. It is taken later than the second year of studies and involves technically challenging topics and real customer expectations. It is more intensive, requiring autonomous work from the students. As a master's level course, it seems to be a tougher version of the second-year group project.

A *taught M.Sc.* project typically takes half of the study time over two terms (here 24 weeks), or full study time over one term. It is a culminating project demonstrating a student's mastery of a subject and its associated skills. An example Fincher et al. give is that project students are expected to focus on a specialism within the domain, perhaps an emergent area. The students are particularly expected to be capable of independent work. The students analyze an issue, find a relevant topic, and then carry out the work to produce the deliverables, which include interim reports and a dissertation. The project may take different forms such as design and build, evaluation, theoretical analysis, and empirical study. Industry may be involved as a customer. Typically, a supervisor oversees the work but does not manage it, and an internal examiner assesses the deliverables without other contact with the student. The students manage the study process themselves, which creates a challenge for monitoring their work. As an individual project the course is labor intensive.

As `TIES405` is now officially a master's level course, usually a design and build type of course expecting autonomous work from the students, other similarities are difficult to identify. `TIES405` in its present form is clearly a software project course, not an individual effort on particular subject that is studied in a research-like manner.

A *project with handover* is usually applied to group working, designed in phases, and usually covers 8–12 weeks. A handover project is based on an attempt to make an academic project relevant to an industrial environment, e.g., to be able integrate one's code with others' code, and deal with ambiguous and incomplete specifications. Fincher et al. identify two forms of this model, competitive and non-competitive. The requirement in both of these is to produce a software component with specified interfaces, to be able to integrate a working system from the components. The phases usually include component design and implementation, integration, and modification of a system. The students encounter the consequences of combining code written by different people. They gain an experience of a project too large for one person. In the competitive version students additionally compete to have their component integrated in the system, which thus introduces economic and social issues into the model. The objective of the model is to provide a realistic experience — at least by imposing environmental constraints. Much of the supervision effort goes into devising an appropriate project, after which it can mainly be monitoring. The phases of the model (such as design, implement, and prepare for sale) set the overall process for the students, while the team structure and processes are left to the students to decide on. Deliverables include a software component, demonstrations, documentation, and the integrated software system.

A similarity to `TIES405` is the encountering of other developers' work, i.e., being able to write code as a part of a larger project. `TIES405` projects are sufficiently large so that the work has to be divided and the students' code depend on the other team members' code. Also, often some part of the product is built using an existing component, e.g. an open source component with an appropriate license from the customer's perspective. In these respects `TIES405` has some handover characteristics. With real customers and authentic customer collaboration, the `TIES405` students have to frequently prepare and demonstrate their in-progress work for the customers but they do not need to compete with the other teams, as each team has its own project topic and customer. Overall, the "handover" is not a fixed arrangement but is used in a realistic manner if it suits the project's purpose.

A *research-type* project, usually an individual project, is used at different stages in the curriculum. The later it is devised the more is expected as an outcome. The length of such a project varies from a half term to two terms. The model follows the traditional research process of natural sciences, introducing students to research methodologies, research standards, and a research discourse. The objective is to develop skills of independent investigation by means of a supervised project. More precisely, the students learn how to design research, how to evaluate relevant related research, how to report research results, etc. The main deliverable is the final report.

`TIES405` is most of all a software project but occasionally some project topics have had a research-project character such that the student team studies different options to meet the customer's needs. The prototype produced suggests the selected solution to the customer's problem. Working with new technologies tends to introduce this quality into a project. In fact, it is not rare that some comparison between a variety of options (technologies, platforms etc.) is made at the beginning of a project, but research is in no way the principal focus of the course.

A *design and build* project is feasible at several levels and can be sized from a small two-week assignment to a large scale project covering two terms. A common characteristics at all levels and with all sizes is the objective of designing and building software that runs. The model simulates a traditional software development team producing substantial software. Depending on the size and emphases, different phases of the software life cycle may be included. The objective is to practice the craft and apply acquired skills and techniques. These include analysis, design, implementation, using a structured method, planning and executing testing, estimating, planning, and managing time. Projects may be chosen by students from a list of assigned topics, or real customers may be involved. Supervision depends on the size but often takes place on a weekly basis. Students follow a prescribed development method or may choose among established methodologies. Deliverables relate to those phases of the software life-cycle that are included.

This seems to be a very generative characterization which many software-centered courses match. `TIES405` is very much a design and build project course.

The aim is to produce working software, and, as described earlier, the emphasis regarding what phases of the software life cycle are included, varies according to the project topic and work process selected. While a `TIES405` project typically starts from identification of requirements and ends after implementation and rough system testing, projects involving re-developing a customer's existing system provide students with some maintenance experience. Use of short iterations (adopted in the context of the present research) also allows rework during the project, introducing some maintenance perspective.

A *project with industrial involvement* takes usually one or two terms (up to 24 weeks) and is substantial (an important part of the degree). Students are rendered more employable by responding to the industry recommendation that students must be exposed to software practices as they stand in industry. How far industry is involved will vary widely. Industry partners may only provide project topics or provide direct and active support to students. Industry's interest is based on a long-term value such as raising the company profile, creating links to research centers, and recruiting. Industry partners act more as employers than customers. The challenge of the model is to find suitable partners, particularly if they are to contribute to teaching. The project topics originate from the industry partners. The students are largely responsible for their own work and their team management. With active industry partners the role of supervision is mentoring and overseeing the student-customer relationship. The typical structure consists of an initial phase, a working phase, and a final phase. The initial phase includes the presentation of the problem by the industry partner and meetings to agree on the development methodology and the standards of practice. The working phase includes issues such as scheduled meetings, interim reports with feedback from the partner providing an opportunity to clarify requirements, diagnosis of problems, and email communication. The final phase includes presentation and report as well as a response from the industry partner. The deliverables vary depending on the problem. The outcome may simply be a report or it may encompass a prototype and software life cycle-related documentation. Usually a final presentation is included.

`TIES405` is close to this model, except in the role of the partner. In `TIES405` the industry partner is an authentic customer, who is, also according to the contract, required to attend actively and give students problem area-related information and guidance. There are not few phases only, but customer collaboration usually occurs in one-week cycles. As explained in Section 2.6, in `TIES405`, the deliverables relate to software development phases, and software is included as opposed to only preparing a final report. Final presentation is also included.

A *project with a client* is typically a small group size project designed to be of feasible scale and scope. It usually takes one term and 3–12 weekly hours. The project exposes students to work with real people who are not computing specialists. The students act as software and computing specialists, and are subjected to the complex constraints of a real environment, including time constraints. The emphasis is on communication skills, as the students need to elicit and comprehend customers' needs. The principal objective of the model is thus to give the students a taste of the real world. The level of realism is sometimes increased by setting up an explicit company structure, and a competitive element may be included, as when several teams work for the same client. Sufficient technical skills are required of the students to provide a genuine service for the clients. The students must be provided with a relevant technical environment (tools). Agreement on who owns intellectual property and copyrights has to be made in advance. Also, agreements on maintenance have to be made. The options identified are either that it is the client's concern or the department establishes a maintenance scheme.

Project issues are obtained by screening potential clients, and then negotiating with them. The project may take different forms such as IT audits or feasibility studies, developing strategies, and product development. Students are selected to work with particular clients by matching the skills and interests, based on students' preference voting, or the clients select the students based on student submissions. Supervision includes guidance of students but also of the customer in order to insure a productive interaction. It is about balancing between the client's requirements and academic objectives. It is necessary to arrive at a stationary target by pre-negotiating the issues before the project commences. Supervisors may act as line-managers and suggest processes for the project. Usually is it about hands-off supervision, allowing the students to take responsibility.

The client project assumes realistic roles with certain responsibilities within the student team. The students usually manage themselves, with some supervisor interaction. Clients present their requirements to the students which are negotiated with the team leader who has the principal contact with the customer. The students present the completed project to the client, with documentation, and prepare a project report for academic purposes. Deliverables include the outcomes of a software project (requirements specification, prototype, software product, software documentation etc.). Some of the deliverables may be targeted particularly to the academic side, e.g., reflections on experience and documentation on team working. The real client project thus sets two directions for the students' work: the client and the academic department.

This project model is a close match with `TIES405`. There are some differences in emphasis. For example, `TIES405` is more intensive as the students spend 15–35 weekly hours on their project during one semester (four months, around 16 to 17 weeks)[6]. Moreover, based on the experiences with `TIES405`, a "stationary target" is not possible in projects that develop new functionalities on the basis of real expectations. Customers seldom have a really detailed view of what they want; instead it must be clarified during the project, and, changes will occur. The stationary context is not even a relevant expectation, agreed by James (2005). Real project topics are also difficult if not impossible to pre-design to be of a feasible scope. The scope is refined after the project commencement through negotiation. Notice also that in `TIES405` all the students in the group meet their customer regularly, implying that those who have the best first-hand knowledge

[6] Note the problem of delayed projects, which was one of the main motivators of this dissertation.

on a particular subject can directly interact with the customer.

A project following the *professional bodies' view* is devised as an individual project. It can take place as a full-time task during one term or as part-time work over two or more terms. It requires around 100–150 hours work from each student, depending on the level of accreditation. The model is thus based on accreditation promoting proficiency and professionalism. It is based on the criteria for producing marketable engineers. Regarding such criteria, Fincher et al. refer to the British Computer Society's (BCS) and IEE's view of computer science education in the UK which are shaped by the Engineering Council's *Standards and Routes to Registrations* SARTOR 1997. BCS states that an accredited course equips students with practical skills essential to begin a professional career. IEE's accredited course aims to insure that a chartered engineer may demonstrate an acceptable level of competence. Professionalism is emphasized also so that the supervisors are expected to be qualified by being members of an appropriate professional body. The supervisor nurtures the growth of the student-as-practitioner. Supervision is typically light-handed, taking place weekly, and may include some monitoring and availability of a technical support.

According to the BCS's criteria, the objective is to provide practical work involving the production of an item of software. It involves structured engineering work with the full software life-cycle, and has an integrative role. It must involve the production of a competent report on the process and the product, including a critical evaluation. Ideally, it must put the problem in context, survey the relevant literature, and involve some novelty.

`TIES405` does not seek any fixed and formalized competence for the students. Thus, the emphasis is not on enhancing the views of particular professional bodies. Regarding the project outcome, some similarities can be found. `TIES405` expects the students to produce a careful report on both the project and the product including, among other things, reflective evaluation on the decisions made during the project, comparison of final outcome to that of planned, and reporting of personal experiences. As described in Section 2.6, basically everything is documented (archived) in the projects.

A *process based* project typically lasts from two weeks to one term. A small team size is typical, but according to Fincher et al., large teams containing up to 20 students have been reported. Project scope and duration depend on the place in the curriculum. Early projects are typically short while the later ones can extend over two terms. The project model emphasizes the process rather than exclusively the product. In the early curriculum, it is a means to bring home relevant ideas early and prepare the students for a later more advanced project. In the later stage the model can attempt greater realism. The model is basically an introduction to team work and industry practice, with due attention to non-technical aspects, and stress on the importance of reflection on practice. The project topics are usually assigned since the demands they make for the students' learning process. Supervision usually takes place in groups because of the emphasis on the team rather than the individual. Supervision is usually in the form of scheduled meetings with the supervisor and the team. The role of supervision depends on the place of the course in the curriculum. In the early courses, the supervisor may act as a client or a project leader, while in the later phase the role is more just an advisory one.

Due to the emphasis on process, the project usually has fixed milestones with particular outcomes assigned to them. The students are assigned to particular roles which may be rotated or fixed during the project. Reflection and critique are typically included in the process but they are not necessarily assessed. Handover elements may be included in order to promote communication. The deliverables relate to the process, and include requirements specification, design documents, interface prototypes, etc. Moreover, process deliverables such as minutes of the meetings, weekly records, work plans etc., are often included.

`TIES405` puts lots of emphasis on process and management. Adopting some form of process (cf. safety) is seen as necessary to work in a sufficiently efficient manner. The students are expected to prepare agendas, write minutes, report on weekly progress, write experiences into the project report, and return a self-evaluation. Thus, in addition to software and the software-related documentation, the process-related deliverables and reflection are an important part of the course.

An *integrative or "capstone"* project assumes from 6 to 35 student hours per week, depending on the intensity of the arrangement. The course is typically substantial, an intensive full-time effort during one term or a half-time effort extending over two terms. It is meant to be a culminating project demonstrating knowledge and skills acquired during a degree program. It aims to make learning real by integrating theory and practice through authentic problems, processes, and deliverables. Ideally such a course gives students a sense of what they have learned as an integrated whole. While it typically is an individual demonstration of what has been learned, group projects are also used as capstone projects. Capstone projects are typically final-year projects but can be used at any level of a curriculum to integrate preceding studies. Early capstone projects tend to be of smaller scope, focusing only on a specific part of the degree. Overall, the arrangements vary a lot, e.g., a capstone project may be a design and build project, a handover project, or a project with a client.

Project topics must be demanding and are typically assigned to ensure appropriate coverage and complexity. Supervision is limited and strategic. A supervisor may act as a consultant but rarely intervenes. Weekly individual meetings are common. Students themselves usually run the integrative projects and aim to show what they can do. They are expected to use and combine what they have previously learned. The students thus need to consolidate their prior learning into an interrelated whole. The project is a summative experience and the deliverables are product-oriented.

While `TIES405` drives both learning and reinforcing of new skills through an integrative learning experience, the emphasis is more in an authentic customer project experience than explicitly necessitating the coverage of what has been previously studied. It should be noted that because of the course's unique role in the curriculum, it has perhaps been the most integrative course for students.

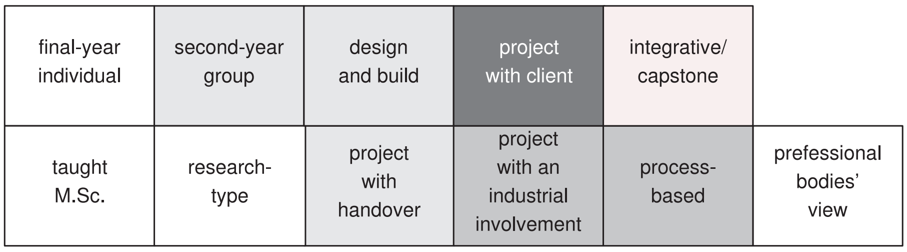

FIGURE 5 Matches between `TIES405` and the models of Fincher et al.: the darker the background color, the better the match

To summarize, the above comparison of `TIES405` with the models identified by Fincher et al. models shows how the boundaries between the project models are difficult to draw, thus, in practice a project course may share some of the characteristics of a variety of project models — as is the case with `TIES405`. There are clearly at least two explanations for this. Firstly, `TIES405`, as an authentic and intensive project course, simply covers many of the characteristics given above, and secondly, since the context of such a realistic course is not fixed, at least some of the projects conducted will match to a variety of characteristics. The closest matches were with the "project with a client", "project with industrial involvement" and "process-based" types, but the second-year model, handover model, and design and build model all shared important features with `TIES405`. Notice that the design and build model could be seen as a best match, but as an abstract and generative model it would not provide a very descriptive comparison for present purposes. Similarities to the integrative model where also found, but this had mostly to do with the `TIES405`'s position in the curriculum. Figure 5 summarizes the matches between `TIES405` and the Fincher's et al. project models. It is important to bear in mind the anecdotal nature and purpose of the comparison.

Although a strict project course taxonomy is difficult to construct, such literature is very valuable as it suggests research tracks and separates the field into dimensions that reflect real practice. The Fincher et al. models reveal the factors that are stressed in the particular course alternatives, and this kind of taxonomy proved capable of providing a best fit with `TIES405` — the project with a client.

Degree of realism

Because real customership is identified as the dominant character of `TIES405`, I will here review the degree of realism in project settings. This is an interesting topic as it appears that when project teachers speak of real projects they may actually mean very different things. The attributes for evaluating realism in the project course, listed by Bothe (2001) and Burge and Gannod (2009), form the basis of the following discussion. Table 2 depicts Bothe's attributes such that more realism is assumed in the right-hand column. The parts that could provide a match to `TIES405` are in boldface.

TABLE 2 Attributes of realism in a project setting according to Bothe (2001)

Factors	Attributes		
requirements source	academic	-	**real demand**
application domain	**system programs**	-	**other**
client (customer)	lecturer -	**non-comm.**	- **commercial**
goal project status	**prototype**	-	**product version**
project priority	**low**	-	**high**
deadlines	no	-	**yes**
project size	low -	**medium**	- **large scale**
phases (and documents)	only selected phases	-	all phases
original requirement's source	academic	-	**real demand**
project requirement's source	academic	-	**real demand**
task class	maintenance	-	reverse engineering

In `TIES405`, the *requirements source* is always a real demand whether the customer is an internal or external organization. This is ensured during customer screening and the initial negotiations with potential customers (see Section 2.4). The *application domain* of the project directly depends on the customer's branch of business. This means that groups working in non-technical domains (collaborating with non-technical customers) better encounter the collision between technical and non-technical worlds.

Customers are either non-commercial or commercial, but never the project teachers themselves. Generally, one can conclude from the literature that there is a shared preference in the SEE community towards real project topics and customers, but contrasting viewpoints have been stated as well (Aygün, 2004). It should be noted that internal `TIES405` projects have sometimes been very challenging because many different customer interests have been involved (representatives from several university units), making decision making very difficult. Teachers have also noticed that industrial customers tend to be better used to effective decision making compared to university customers. Therefore, a non-commercial customer does not necessarily constrain realism but rather introduces certain kinds of challenges (cf. Table 2).

The *project priorities* have varied from low to high, with respect to whether the project topics are on a critical path. Clear et al. (2001) suggest the rejection of project topics on a critical path. My experience with `TIES405` is that the appropriate contract model together with the teachers' responsibility/courage to intervene prevent problematic situations (this issue is included in the results in Section 5.2.2 under the title "teaching as coaching").

`TIES405` exposes students to work with *deadlines*. The course expects project planning, including the setting of milestones, and the one-semester timescale itself is a major deadline. The *project size* is small or medium as to team size and timescale, but the size of the problem is large, often leading to long future development and maintenance, providing internship and job opportunities for the students. Regarding the *task classes*, Bothe contrasts maintenance projects with reverse engineering projects. This does not characterize `TIES405` where one of the most usual cases is a from-scratch project. Moreover, the phases and task classes that are emphasized in `TIES405` vary according to the particular demands of the project.

Burge's and Gannod's realism-related dimensions for categorizing capstone projects include the following:

- Customer Identity — external customers, particularly those with limited technical expertise, add to the challenge and realism in a capstone project.
- User Identity — eliciting requirements from users with a different level of technical expertise or who have a different domain vocabulary is a very useful skill.
- Deliverable Audience — students need to develop deliverables focused on the client and on future developers in order to meet the course outcomes and learn how to support both audiences for their software.
- Other Deliverables — additional deliverables can be requested to ensure that course outcomes are met if the deliverables required by the customer are not sufficient.
- Project Type — most projects in industry are not developed from nothing; however, maintenance/enhancement projects may not always involve sufficient new requirements development to give them experience with requirements elicitation.
- Availability of Original Developer — often the original developer is unavailable to assist or has limited time available. Learning to cope with these difficulties can be a valuable, if painful, learning experience.
- Number of Teams — multiple teams working on the same project is not as realistic as having a single team fully responsible for creating a working system.

These dimensions overlap with those given by Bothe. Regarding the *customer identity*, *user identity*, *project type*, and *availability of original developer*, there is variation in `TIES405`'s arrangements. As to the project type, Burge and Gannod seem to prefer projects that build from scratch. In my experience this creates a communication learning opportunity that arises from the necessity to elicit requirements together with the customer.

Regarding the dimension of *other deliverables*, `TIES405` projects do not usually lack requirements but are very intensive; see Sections 2.6 and 5.3.1. Regarding the *number of teams*, in `TIES405` each team works independently with its own customer and project topic, aiming for a working system, and the anecdotal rationale is the same as that given by Burge and Gannod. The dimension of *deliverable audience* likely relates to what Clear et al. (2001) called a conflict between the customer's dictates and a soundly engineered product (two purposes/audiences). I have noticed that this issue can arise with non-technical customers who may assume that software functionalities are mature enough when first demonstrated. Two audiences in this sense are acknowledged. For example, code inspections are included and the project outcome includes a reflective software report in which future development issues are identified (see Section 2.6).

It can be concluded in the light of the attributes identified by Bothe and Burge and Gannod that `TIES405` is a realistic context with some variation due to the varying nature of real project topics. Intensity (see Section 2.6), contracting (see Section 2.4), and the information that customers pay for their projects can be assumed to make realism very apparent and concrete for students. Perhaps `TIES405` belongs to such a category of department-located project work models which represents the closest option to the models where students are paid and/or carry out their work (learn) at the facilities of employers; cf. cooperative learning models and student companies mentioned at the beginning of this section.

2.9 Summary

The dominant characteristic of `TIES405` is the collaboration with real customers. Both the end product (software, plans, specifications, design artifacts, etc.) and the process (roles, phases and tasks, scheduling, time logging etc.) by which the product is produced are paid attention to. Real customer expectations imply the first, and without the latter, i.e., the adoption of a sufficiently structured way of working, it becomes a difficult task to achieve the former. This is, of course, just an anecdotal rationale and summing up of the emphases in the course arrangements.

I have noted the variation in `TIES405` arrangements throughout the chapter. Here one could identify a problem, as the project students inevitably encounter different things and a different degree of challenge. On the other hand, assigning a real issue and a real customer to each project — sourced from several real-life branches — enables students to notice how software is embedded in society. There is also variation in the students' work within a single group as the students work in specific roles. Similar variation has been pointed out in the literature (Daniels and Asplund, 1999; Hagan et al., 1999), both studies reporting on collaboration with real customers.

In reviewing the project models of Fincher et al., I noticed that some of the `TIES405`'s features are due to the course's position in the curriculum as the students' first project study. In particular, important challenges arise from this setting. The course is often a student's first practical exposure to almost everything that is encountered, e.g., real problems, ill-defined requirements, real expectations, customer interaction, formal meetings with agendas and minutes, contracting, team work, process, project management, technical documentation, presentations, programming in a team, and uncertainty in general. This implies an "everything-is-new" starting point for the students. Because of this starting point and the focus on both the process and product, it is evident that the project course semester is very intensive for the students.

I have previously noted that the course could be supported with a second-year project course in order to lessen the several simultaneous challenges now encountered in the course (Isomöttönen and Kärkkäinen, 2008). Correspondingly, Clear et al. (2001) note that when the goals are set for a project course, the course's role in the curriculum should be considered to find out if adequate support is available from the prior curriculum. This is not however the research focus of this dissertation.

The chapter outlined `TIES405`'s arrangements and discussed these in relation to the course models identified in the field. The literature references foregrounded a number of issues to be addressed when devising a project course. It is evident that a dissertation on a software project course needs be restricted to a certain area to achieve some depth in the study. The options recognized in this dissertation were either to remain at the holistic level and study relationships at that level, or to drill down through a certain course mechanism. The first option was chosen, but also restricted to a concern with student perceptions and operational issues in the course.

3 RESEARCH METHODS

The aims and approach of the dissertation have so far been presented in the Introduction, and the research context described in Chapter 2. This chapter reviews the research methods used in the dissertation. The dissertation has two results chapters (Chapters 4 and 5) the research methods of which differ.

The research presented in Chapter 4 utilized Grounded Theory (GT), a research method originating from sociology. The term grounded theory refers not only to a research method, which here is a theory generation process, but also to the theory that results from such a process. The original publication on the method, *Discovery of Grounded Theory* (DGT), was written by the sociologists Glaser and Strauss in 1967 (Glaser and Strauss, 1967). Basically, DGT was a conceptualization of the method the originators had used when conducting and writing up their own research (Glaser, 2004). Most of all, DGT introduced a research method that yields theories that are systematically generated from the data, and thus grounded. While it emphasized theorizing from qualitative data, it also introduced the use of quantitative data. GT should thus be seen as a method for theorizing social reality in which different types of data on the same subject can be utilized, although it is usually applied in the qualitative domain.

The primary research method underlying the results in Chapter 5 can be described as action research, which was first popularized by Lewin (1946). The action research method is commonly seen as appropriate when addressing local problems. Ellis and Kiely (2000) characterize it as a cyclical process whereby knowledge is created in and for action. The cycles, consisting of planning, action, and analysis of action, can be repeated so that the previous cycle informs the next one. Action research types of inquiry have been referred to by many different names (e.g. emancipatory research) and many definitions have been proposed. In educational contexts, action researchers study their own educational practices with the aim of improving their understanding of these (Carr and Kemmis, 1986, p. 180).

In order to give an overview of how these research methods were used in this dissertation, the relationship between action research and grounded theory is depicted in Figure 6. The action research process is associated with my teaching

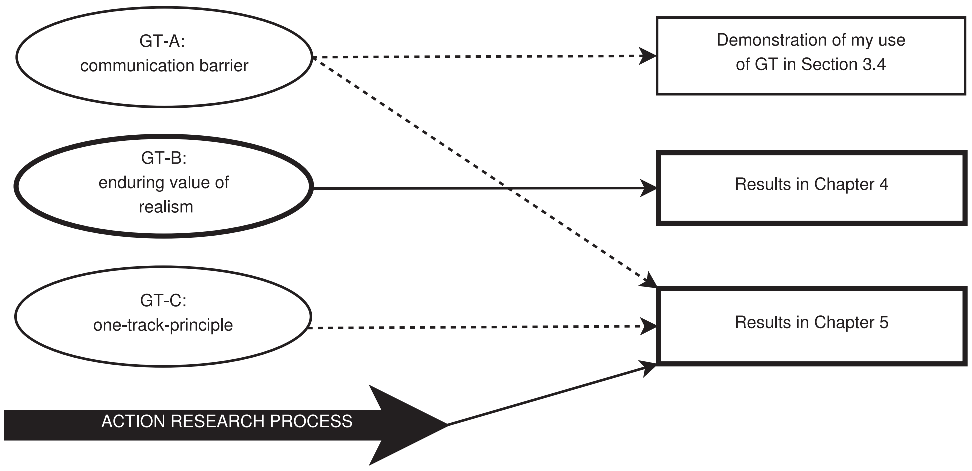

FIGURE 6 Relationship between action research and grounded theory

and course development work, and its results are largely based on teacher reflections on the projects supervised during the course development work. In contrast to this, the grounded theory processes (see Figure 6) were developed separately from teaching, the analyses focusing on student course experiences documented in project reports. These two research methods were thus employed in parallel.

However, towards the end of the dissertation process, these two kinds of research activities (action-related course development work and theorizing from the textual data) were observed to have a methodological connection. This arose from the theorizing aim of the dissertation. That is, I noticed that when developing my reflections on action research into a theory, a GT-like comparison in an intuitive way entered the action research process as well. Both of the research methods of the dissertation can thus be associated with theory development, bearing in mind that in the other case this is directly connected to the action-taking context.

The two methods are also connected by the results. As depicted in Figure 6, there were altogether three grounded theory themes (processes) that were developed in the course of the dissertation (the order of the grounded theory processes refers to the temporal order of their completion, GT-C, marked with a dotted line, was never completed into a stand-alone theory). Chapter 4 is dedicated to one of these themes (GT-B: enduring value of realism). The other two GT processes are referred to or presented in the action research part of the dissertation (Chapter 5) because, as subject matters, they fit the theorizing conducted in the action research context. Therefore, they can be integrated into the theory developed through action research. In addition, in Section 3.4 GT-A (communication barrier) is used to demonstrate how GT was applied in this dissertation.

The remainder of the chapter is organized as follows. The chapter starts by outlining the main ideas and methods introduced in DGT (Section 3.1). I purposely start by referring to this original work on grounded theory because it furnishes all the basic GT concepts. After this, in Section 3.2, the main differences between the two subsequent grounded theory approaches are foregrounded. The originators, Glaser and Strauss, took different directions after publishing DGT, and these directions became the two main GT approaches. After the GT basics and the nuances, a review on GT usage in computing education research is given in Section 3.3. A set of computing education studies is analyzed so that the GT principles presented in Sections 3.1 and 3.2 provide the reference base for the analysis. The review, suggesting that it is difficult to see genuine theory generation and finished grounded theories in the field, supplies the motivation for carefully explicating the GT approach followed in this dissertation. This is done in Section 3.4 which provides background information on why a particular GT approach was selected, and illustrates the use of grounded theory principles in detail.

An overview of action research is given in Section 3.5. The role of action research in this dissertation is described in Section 3.6. This explains how the theorizing aim affected the research and the presentation of its results, and outlines the main phases of the action research conducted. The reader should pay special attention to the nature of the present action research as described in Section 3.6.

Section 3.7 describes the data sources of the dissertation and explicates how these relate to the use of GT and action research, i.e., how the data and the results chapters of the dissertation interrelate. The chapter concludes with a summary of the use of the research methods in Section 3.8. The summary presents the timeline of the research and locates the research methods used and the data along this timeline, giving a more detailed view of the overall research process than that outlined in Figure 6.

3.1 Discovery of Grounded Theory

This section outlines the general idea, the analysis method, and other key aspects of theory generation introduced in DGT. It should be first noted that, in the context of grounded theory, the process of theory generation has an implicit component: the grounded theory literature often speaks of theoretical sensitivity of a researcher (Glaser, 1978). In this connection, the present section, and also the subsequent sections, emphasize the methodological ideas of grounded theory as opposed to laying out prescriptive data analysis techniques.

3.1.1 General idea of DGT

Perhaps the idea uppermost in DGT was that the adequacy of a theory should not be separated from the process by which it is generated (Glaser and Strauss, 1967, p. 5). This meant a strong linkage between a theory and its process of generation, and differed from what Glaser and Strauss described as logico-deductive theories. Such theories could be based on abstract ideas that needed to be tested and verified (Glaser and Strauss, 1967, p. 10):

> *Currently, students are trained to master great-man theories and to test them in small ways, but hardly to question the theory as a whole in terms of its position or manner of generation.*

In contrast to logico-deductive theories, DGT preferred emergent ones. A grounded theory consists of categories and properties of the categories, both of which are concepts[1]. DGT suggested that concepts and their integration are to be based on emergence, thus discovered from data (Glaser and Strauss, 1967, p. 37).

Emergent concepts were preferred over borrowed ones to solve the problems of richness, fit, relevance, and forcing: emergent concepts, discovered in the data, allow the generation of a rich theory that fits the empirical situation, whereas borrowed concepts require justificatory explanation, may turn out irrelevant, and create a risk of forcing. In fact, DGT suggested that theory generation should be started by ignoring the literature (borrowed concepts) until analytic categories are found. This reduces the chance that external conceptions about the research target would contaminate the theory being developed.

Another key aspect highlighted in DGT was the on-going nature of theory development. This was supported with metaphors "theory as a process" and "ever-developing" (Glaser and Strauss, 1967, p. 32) which emphasized *generation* as opposed to research aiming for fixed conceptualizations. It should be noted that theory as a process does not mean that a grounded theory is never so complete that it can be reported on. Because it is "ever-developing", GT puts forward conceptual hypotheses. Those conceptual hypotheses that the analyst is confident with, as a result of careful analysis, are finally written out as a GT.

Overall, DGT popularized a method for theory development that is essentially a process, thus has an on-going nature, and is based on emergence. This was called grounded theory, which refers both to the method and the resultant theory. As the benefits of grounded theories, DGT points out that such theories can be expected to suit their supposed uses (Glaser and Strauss, 1967, pp. 1,3) and become readily understandable to both sociologists and laymen (Glaser and Strauss, 1967, p. 1). Theories based on emergence were argued less likely to be completely refuted by more data or replaced by another theory, whereas logico-deductive theories were argued more likely to lead a follower astray (Glaser and Strauss, 1967, p. 4).

It is important to note that DGT, and subsequent writings by Glaser and Strauss, usually speak of grounded theory as a method, without addressing foundational issues. In this sense, the value of grounded theory is associated with the methodological manifesto described above (the fit and the usefulness of an emergent theory). It is easy to observe in the literature that DGT, and subsequent work by Strauss and Glaser, are being interpreted in order to address the foundations of the method. In this connection, DGT has been subjected to a great deal of

[1] Concepts as categories and properties in the context of DGT points to the fact that certain concepts (properties of categories) further characterize certain other concepts (categories). Therefore, categories can be seen as stands-alone concepts compared to properties. See (Glaser and Strauss, 1967, p. 36).

criticism, which, for example, concerns whether grounded theory constitutes a scientific method; see discussion by Haig (1995).

3.1.2 Constant comparative method

DGT introduced a constant comparative method as a means to generate grounded theories. Roughly speaking, this was about generating theory from similarities and differences in data, thus by comparison. In this process, a theme that emerges as relevant is taken as the main theme of the theory being developed, and then the rest of the (comparison) process can focus on refining this theme into a coherent theory. In the light of the observation that grounded theory is essentially a comparative theory development process, it is noted here that grounded theory has been successfully adopted within many disciplines, including education, see (Haig, 1995).

DGT identified four overlapping stages (or tasks) in its comparative theory generation process (Glaser and Strauss, 1967, p. 101-116). These are

- comparing incidents[2] applicable to each category
- integrating categories and their properties
- delimiting the theory
- writing theory

"Overlapping" above means, in short, that the analyst should jointly code and analyze. Thus, the analyst collects data and codes systematically, but also analyzes the data at the same time in order to allow the analysis to affect the data collection and coding. With the systematic coding, GT aims to be rigorous, as those methods that try to quantify qualitative data and then test hypotheses, while it also allows theory generation in which separating data collection, coding, and analysis is not reasonable. The constant comparative method does not guarantee that two analysts will achieve the same results, and in this sense, it is about theory generation which is affected by the skills and sensitivities of the analyst (Glaser and Strauss, 1967, p. 103). The stages of the method likely start in a given order, but they can continue in parallel until the theory generation is terminated. The four stages are explained as follows.

Stage 1. The basic defining rule of comparison is that, in coding an incident, the analyst compares the incident with previous incidents coded in the same cat-

[2] An *incident* refers to a data segment meaningful for the development of a theory (my interpretation of GT literature). What constitutes an incident depends on the context of comparison. A certain topic can be discussed meaningfully at several points in the context of one human response (e.g. an interview), in which case comparison should take place within this one response (e.g. an interview), as opposed to only between different human responses (e.g. between different interviews). On the other hand, I want to refer here to Chapter 5, where I illustrate comparison between broad reflections on the operational behavior of project groups (between the use of an iterative, incremental, or waterfall process). In terms of grounded theory, I consider these group behaviors as data incidents.

egory. This happens within the same and different comparison groups. The second rule of comparison says that the analyst should stop coding and record a memo on his/her ideas. The latter underlines the necessity to capture the fresh theoretical ideas that the comparison evokes.

Stage 2. Integration means the discovery of how concepts and their subconcepts interrelate, and hence, result in a unified whole. As coding proceeds, the analyst compares, not only incidents with some other incidents, but with the properties of the category that resulted from the initial comparison of incidents (Glaser and Strauss, 1967, p. 108). This leads to integration. The constant process of comparison is said to lead naturally to the discovery of integration.

Stage 3. According to DGT, delimiting the theory is forced through the constant comparison. Theory becomes solidified, modifications become fewer and the emphasis turns to clarification and reduction. Delimiting takes place with regard to the theory and categories. The theory can be presented with fewer categories as the analysis may yield concepts that seem overlapping and can be generalized into a higher level concept. The analyst can be more focused restricting the work to certain parts of the findings. The categories identified seem to have obtained all of their meaningful properties and new findings only add unnecessary detail to the theory (see *theoretical saturation* in Section 3.1.3). DGT states that the analyst starts to achieve two major requirements of a theory: parsimony of variables and formulations, and scope in the applicability to wide range of situations while the theory remains close to data (Glaser and Strauss, 1967, p. 111). Grounded theory literature has later adopted term *core category/variable* in order to explain how the theory becomes delimited around one theme; see Section 3.2.4.

Stage 4. Writing a theory means the process of transforming the coded data, notes, and a theory into a systematically written form. As the stages overlap, one may continue to do further analysis and return to the data to include illustrations in the text when writing a theory. DGT says that the discussion in the research notes provide the content behind the categories which become the major themes in the theory (Glaser and Strauss, 1967, p. 113). An example of writing GT is the way in which Glaser and Strauss (1964) arranged the main categories as section titles in their grounded theory on social loss.

3.1.3 Theoretical sampling and saturation

In a GT process, a potential theory starts to emerge as soon as the coding and analysis take place. Theoretical sampling means that the (initial) theoretical framework being generated guides the process of data collection, coding, and analysis. In other words, the decision about what data should be utilized is controlled by the emerging theory. As the analyst finds an initial theory, it needs to be furthered. All the relevant properties of the categories need to be found, and the relations between the concepts clarified. The theoretical saturation of a category means that the category seems to have obtained all the relevant properties. At that point of saturation, the analyst starts to see similar indicators in the data over and over again.

Theoretical sampling and saturation are closely related to the integration and delimiting of a theory. Theoretical sampling together with joint data collection and analysis is said to facilitate the integration of a theory: *...the integration of the theory is more likely to emerge by itself* (Glaser and Strauss, 1967, p. 109). Correspondingly, as categories become theoretically saturated, the theory is undergoing delimitation (Glaser and Strauss, 1967, p. 111).

3.1.4 Sensitizing and analytic concepts

DGT (1967, p. 38) emphasizes two joint features of concepts. Concepts should be both sensitizing and analytic. "Sensitizing" means that the concept enables the formation of a meaningful picture of the entity it describes. It enables sensing of the empirical level of the phenomenon. An analytic concept is more profound than just a word underlined in the data. The level of abstraction of an analytic concept is raised compared to raw data. It characterizes the entity rather than being the entity itself. It is potentially meaningful in the given social context, even if the findings later indicate new angles or even a change in the research target.

These characterizations of a concept provide a concrete guideline to create an analytic substantive grounded theory which may be furthered to create a formal grounded theory (see the next section).

3.1.5 Substantive vs. formal theory

DGT differentiates between two types of grounded theories (Glaser and Strauss, 1967, p. 32-33). A substantive theory is grounded in the research on a particular substantive (empirical) area whereas a formal theory addresses more abstract, conceptual area of research. Both are middle-range theories, grounded to data, and fall between the minor working hypotheses of every day life and all-inclusive grand theories. When generating substantive theory, comparison is made within one substantive area. An example from DGT is that when studying a status passage, such as in dying, the substantive area is dying, and the comparison is made in the context of dying. When generating a formal theory, one can focus on the status passage in general. Then the comparison is made between different substantive areas in order to create understanding of the status passage without a particular connection to a specific substantive case.

A substantive theory is suggested to provide a strategic link in generating a formal theory (Glaser and Strauss, 1967, p. 79). Thus, an analytic conceptualization that is derived by starting with a substantive objective, may provide a conceptual hypothesis which becomes a research interest when a formal theory is generated. One can here consider the first results chapter of the dissertation. The core category (see Section 3.2.4) puts forward a mechanism that explains students' course experience from a one-semester real customer project course model. This is first advanced with a comparison to the literature on several different course contexts and curriculum levels (see Section 4.2), and then discussed as a potentially interesting hypothesis (worth studying) with respect to any curriculum that includes realistic learning contexts (see Section 4.3).

3.2 Glaser vs. Strauss

This section sheds light on the differences between the two major grounded theory approaches that emerged after DGT. Glaser published *Theoretical sensitivity* (Glaser, 1978), which is a continuation and elaboration of the methdological manifesto started in DGT. Strauss took a different stand and published a more proceduralized view of GT with Juliet Corbin in 1990 (Strauss and Corbin, 1990), titled *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. These represent the two major approaches to GT, sometimes called *Glaserian* and *Straussian* approaches.

Basically, Glaserian approach is the method introduced in DGT (thus see Section 3.1), but the formulation of the method became more explicit and thorough. For example, the delimiting of theory around one theme is explained with the term core variable, which refers to a matter problematic (relevant) for the people the theory concerns. Relating categories with each other is supported by theoretical codes which represent the manners of how concepts might relate with each other within a theory. Memoing and the emphasis on the researcher's thought process are highlighted, as was the case in DGT. Straussian grounded theory is a more mechanistic one, as interpreted in the literature (Adolph et al., 2008). It is laid out as a sequence of coding procedures that should be strictly followed. These procedures are called open, axial, and selective coding, and associate with the tasks of extracting concepts and grouping them into categories, relating categories with each other, and finally selecting one main theme among the findings to present a focused theory.

It is important to be aware of these two approaches and their differences to avoid confusion when applying the method. Researchers have noticed this problem in their first experiments with GT (Adolph et al., 2008). Glaser published a response to Strauss' and Corbin's work in 1992 (Glaser, 1992). This response, titled *Emergence vs forcing*, explains the differences between the two GT approaches, chapter by chapter. In the following, I discuss the differences I considered most important and illustrative.

3.2.1 Induction vs. deduction

Strauss and Corbin (1990, p. 111) propose that the analysis is about continuous interplay between inductive and deductive thinking. Hypotheses are deduced and then verified. Glaser (1992, p. 71) argues that induction is not about proving or disproving a hypothesis with data but about figuring out from the patterns in the data what concepts and hypotheses emerge.

3.2.2 Thought process vs. mechanical process

Strauss and Corbin introduced a set of rules and models for generating concepts and integrating them. *Coding is a systematic and precise set of procedures that can't be done haphazardly or at the whim of the researcher. In order for the emerging theory to be grounded, as well as valid and reliable, the procedures must be followed just as carefully as those that govern good quantitative studies* (Strauss and Corbin, 1990, p. 46). This appears to be an attempt to explicate theory generation as a mechanical process compared to Glaser's emphases. Glaser (1992, p. 76) says that the mechanistic approach based on a set of rules leads to creating, which then involves testing/verification.

Glaser calls for theoretical sensitivity on the part of the analyst. As stated in Section 3.1.2, in DGT, the analysis was advised to be terminated in order to catch the fresh ideas discovered in the data. Glaser (1978, p. 7) continues this in his Theoretical Sensitivity by emphasizing the *thinking* required in the analysis.

I find myself agreeing with Glaser, and consider the viewpoint of discovering vs. creating very illustrative. Serious thought efforts are needed to understand what is being discovered, i.e., what the emergent theory is. In contrast, creation is associated with deduction, produces a model of reality with mechanistic rules, which then requires verification to see whether the model — created not discovered — matches reality.

3.2.3 Paradigm model vs. emergence of theoretical codes

An illustrative example of the differences between the approaches is how the authors suggest the relation between the concepts is to be discovered. Strauss and Corbin present a single model called the *paradigm model* for relating subcategories to their categories, whereas Glaser gives eighteen families of theoretical codes that can help in putting a theory together, and notes that codes continually emerge and enter sociology from other fields. Glaser also says that other theoretical codes can be developed in order to avoid being trapped into "pet" codes, see (Glaser, 1978, p. 82). In (Glaser, 1992, pp. 62-63) he criticizes the Strauss and Corbin single model approach by arguing that GT cannot be developed in terms of a single predefined model as this would force the data to fit the model and restrict the analysis to thinking of data only in terms of a single model.

According to Strauss and Corbin (1990, p. 99) the paradigm model is a sequence of causal condition, phenomenon, context, intervening conditions, action/interaction strategies, and consequences. An example the authors use is a broken leg (causal condition) leading to pain (phenomenon) with certain properties (context), which again leads to certain actions, e.g., go for emergency help, and finally to a pain relief (consequence). An intervening condition, for example, might be a long distance to go for emergency help, i.e., a general condition related to action strategies. The authors say that in axial coding — the process of relating sub-categories to categories — each category is developed in terms of the paradigm model (Strauss and Corbin, 1990, p. 114). A code similar to the paradigm model is included in Glaser's coding families, see (Glaser, 1978, p. 74).

3.2.4 Selective coding and core category

Selective coding refers to how a grounded theory can become focused on a certain theme. Strauss and Corbin (1990, p. 116) say that selective coding is the process of selecting a core category, relating it to other categories, validating those relationships, and filling in the categories that need further refinement and development. Relating a core to other categories is suggested to be done in terms of the paradigm model; see Section 3.2.3.

Glaser (1992, p. 75) speaks of core variables which relate to basic social processes in the phenomenon under study. According to Glaser, selective coding is about arrival at and selection of such a core category in the analysis. Only after discovering such a core variable with confidence, with relevance to those involved, can the analyst focus on elaborating the selected area of the study, and proceed with theoretical sampling to provide an analytic theory around the core variable. The core category thus guides the analysis. The core category and how it is integrated to other categories and their properties emerge. Strauss and Corbin (1990, p. 123) argue that the core need not be a basic psychosocial process because GT is an action-oriented model.

The difference between the approaches is that in the Strauss and Corbin approach selection of the core is done late in the process (open coding $\rightarrow$ axial coding $\rightarrow$ selective coding), but in the Glaser approach the core should first emerge, leading to a subsequent more focused analysis. Both approaches highlight the importance of a theory in the context of a single core category.

3.2.5 Coding levels

Strauss and Corbin (1990, p. 73) suggest that coding starts word-by-word and line-by-line. They give conceptual labels to single observations. Glaser (1992, p. 38-45) commented that the analysis should focus on a small set of relevant categories and their properties in order to avoid the "helter skelter of too many categories" that he says happens in the Strauss and Corbin micro-level approach. Glaser's criticism foregrounds that finding a relevant problem (cf. selective coding) becomes difficult with micro-level conceptual labeling. He sees such an approach as burdensome and tedious. Allan (2003) calls Glaser's type of coding key point coding, which is the term I use if to refer particularly to coding that takes account of relevant points in data as opposed to coding starting from the conceptual labeling of micro-level units.

3.3 A systematic review of GT in computing education research

It can be observed that GT papers in the IT field have focused on both discussing the method (Hughes and Jones, 2003; Adolph et al., 2008; Hansen and Kautz, 2005) and the actual results of using it (Coplien and Devos, 2000; Coleman and O'Connor, 2008). This division of interests has been noticed in other fields as well (Sudol, 2008). In this section, a systematic literature review, as discussed by Kitchenham (2004), focuses on computing education research papers that used (or referred to) GT as their method. The review illustrates GT adoption in the field and enables the dissertation to be situated in just this context. This illustration is also seen important because grounded theory has been noticed to be a difficult method, and because different definitions of grounded theory may have entailed quite different interpretations of it. According to Glaser's criticism, one of the biggest challenges is that grounded theory has been, as opposed to theorizing, reduced to a qualitative data analysis technique (Glaser, 2004).

3.3.1 Review criterion

The criterion for including an article in the review was the following: it deals with computer science education, or software engineering education, or CSE/SEE-related capstone projects, and involves students. Based on an initial screening, I concluded that focusing on a more specific criterion, for example, reviewing grounded theory studies on one-semester student projects in software engineering, would not have yielded sufficient material for the review.

3.3.2 Sources

The data sources used were the IEEE and ACM digital libraries, Lecture Notes in Computer Science, *Journal of Computer Science Educations (JCSE)*, and *Journal of Systems and Software (JSS)* Using the first two enabled access to *Journal of Educational Resources in Computing (JERIC)* and *IEEE Transaction on Education*, and proceedings of several computing education conferences, such as CSEE&T, SIGCSE, ITiCSE, and ICER.

Searches

I first experimented with the databases (how to arrange the search words) to be sure that the final form of the searches would allow a relevant outcome. With IEEE Explore (April 24, 2009), the following words were used while searching: *"software engineering education" and "grounded theory"*, *"computer science education" and "grounded theory"*, and *"capstone project" and "grounded theory"*. With the ACM library (April 27, 2009), the following were used: *"software engineering education" "grounded theory"*, *"computer science education" "grounded theory"*, and *"capstone project" "grounded theory"*. With LNCS (April 28, 2009), the following were used: *"software engineering education" "grounded theory"*, *"computer science educa-*

tion" "grounded theory", and *"capstone project" "grounded theory"*. With JSS, (April 28, 2009), the following was used: *"grounded theory"*. With JCSE (July, 2009), I opened by hand every paper of the journal available at JYU (from 1998 to 2008 issues 1 and 2), and searched each opened document with the words *"grounded theory"*[3]. In addition, I made a *"grounded theory"* search using the journal's search engine to cover the latest issues. All searches were documented in a text file.

3.3.3 Selection of papers

When selecting an article to the review, each document received in the search was opened. I first read the document's title and abstract, searched the document with words "grounded theory", skimmed through the method or data analysis section, and skimmed the references. Any reference to the use of GT or its procedures was accepted. I was, finally, not strict with the criterion in the CSE/SEE context. For example, the study context of Lappenbusch and Turns (2005) is students' technical documentation, and that of Kautz and Pries-Heje (1999) is students' systems development methodology adoption — without being particularly CSE/SEE-specific. Such studies were seen as close enough contexts, and were included to supply sufficient data for the review. Another exception concerns the continuation studies. Deibel's study (2008) is a continuation study of her former study (2007). Almstrum's studies are poster papers, and here too, the latter one adds to the former one (Almstrum and Last, 2005, 2006). I included these continuations papers because they involved further analysis, by making new comparisons or adding more study subjects to the analysis, and, because they immediately illustrated how GT has been applied little by little and partial results published based on small steps in a research process. As I was only interested in exploring the use of the method (how GT had been applied), I did not consider the quality of the papers, as suggested by Kitchenham (2004). Selected papers were added to the reference management system *Bibsonomy*[4].

The reviewed studies were found in the IEEE and ACM libraries, and JCSE. GT articles were also found in other sources, for example, in the Journal of Systems and Software, but these did not meet the criterion of dealing with education and students. Altogether 30 matches were found. They are given with the source, the search words used, and research context in Table 3.

TABLE 3 Reviewed papers with their research contexts

Authors & Source	Research context
Isomöttönen and Kärkkäinen (2008), IEEE CSEE&T, search: SEE & grounded theory	Capstone project in software engineering (one-semester real customer model).
Williams et al. (2007), IEEE ICSE, search: SEE & grounded theory	Social interaction and pair programming.
Ho et al. (2004), IEEE Frontiers in Education, search: SEE & grounded theory	Effect of pair programming on female students.

(Continues)

[3] This "full" search was done due to a parallel interest in several research topics.
[4] `http://www.bibsonomy.org`

TABLE 3 (Continues)

Last (2003), IEEE Frontiers in Education, search: SEE & grounded theory	Team development in an international student collaboration project.
Shull et al. (2000), IEEE Transaction on SE, search: CSE & grounded theory	Reading techniques in OO framework learning.
Lappenbusch and Turns (2005), IEEE International Professional Communication Conference, search: capstone project & grounded theory	Technical communication.
Begel and Simon (2008), ACM ICER, search: SEE & grounded theory	From a novice software developer to an expert, the tasks encountered in moving to industry from university studies.
Almstrum and Last (2006), ACM ITiCSE, search: CSE & grounded theory	How computing attracts males/females?
Almstrum and Last (2005), ACM ITiCSE, search: CSE & grounded theory	What attracts women to CS?
Chen et al. (2007), ACM SIGCSE, search: CSE & grounded theory	Students' sorting abilities (when they undertake CS1 course).
Cheng and Beaumont (2004), ACM ITiCSE, search: CSE & grounded theory	Communication tools used in distributed PBL teams.
Chinn et al. (2007), ACM ITiCSE, search: CSE & grounded theory	Problem solving in data structures and algorithms.
Deibel (2007), ACM ITiCSE, search: CSE & grounded theory	Inclusive practices in computing education (experiences of students with disabilities).
Deibel (2008), ACM SIGCSE, search: CSE & grounded theory	Inclusive practices in computing education (experiences of students with disabilities).
Dick et al. (2003), ACM SIGSCE Bulletin, ITiCSE working group report, search: CSE & grounded theory	How academics can handle student cheating.
Fitzgerald et al. (2005), ACM ICER, search: CSE & grounded theory	Problem solving; code tracing strategies.
Hanks et al. (2009), ACM SIGCSE, search: CSE & grounded theory	How students advice other students in learning to program?
Herman et al. (2008), ACM ICER, seach: CSE & grounded theory	Logical misconception in novice students' problem solving.
Hewner and Guzdial (2008), ACM ICER, search: CSE & grounded theory	Students' attitudes towards computing.
Jadud (2006), ACM ICER, search: CSE & grounded theory	Compilation behavior of novices.
Kautz and Pries-Heje (1999), ACM SIGCPR Computer Personnel, search: CSE & grounded theory	Methodology adoption of systems development students.
Kollanus and Isomöttönen (2008), ACM ITiCSE, search: CSE & grounded theory	TDD in education — students' experiences.
Melin and Cronholm (2004), ACM ITiCSE, search: CSE & grounded theory	Students experiences on project oriented course work.
Nagarajan and Edwards (2008), ACM ACE, search: CSE & grounded theory	Non-technical work experiences of information technology graduates.
Schulte and Knobelsdorf (2007), ACM ICER, search: CSE & grounded theory	Freshmen's earlier computing experiences affecting their view of CS.
Yuen (2007), ACM SIGCSE Bulletin, search: CSE & grounded theory	Novices' knowledge construction of difficult CS1 concepts.
Sudol (2008), ACM SIGCSE, search: CSE & grounded theory	Creating connection between life and class by introducing reading assignments.
Simon et al. (2008), ACM SIGCSE, search: CSE & grounded theory	How students' public note blogging has effect on the class.

(Continues)

TABLE 3 (Continues)

Levy (2001), JCSE, search: grounded theory	Students' perceptions on the concept of recursion.
Kolikant and Mussai (2008), JCSE, search: grounded theory	Students' (mis)conceptions of correctness.

3.3.4 Review questions and analysis

The starting point of the analysis was a set of pre-defined questions. The questions arose from the review author's own experience and discussional papers on the use of GT, such as (Adolph et al., 2008). I then conducted the review by allowing new viewpoints to emerge from the articles. This was because I wanted to explore the use of the method in a rich manner. I added the new viewpoints to the list of review questions, and repeated the analysis on the reviewed papers from the added new perspectives. The final set of the questions was the following:

- Q1: Had GT been used for theorizing or as a means of data analysis?
- Q2: How were the data collection and analysis described?
- Q3: Joint coding and analysis?
- Q4: Approach to GT?
- Q5: Correct citing (e.g. to Strauss when using Strauss' approach)?
- Q6: Can one identify a core category?
- Q7: What was the research process like?
- Q8: Preconception?

The questions are overlapping so that a certain perspective can help in deciding on the other. For example, a study demonstrating a core category (Q6) would perhaps have potential regarding the topic of theorizing (Q1). In many cases it was difficult to answer these questions, but in such cases it was decided to write any observations down as indicative notes. Because the aim of the review was to gain an understanding of GT usage in the CSE/SEE field, the following review results are of a qualitative nature. Numbers are occasionally given for additional illustration. Notes on the analysis are included in Appendix 1. The notes are from the first cycle of analysis of the papers, and thus a snapshot of the review process. Many notes were afterwards hand-written in the margins of those tabulated notes.

3.3.5 Review results

How to read the review results

The following analysis is based on reflection on the review papers according to the principles of GT given in the beginning of the chapter. The review does not claim that if the use of GT in a reviewed paper differs from these principles and guidelines, this would imply concerns regarding their contribution or reliability. The interest of the analysis is only to explore the adoption and usage of the method.

Research process characteristics?

Nearly all of the studies could be characterized as one-shot studies, which does not involve knowledge sophistication over a long period of time, but concerns pre-designed research where the data are collected first and then analyzed (25/30 ≈ 83 %). The deviations (5/30) from this mode are the following: Isomöttönen and Kärkkäinen (2008) give an impression that the student data add to the experience previously gained by teacher observations and the small student data analyzed in the study originate from a period of several years. William's et al. (2007) conceptual framework is re-developed upon a previous work. Last (2003) reports results from a three-year period in an on-going research project. Cheng and Beaumont (2004) include two action research cycles, with the former informing the latter. In Kolikant's and Mussai's study (2008), which appears as a one-shot study, collected survey results inform the conducted interviews. Within the remaining 25 of the one-shot mode, 6 of the studies (Ho et al., 2004; Almstrum and Last, 2006, 2005; Deibel, 2007; Nagarajan and Edwards, 2008; Jadud, 2006) are the initial steps of a potential follow-up study involving grounded theory, and in the remaining 19, a future work reference is often made so that the results are seen as tentative and could be furthered with more sampling.

Theorizing vs. data analysis?

If a strict GT requirement is applied, i.e., the paper presents a carefully explicated theory derived from data, with integration in the context of a single core attribute, using parsimony of terms (cf. conceptualization vs. description), and having a clear inner logic, none of the reviewed papers can be interpreted as being a grounded theory. Many of the papers demonstrated features of theorizing, if integration between categories is taken as a criterion. Last (2003) presents a typology based on the core attribute of team cohesiveness and four other themes. However, the study does not explicitly mention an aim for theorizing, and a reference is made to an external source regarding the core attribute whose emergence then remains unclear. In Kolikant's and Mussai's study (2008), students' conception of partial correctness stands out as a core attribute and is the central theme in the discussion of the grounded theory part of the study. However, while the study mentions the goal of theorizing it focuses on more than one core finding and does not itself explicate any single core attribute. A part of the study be Fitzgerald et al. (2005) focuses on emergent theories, and there the study demonstrates a temporal ordering of the student code tracing strategies, which integrates the categories found and thus provides unity in the findings. Williams et al. (2007) present a social interaction model of pair programming (SIMPP) that includes cyclic relations between the categories found. However, it is difficult to identify an emergent core, as the term SIMPP appears to be more a label for the model. Melin and Cronholm (2004) present explicit condition-consequence causalities, but no core attribute is highlighted. Shull et al. (2000) state tentative hypotheses which also demonstrate causal relationships. In addition to these, many of the studies group the categories found into higher level concepts or themes.

While one can identify potential regarding theorizing, these observations demonstrate that GT is usually adopted as a means of data analysis, and that conducting a "full" GT study would take plenty of time and concentration. Partial results are often published. Those sections of the reviewed papers that demonstrate theorizing are hidden within the traditional formulation of a research paper which differentiates between a data collection phase and the subsequent analysis. This differentiation, reflecting mechanically conducted research actions, is contradictory to grounded theory generation with joint coding and analysis (see Section 3.1.2) and a high intensity of concentration and thought (see Section 3.2.2).

This conclusion — hardly theorizing — is similar to Urquhart's (2002) questioning of GT usage in the IS field.

Core category?

Again, on the strict requirement, i.e., that theory is provided in the context of a single relevant, analytic and sensitizing, and emergent, core, it is difficult to regard any of the reviewed studies as meeting this requirement. Three of the studies ($3/30 \approx 10$ %) demonstrate a kind of main attribute but have all some concerns regarding it. Isomöttönen and Kärkkäinen (2008) derive a single core concept from their analysis on student data but yet another, the most analytic hypothesis, is not made salient in the paper. Last (2003) provides a conceptualization around the concept of team cohesion, but, as noted, makes a reference to such a concept, which makes it difficult to be certain of how the core was arrived at. Kolikant and Mussai (2008) foreground one of the findings and discuss it giving the impression that it is a core. In addition to these, the temporally ordered code tracing strategies could perhaps have been refined to a core (Fitzgerald et al., 2005), and the social interaction model of pair programming is a kind of core, yet seems more like a label for a framework (Williams et al., 2007).

Some problems are identified which may hinder arrival at a single relevant core category. The categories under which the discovered themes are presented are often neither analytic nor sensitizing; see Section 3.1.4. For example, the two main categories in the study focusing on method adoption are those of adopters and non-adopters — which do not analytically explain the behavior of those involved but are about some general taxonomy arising from the research question (Kautz and Pries-Heje, 1999). This use of a general taxonomy, arising from research questions, makes it difficult to arrive at a fresh relevant core, and may stem from the tradition of conducting qualitative research as a pre-designed study with carefully formulated pre-determined research questions. From the GT usage perspective, using strict research questions, or building the analysis upon an external theoretical background, which is then also used to interpret the results, in a pre-designed one-shot study, appears to limit and/or guide the analysis, e.g. (Yuen, 2007; Schulte and Knobelsdorf, 2007; Shull et al., 2000; Lappenbusch and Turns, 2005). Schulte and Knobelsdorf (2007) use a theoretical background as a lens for the analysis, and this appears to be carefully and purposefully the research approach of the study, and yet GT is referred to.

The implication is that more focus is needed in the process of arriving at the research questions. Thus, in GT, the lens through which the data is analyzed (cf. research question) should emerge from the data itself (see the arrival at the core category in Section 3.4.1). This process should be managed in a systematic manner using constant comparison. This would promote the formulation of emergent and relevant (in the sense of grounded theory) core categories. Within the dominant research mode of the reviewed papers, identified above, the research questions are pre-determined and how they are arrived at does not result from using GT procedures (cf. constant comparison and selective coding according to Glaser). How GT fits into the traditions of qualitative research is consequently a concern.

Approach to GT, use of citations?

Differences between GT approaches were not commented on in any of the reviewed papers. There seems to be a preference for the approach and techniques favored by Strauss and Corbin. Roughly half of the studies followed, referred to, or their work resembled the techniques advocated by Strauss and Corbin. Glaser was cited only in one paper (Lappenbusch and Turns, 2005), but in that paper other GT sources were also cited.

Correct — as opposed to misleading — citations were found in the case where Strauss and Corbin was cited in a study that explicitly named and/or used procedures of open, axial, and selective coding proposed by Strauss and Corbin, and the case where GT was generally mentioned by citing DGT, or DGT was cited as the original GT source. The following ways of referring to GT were problematic.

- DGT cited when the procedures of Strauss and Corbin were being referred to.

- DGT, Strauss and Corbin, and Glaser all cited without any commentary on the differences.

- GT mentioned/applied but not cited at all.

- Lack of clarity in method concepts due to secondary sources, i.e., concepts contradictory to those in the original sources.

Characteristics of the analysis?

Data sources and data collection techniques were generally given, e.g. a use of semi-structured interviews with a set of students. Analysis procedures were commonly outlined according to a literature source without describing the actual application of the methods used. Usually the categories found were supported with data extracts, which indicates interpretations from the data, but does not explicate the methodological perspective. Thus, while the papers describe the data collection, and assure their rigor, for example, by using several researchers to cross-check the data for agreement, only one study gave an explicit example of comparison (Shull et al., 2000). The methodological aspects of GT usage are thus generally omitted, and the rigor in GT is understood according to what is generally perceived as rigor in qualitative research. This observation is consistent with the observation above, i.e., the trend towards data analysis in pre-designed one-shot research instead of theorizing.

The joint approach means that the researcher is allowed to follow emergent areas of interest, change focus along the course of the research, thus allowing theorizing about what emerges as relevant. Only the studies by Deibel (2007; 2008) (the latter continues the former study) explicitly mention the joint approach. In most cases this is however impossible to interpret without interviewing the researchers. In general, as many of the studies present the data collection as a separate step, a conclusion is that the joint approach is not taken.

The studies that continue their previous work or that report findings from a period longer than one course offering demonstrate the character of joint coding and analysis without explicating it. The results are then snapshots of the on-going process. Isomöttönen and Kärkkäinen (2008) analyze student data on top of previous teacher experiences. Williams et al. (2007) complement previous findings with newly collected data from another data source. The findings of Last (2003) originate from a three-year research process. Similarly, the studies taking initial steps, by presenting a tentative categorical system or hypotheses, are potential joint approach studies, e.g. (Levy, 2001). It should be noted that many of the reviewed studies used semi-structured interviews with probing questions, and the interviewees were allowed to take the directions they found relevant. This is also in line with the joint approach.

Summary

The reviewed papers give an impression that the grounded theory method is adopted through a take-and-use strategy and inserted in the conventional research format, including the way of reporting such a study. While the use of GT demonstrates potential with respect to theorizing, the method often appears to be perceived as an alternative to conducting qualitative data analysis. Adolf et al. (2008) state that the use of GT procedures as data analysis techniques is acceptable, but the aim of the study (theory vs. thematization) should be explicated. Easy-to-remember theories in the context of a single core category — as opposed to a list of categories — could not be found without any reservations. The reviewed studies were often incomplete to be judged as representing a theory. Grounded theory takes time and under academic publishing pressures developing complete grounded theories may become problematic. This symptom may continue to arise in the literature.

Because of these observations and conclusions, it is seen that it is important to experiment with GT with the aim of following the original ideology and objective of the method. This is an intention of the present dissertation. The dissertation aims at theorizing generated over a several-year period. The dissertation provides a core category (Chapter 4), explicates the arrival at the core through use of comparison (Section 3.4.1), and it also starts from a reduced overview (cf. core) when laying out the action-research based conceptualizations. The dissertation aims to follow DGT and Glaser's approach, which is the opposite of that taken in the reviewed papers. The dissertation identifies the two main approaches to GT and discusses the selection of a particular approach. The use of comparison is illustrated, and this is seen to be more important than describing data collection techniques and data sources.

3.4 GT in this dissertation

The section first outlines how the research arrived at a particular approach to GT. This is followed by illustrations on selective coding, use of the constant comparative method, and the analysis techniques. To consider these in relation to the use of data, see Section 3.7.

3.4.1 Getting started

I first applied Strauss' and Corbin's word-by-word and line-by-line manner of coding (Strauss and Corbin, 1990, p. 72). I analyzed course feedback data collected with a questionnaire together with a student organization (see Section 3.7). I restricted the analysis to what the students had gained from the course, whether they had experienced some change as a result of the course. I noticed that the use of micro-level explicit coding resulted in a richer outcome of the analysis compared to the first reviews of the data. For example, I noticed that the students often started their response with a doubt or a brief modest statement, followed by a richer statement that carries the actual content for the analysis. I assumed that Finnish students tend to do this kind of thing when asked about their experiences. The data set was relatively small, which also allowed micro-level analysis. Technically, the analysis was managed in an excel table, by transferring the data to such a table, assigning concepts to one column, causalities to another, and finally selecting a main theme from the concepts.

I also applied a key point type of coding to the rest of the questionnaire data. While a richer set of viewpoints emerged with the micro-level coding, I noticed that the most relevant issues were derivable by focusing on key points at a higher level of abstraction. The most interesting outcome was the hypothesis that resulted from comparing the results of this key point coding and the main theme gained by the word-by-word micro-level coding. I discovered that despite having clearly encountered problems and experienced frustration, almost all of the students reported on personal growth when they were specifically asked whether they had experienced a change as a result of the course. This conceptual hypothesis later served as a starting point for the subsequent analysis with more extensive data (see theoretical sampling in Section 3.1.3), it suggested a core category (see Section 3.2.4). After discovering the initial hypothesis, I adopted the constant comparative method set out in DGT, and the ensuing approach of Glaser, which elaborates DGT. Selection of this approach was based on the experience that seizing at key points (cf. mechanical micro-level conceptual labeling) better promoted thinking about the data and allowed analysis at the very beginning of the study. I also found DGT and Glaser's approach to be more intuitive encompassing the right amount of rigor (not too much) and conceptual ideas to enable theorizing. I have no experience in using the procedures of Strauss and Corbin on a larger data set.

The above discovery finally led to the first results chapter of the dissertation. In the following, I illustrate the use of GT with another core category in order to not confuse the results of the dissertation with this explication on the use of the method. The category that will be presented is thus not the main focus in this dissertation but applied here to illustrate how the method is used. It relates to the first "full" GT process during the course of the research. It addresses students' communication challenges with real project customers. Both arrival at the core category and the use of DGT's comparative method will be presented. This communication thread of the research is published (Isomöttönen and Kärkkäinen, 2009). The following sections are based on the paper.

3.4.2 Selective coding and arrival at a core category

The following core category was generated using teachers' experiences and students' documented course experiences as data: *inexperienced students' entry into a realistic software development context reveals a potential communication barrier in the direction from the students to the customer.*

This core was selected on the basis of teachers' experiences that had accumulated during the history of the course (word-of-mouth knowledge). As I had a close relation to such experiential data (my own direct observations and discussions shared among the teachers) I also studied some of the course evaluation statements that project teachers had written for completed projects. The purpose of such sampling was to take account of subjectivity, and thus to be sure that enough attention had been paid by the teachers to the communication direction I was about to select for further investigation. The results of this sampling indicated that students' communication towards other stakeholders (both the customer and the instructor) was a central success factor in the course. This was in line with the word-of-mouth knowledge among the teachers. Through "teaching as coaching" -enabled observation on students' performance I became confident that in particular the direction towards customers was highly relevant. Afterwards, when starting to analyze the student feedback data (student experiences in publicly archived project reports), I found confirmation that problems with this communication direction lead to negative consequences from the perspectives of both students and customers. Problems in this communication direction mean that the necessary feedback loop between students and customers does not emerge and misunderstandings appear. As a consequence, wrong software functionalities are developed and a project may slow down considerably. The problem affects the project outcome and decreases students' satisfaction with their own performance:

> [ST][5] *At the end, a doubt arose as to whether we had interpreted the priorities correctly. The project focused on telecommunication and encryption modules, but was the customer actually more interested in database modules? Too often we made decisions by ourselves and did not utilize the teachers' and the customers' opinions and expertise.*

I thus considered this communication direction a relevant problem for those involved (teachers, students, and customers). Basically, this illustrates how a core category has to emerge and how the analyst must become confident about its relevance. Here the arrival at the core started from teachers' experiences, while Section 3.4.1 described how a core category was first discovered using student data.

3.4.3 Using the constant comparative method

The following analysis actions demonstrate the use of the constant comparative method, again from the research thread on students' communication challenges towards customers.

Comparison

The first explanation for the communication problem towards customers was inadequate awareness. Students may not recognize customer communication as a key area in software development work, implying that there may exist a communication problem in a project without noticing. This issue of awareness is indicated by the fact that the students usually reflect on the project areas which are "close" to their course work, including group work and technical learning. In the students' course experiences, such "close issues" have more volume compared to the issue of communicating with the customer. Inadequate awareness

[5] Throughout the dissertation, the marking [ST] in the beginning of a data example denotes a student comment from the student written project reports. The marking [STQ] denotes a data example from the anonymous questionnaire data collected for both a course evaluation and a research purpose. See Section 3.7 for data sources.

is also illustrated by a difference in the students' and the teachers' preferences. Whereas the students focus on "close issues" other than customer communication, the teachers, who have had a chance to collect experiences from several customer projects, have regarded this latter area as a critical success factor.

Then, by continuing to analyze incidents where student groups seemed to demonstrate difficulties in communicating with the customer, I noticed that despite a group becomes aware of the importance of communicating, it can hesitate to communicate. Here I compared an incident with the previously coded properties of the same category, and through this comparison found that there has to be another explanation for the communication barrier. The property that was identified was the students' timidity in initiating communication, i.e., timidity also seemed to explain the difficulties.

Integration

As the analysis proceeded, it was noticed that some students who had difficulties in noticing the importance of customer communication, demonstrated a clear technical orientation. These students continue working with design and implementation, not noticing that communicating with the customer is a necessity to get feedback on the outcomes. Here, integration was very straightforward: the individual background factor, students' technical orientation, explained the students' difficulty in noticing the importance of communicating with the customer.

Delimiting theory

An incomplete theoretical framework that emerged when coding student experiences in the communication GT thread is displayed in Figure 7. It originates from the point where the coding was terminated and time was spent on sketching the relations between the concepts. The drawing is a note copied exactly from the handwritten original in the coding booklet. The important issue here is reduction. The figure presents an unclear view of the findings as a whole. As the analysis proceeded, an overlap of the concepts in the figure's framework was noticed, and a reduction could be made. For example, it was noticed that both timidity towards and unawareness of importance of communicating with the customer can be explained in terms of inexperience. The outline of the conceptualization being generated could later be stated more clearly and with fewer main categories while still retaining a viable model. In this figure, the problem of awareness is only illustrated by a student comment "we should have communicated more in the beginning", but it was the first explanation for the teachers' experiences coded earlier in this research thread.

Figure 8 illustrates another attempt to reduce and explicate the logic of the findings. The contents of the drawing are very close to what was finally the output of this research thread; see (Isomöttönen and Kärkkäinen, 2009). Here, the student inexperience is a central category to which the other categories and their properties seem to interrelate. The category of how students can make progress

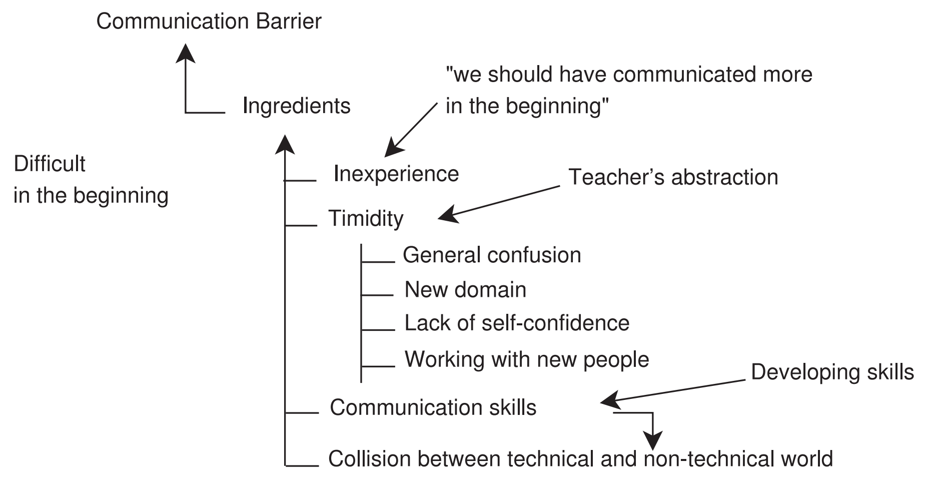

FIGURE 7 An incomplete theoretical framework based on the analysis of the student data

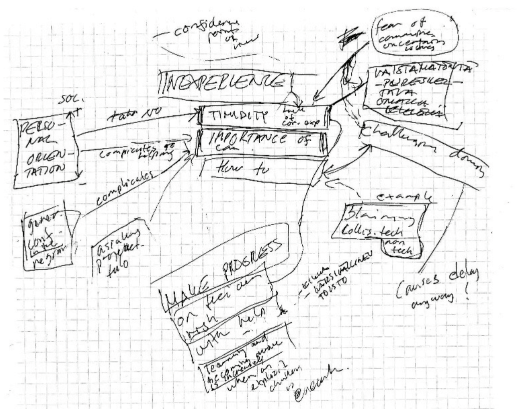

FIGURE 8 A draft of the overall conceptualization

regarding the communication barrier has now been identified. The third main category, that of how this problem (communication barrier) is individual to each student, emerges as the figure includes notes on students' personal orientations. These figures illustrate just a few of the attempts to reduce and explicate the findings during the research process. Here, as well in integration, one should clearly pay attention to the second rule of comparison (see Section 3.1.2). A lot of time is needed to make sense of the data in order to discover integration and delimit the theory being generated.

Writing

In DGT, it is suggested that writing a grounded theory (stage 4) starts from the main (core) category that puts forward the theory. This is followed by going through the main themes of the theory. This type of formulation was followed in the communication GT thread, and also in both of the results chapters of the dissertation. Glaser points out that when writing an outline of the theory, the analyst may notice that integration falls apart. He suggests that the writing should be started anyway, as it potentially leads to reintegration of what has fallen apart (Glaser, 1978, p. 132). Writing from this methodological perspective (as opposed to technical perspective) was considered very useful in this dissertation. The following quote captures the challenge that was encountered (Glaser, 1978, pp. 128–129): *Rather, writing must capture it. It must put into relief the conceptual work and its integration into a theoretical explanation. So very often in qualitative research, the theory is left implicit in the write-up, as the analyst gets caught up in the richness of the data.* Notice that Strauss and Corbin (1990, p. 129) also suggest rewriting if difficulties in integration are encountered.

The writing in the illustrated communication GT thread, and in the whole dissertation, was thus about reduction and explication, about making implicit knowledge sufficiently explicit. Drawings were often used in parallel with writing, and Figure 8 was actually sketched to support writing. Writing forced to think of sufficiently explicit conceptualizations at a high level of abstraction. The writing of small textual fragments and outlining the unified whole was very difficult. It sometimes took hours to formulate a fragment of the outcome in a consistent textual form that carried the correct idea, and these formulations had still to be reworked several times. The work could be characterized as continuous knowledge sophistication. This was not experienced as deduction and forcing but reduction and explication.

It should be noted that the writing of the theory in Chapter 4 attempts to remain at conceptual level. Descriptive data examples are given parsimoniously to keep the focus on concepts and their relations. This is in line with the guidelines given by Glaser (1978, p. 133).

3.4.4 Analysis from a technical point of view

In the communication GT thread, the analysis techniques included a use of a coding booklet for codes with data extracts, writing notes in the booklet, drawing diagrams, and writing the emergent theory. Figures 7 and 8 above demonstrate the use of diagrams. The GT thread regarding the first results chapter of the dissertation applied the same techniques plus a coding poster. Figure 9 provides a snapshot of the poster prepared while coding and analyzing the students' experiences. Here I transferred the coding booklet contents to a single large poster to better obtain an overview of the coding, to be able to consider theoretical codes, and delimit the theory. In my view, the techniques to be used are up to the analyst and are not a validity concern. The validity concern is whether the analyst is capable of conceptualization, able to remain close to the data, concentrate, think, and be patient — be theoretically sensitive (Piantanida et al., 2004); see Section 4.4.

Overall statements
Dualism
VALUE OF REALISM
AXIAL CODING
self-help
challenge
negatives
consequence
Notice

FIGURE 9 A snapshot of the poster prepared to manage the analysis

3.5 Action research

Action research is a wide research arena encompassing a variety of foundations and intentions, which complicates its definition. Many agree that action research is inquiry that is done by or with insiders in an organization or community, but never to or on them (Herr and Anderson, 2005). It is generally seen as a reflective process, meaning cycles of actions to be taken. The idea of the actions is that a change occurs and that the change can be studied.

The origins of action research can be seen in the work of Kurt Lewin, who is credited as the person first theorizing this type of research. Lewin's work made action research a respectable form of research in the social sciences (Herr and Anderson, 2005, p. 11). Lewin (1946) argued that social research needs to take account of the situation at hand, and that such research requires both fact-finding and experimental comparative studies on the effectiveness of various techniques of change. Lewin outlined action research as a sequence, starting from the initial step of identifying objectives and the means to achieve those objectives. During this initial step the first general idea is carefully examined, which requires fact-finding on the current situation. If the initial step is successful, an overall plan emerges and the first step regarding the actual actions to be taken is decided. After the initial step, the process is about taking actions and evaluating their outcomes. The evaluation may imply new planning and changes in the overall plan.

In short, "Rational social management, therefore, proceeds in a spiral of steps each of which is composed of a circle of planning, action, and fact-finding about the result of action" (Lewin, 1946, p. 38). Notice here that action research does not dictate the number of cycles needed in the study. For example, Blum (1955) speaks of two fundamental phases: a diagnostic phase to uncover the problem and a therapeutic phase to test the hypotheses aimed at solving the problem. The latter translates into consciously directed change.

Since Lewin's early work, several directions in action research have emerged. To name a few (Herr and Anderson, 2005), these include action research as organizational development, action science, participatory research, participatory evaluation, educational action research, the teacher-as-researcher movement in the UK, the practitioner research movement in North America, action research as self-study, and autoethnography. Some of them have a critical edge with regard to changing the status quo of an organization or a community (e.g. organizational development), some have aimed for greater humanization (e.g. participatory research), and some emphasize a reflective practice to create and share knowledge (e.g. educational action research). The theoretical foundation of educational action research is seen in the work of John Dewey, who saw human experience as an integral part of learning.

Hiltunen (2010, pp. 53–54) gives examples of how the literature often demonstrates three major lines in action research, even though different concepts are employed to describe these lines. For example, Grundy (1990, p. 353) differentiates between technical, practical, and emancipatory action research and McCutcheon and Jung (1990, pp. 145–147) speak of a positivist perspective, an interpretivist perspective, and a critical science perspective. Technical action research points to the solving of pre-defined problems with scientific procedures. Practical action research tackles problems that arise in the study context, and is interested in developing understanding of the practices that can solve such problems. Emancipatory action research refers to the aim of emancipating participants in the action from the dictates of compulsions of tradition. The most important view of action research with respect to the present work is the one where theory development is advocated. Such literature is referred to when describing the theory-oriented approach of the present action research in Section 3.6.

There are many critical viewpoints that need to be considered in doing action research. One of these is the role of the researcher. The term action research leaves the position of the researcher open, i.e., whether the researcher is an insider or outsider (Herr and Anderson, 2005, p. 3). Moreover, while action research aims at change, it is not evident what change constitutes an improvement (Herr and Anderson, 2005, p. 4). Perhaps the most interesting issue for me among such critical considerations is that the sequential form of action research is open to a literal interpretation which can lead to practice that is correct rather than good (Smith, 1996; 2001, 2007). Debate on action research has pointed out that this "iconic simplicity" has meant separating action research from its fundamental values (McTaggart and Garbutcheon-Singh, 1988, p. 410). At this point, I would refer to debate on the merits of two GT approaches, the one of which promotes

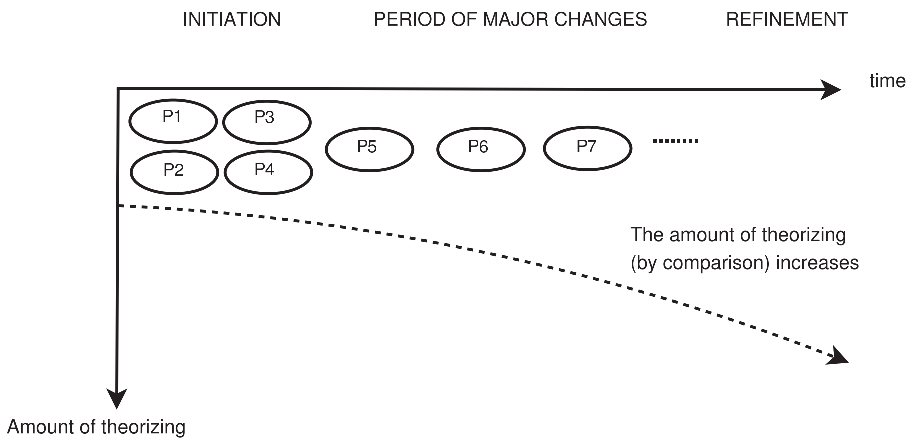

FIGURE 10 Nature of action research in this dissertation

theoretical sensitivity while the other offers a more proceduralized guidance on the method, and I would also point out that GT is often adopted solely as a data analysis technique. In the light of these viewpoints, action research followed as a proceduralized sequence of steps, yet aiming at theorizing, may share the critique on the adoption of GT as a series of discrete research steps in a pre-designed study, and thereby hindering theoretical sensitivity.

Action research has been applied in software engineering education research. For example, Dubinsky and Hazzan (2005) developed a framework for teaching software development methods.

3.6 Action research in this dissertation; theorizing in the action-taking context

The present action research is an example of educational action research, and can be characterized according to Johnson (1993), who states that through action research teachers become more reflective and attend more carefully their whole teaching process. With respect to the three-fold action research categorization given above (technical, practical, and emancipatory), the present action research has an emancipatory motivation, as discussed in the Introduction, and aims to develop understanding of local (emergent) challenges, as is the case with practical action research. Here, the development of understanding had a clear theorizing aim (cf. description), for which the approach taken could be characterized as "theorizing in the action-taking context".

3.6.1 Rationale for the emphasis on theorizing

It is important to give a rationale for the emphasis on theory development. As stated, this work views project education holistically, as it is not assumed, that a fundamental theoretical view of how to manage a realistic course can be found by focusing solely on a specific question about project work, e.g., "how can a project be started efficiently?". The position here is that if one is concerned with real-life events, such as student projects, where customer types vary, software domains vary, etc., i.e., there is a lot of natural variation, one needs to develop (middle-ground) theory that can cover such variance. It is this theory that can guide in managing teaching. From this position, instances need to be compared so that the resulting theory is an abstraction covering several issues and the variation involved.

Altogether ten projects were supervised during the action research undertaking. With this theorizing approach, the temporal order of project groups — and the associated teaching experiences — is less important than developing knowledge from similarities and differences. Theorizing becomes, as opposed to a description of events, free from time, place, and people (Glaser, 2002). Figure 10 illustrates this theorizing-oriented action research process in which the amount of theorizing grew towards the end of the process, as the project instances (PI:s in Figure 10) could increasingly be compared with each other: the greater the emphasis on theory development, the less important the temporal order. Theory development is about making abstractions and integrating at a high level of abstraction. The emphasis is on the vertical direction of increasing abstraction over the horizontal direction of improving a certain action.

In order to demonstrate that it has been acknowledged that social problems are holistic in nature, I again refer to Lewin's (1946) work. He used intergroup relations as his example case and stated: *An attempt to improve intergroup relations has to face a wide variety of tasks. ....We are beginning to see that it is hopeless to attack any one of these aspects of intergroup relations without considering the others.* He also stated that there is a need for comparative research within the action scene: *The research needed for social practice can best be characterized as research for social management or social engineering. It is a type of action research, a comparative research on the conditions and effects of various forms of social action, and research leading to social action.* Lewin seems to refer to comparison between instances where conditions are also "inside" the comparison, and thus not only the effects, by assuming that the context is stable.

Because of this theorizing orientation, the results of the present action research are not given in the form of a cyclical action research sequence. They are given under a single conceptual framework that provides the core for the whole results chapter, and hence, in terms of grounded theory, resembles a core category. This means that the timeline of the present action research is rarely foregrounded in the results.

3.6.2 Theory development in the action research literature; the rationale continues

Theory development in action research is gaining more and more attention. Generally speaking, the statement by Dick et al. (2009, Prologue) well characterizes the position taken by this dissertation: *if informed action is the goal, how is the action to be informed if not by theory?*. There seems to be a quite direct association between this statement and the grounded theorian one by Glaser 1978: *Men in the know easily become locked-in or status quo-oriented, as their knowledge becomes stable, consistent and consolidate their position.* `With theory` (my emphasis) *their perceptions are more amenable to change since they can begin to see the processes making for change and can modify their ideas to handle the new knowledge. They begin to transcend what was seen as inviolate, by seeing precious happenings as merely elements of patterns in process.* `They can work with familiar occasions purposefully` (my emphasis).

There are many action research references where theory development is advocated. Genat (2009) points out that many action researchers see the action component itself as an end result, and calls for the generation of situated emergent knowledge (local theory) in the context of participatory action research. Here a specific participant group develops local knowledge about their situation together with a researcher, which will eventually help in communicating the position of the participant group for the other stakeholders in the local scheme of things. The key challenge stated by Genat is arriving at a key theme that reveals critical elements in the experiences of the participant group. I observe that this challenge resembles the emergence of a core category in classical grounded theory. The epistemological status of the findings in the Genat approach exists as a truth of the participant focus group regarding a particular phenomena at a particular moment in time. Genat emphasizes verisimilitude in the findings, but does not directly speak of real mechanisms, which are emphasized in this dissertation; see Introduction.

Huxham (2003) clearly states that action research can contribute to understanding about phenomena (cf. the focus on action as an end result), as it can lead to conceptualizations about what can happen in practice and the reasons for this. In line with classical grounded theory, emergence is the guiding principle, meaning that predefined conceptualizations should not be used to guide the data collection, although it is also acknowledged that predefined theory can act as an opener and direct attention to aspects that could otherwise be missed. Huxham refers to the time-consuming aspect of theory development by pointing out that theoretical insights gradually become enriched and refined. Naturally occurring data are valued over interviews, and the term theory is associated with descriptive theory that can capture the experienced world.

Poonamallee (2009) presents an interesting action research framework where a great deal of attention is given to the self-reflexivity of the researcher. She explicitly refers to the development of grounded theories. The grounded theory method of constant comparison is adopted, and used, in particular, for clarifying the position of the researcher. That is, the world view of the researcher towards the phenomena under study is compared with the world views of the people the study principally concerns. This is said to lead to objective epistemology, which thus means exploring subjective experience objectively. Interestingly, I reflect similarly on the mindset of the researcher when, merely, engaging in the comparison procedure introduced in classical grounded theory; see Section 4.4; I refer to this mindset as an “objective mindset”. In Poonamallee’s framework the action research outcome consists of theoretical categories and the researcher’s stance towards the categories is carefully explicated.

The closest position in the literature to the present action research is stated by Friedman and Rogers (2009), in that their view of theory development is anchored in critical realism. This means that research arrives at and brings to the foreground the real mechanisms (unconscious patterns, generative structures) present in the action scene. Friedman and Rogers do not refer to the grounded theory comparison procedure, but instead quite literally aim at revealing the unconscious generative structures embedded in the participants’ views. They intervene in the research context and theorize in the background, and then test and refine their theoretical hypotheses together with the research participants. To be able to reveal unconscious patterns they need be alert to multiple and conflicting meanings in the participants’ views. The authors refer to the research result as an “actionable theory”. The benefit of this theory-of-action approach is rather similar to the one stated by Glaser above. Thus with theory, practice can be approached more purposefully.

Debate on theory development in action research revolves around the issues that are also found in the grounded theory literature: emergence vs. preconceptions, arrival at a core category, epistemological positions, and the view of ontology. Rather than emphasizing the cyclical sequence of change these theory-oriented references speak of interventions in a close connection to how emergent theory can be developed in a particular research setting. The references above give a rationale for the action research approach of this dissertation, as they also put the accent on developing emergent theory in the context of action taking.

It can nevertheless be observed that while theory development is advocated, it is done with a moderate tone. Friedman and Rogers (2009) point out that skepticism about theorizing in action research is rooted in the action research community’s critique of positivism and its attempt to develop general laws of human behavior. In this connection, I might think aloud that the position and the value of middle-ground theory is not widely recognized. Such theory becomes free from time, place, and people, while, at the same time, does not associate with fixed laws.

3.6.3 Researcher’s role

The teacher acted as a researcher, implying a clear insider role. I find this a beneficial position because an insider knows the context and from that perspective the interpretations are more likely to be viable. On the other hand, as I referred to above, my experience is that comparative theorizing gives the researcher an objective mindset, which in turn gives the researcher an outsider role. Thus I could speak about an insider-outsider role, in line with Poonamallee (2009), who proposes the objective analysis of subjective experience. In this connection, I should also point out that my course development work was challenged by other teachers in frequent discussions. My views were agreed with, complemented, or disagreed with. These discussions were thus critical and forced the acknowledgement of personal bias. Student views were also incorporated into the action research theorizing owing to the parallel grounded theory research, as stated at the beginning of this chapter.

In the set of ten projects supervised during the present action research, the interaction between teacher and students was purely educational. The students did not consciously take part in any of the action research. The actions to be taken were decided in the background, or they took place as a natural progression in the course of the projects. They were analyzed and discussed in the background and the conclusions were fed back into practice. Huxham and Vangen (2003) point out that when action research focuses on developing conceptual tools instead of addressing the situation within a certain participant group (e.g. an individual, an organization), it is not necessary that participants are aware of the research aspect of the intervention. This applies to this dissertation work where the research aimed at developing theory for managing the course instead of analyzing the positions of certain student groups.

Theorizing in the background of teaching means that "observation" is associated with retrospective reflection on teaching situations from the perspective of how to manage the course. I could characterize this position by putting my self in the place of someone asked to explain how a realistic project course is managed. In this sense Chapter 5 is teacher reflection that through the grounded theory comparison procedure takes the form of theory. This emphasis on reflection is supported by Eden and Huxham (1996), who point out that developing theory implies that valuable research insights emerge and cannot be foreseen. It is thus not clear what observations or situations will be indicative with respect to the emergent theory. Accordingly, in this dissertation process certain experiences (or situations) in naturally occurring activity turned out to be indicative in a retrospective manner, when reflective theorizing was engaged in.

3.6.4 The larger-scale phases of the present action research

While the above discussion refers to action research, which, like a grounded theory process, has an implicit nature, three larger-scale phases can nevertheless be identified in the present action research. These are *initiation*, a *period of major changes*, and *refinement*. *Initiation* refers here to the identification of the starting situation, the main problem, the objective, and a strategy. *Period of major changes* literally refers to the time during which major operational changes were undertaken, and *refinement* is a phase particularly oriented to theorizing, leading finally to the formulation of Chapter 5. The three phases are further explained below.

Initiation: 2005 autumn – 2006 spring

Background

I pair-supervised two projects with two experienced supervisors in spring 2005. I was hence able to consider teaching on the course from two perspectives. While I was the supervisor-in-charge in these projects, the colleagues I worked with took part in all parts of the projects, shared a great deal of their time familiarizing me with the work, and in particular discussed the various issues encountered during the projects. Based on this first experience as a teacher on the course, I wrote down my first conclusive experiences (success factors). The important thing was that I acquired a wealth of word-of-mouth knowledge of supervision work and became closely acquainted with the research context.

Starting situation

I was given the responsibility to further develop `TIES405` from autumn 2005 onwards. `TIES405` was already a very mature project course from many stand points. The maturity can be illustrated by comparing the course against the taxonomy-like paper by Clear et al. (2001) on addressable issues in devising a project course in computing. Several (if not all) the areas of concern identified by Clear et al. are addressed. For example, students are asked in advance whether they have sufficient resources for the high workload course. Concerns regarding potential conflicts with customers are addressed in a project contract template which defines responsibilities and entitlements of the stakeholders, including how to act if serious problems occur. Negotiations between the stakeholders are started, additionally with a smaller management group in which all the stakeholders are represented, and in the end, a solution is sought in court — the last has never been encountered. With respect to maintenance and liability issues, which are regarded as major challenges in real-world student software projects (Sun and Decker, 2004), the contract model defines where the responsibilities of the students and the department end. Project topic selection, student selection, and allocation of students to groups are conducted following well-defined procedures, and so forth (see Chapter 2).

Main problem

Despite the maturity of the course, some challenge had remained in practice and was identified as local motivation in this research: **the main problem was difficulty in getting the real customer projects completed on time.** This was an issue that had to be resolved: customers pay a fixed project price to the department and a project delay creates an annoying situation that risks student rights.

This “tension” in completing projects on time is similar to that described by James (2005):

A successful application will require modifications and continued “customer *support" as the sponsor becomes familiar with its operation.*

James suggests the following solution to the problem:

The faculty makes and satisfies the promises to the customer - no matter how open-ended and non-guaranteed.

This kind of solution, which has occasionally been followed in `TIES405`, was considered problematic for at least the following reasons:

- The educational organization mediates a view of working life in which a service provider needs to bend under any demands made by the customer. The work-without-limits view is especially risky at a time of, internationally noticed, student decline in computer science, see e.g. `http://www.cra.org/CRN/articles/may05/vegso`.
- The educational organization regards an external project customer here as its primary customer instead of the student who should preferably enter and exit his or her seat of learning with pride.
- Continuing work on a project after an initially scheduled period imposes unpredictability on the curriculum. Even if a project is continued on another course so that students' work can be compensated with credits, as is the case in James' paper, students' study design might nevertheless be dictated by continuing demands from the customer.

On top of this "snowball" character of customer projects, the tension in project completion at JYU/MIT was also caused by the large set of process deliverables and academic expectations of these deliverables, as indicated by student data and the present action research efforts (see Sections 2.6 and 5.3.1).

Objective

The dissertation presents a different solution from that given by James. The focus is on operational issues in the running of the course, instead of how to resolve the problem by changes in the prior or the post curriculum or changing the course constraints (the one-semester timescale and collaboration with real customers). The action research of the dissertation aimed to improve the course implementation so that the course becomes feasible with respect to the stated problem. More precisely, **the objective of the action research was to find an operational approach that enables the production of software for project customers in the fixed one-semester timescale without neglecting student learning.**

Strategy

The action research strategy was to strictly adhere to the one-semester timescale and support students' work as much as was needed to monitor what was actually going on in their project work. I started to consider teaching as coaching.

From the software process perspective a move from a waterfall process to short iterations started, first adopting an incremental development method with two or three increments during the semester — this initiative was partly based on a departmental suggestion to modernize the course. Appropriate roles and concrete guidelines for teachers and students were identified and suggestions on an applicable process approach made. These findings, achieved through teaching experiences and reflective discussions with colleagues, were documented into a course development plan (Isomöttönen, 2006) during the initiation phase. This was the overall plan that was further elaborated when the first actions were still being taken. In this sense the course development work did not consist of discrete steps but was continuous knowledge sophistication from the very beginning of the research.

Period of major changes: autumn 2006 – spring 2008

Major changes commenced in autumn 2006. For example, from the software process perspective, I gave up the sequence of a few increments, and started to use short iterations in the projects. The major step here was the first of such projects, the pilot project where the supervisor of this dissertation provided a project topic and acted as a customer. This arrangement allowed relevant retrospective research discussions[6]. I also reduced the amount of traditional project documentation, encouraged the students to use electronic management systems, simple excel files and the Trac system, and coached the students actively, almost on a daily basis.

Towards the end of the phase, the actions taken in the pilot project could be repeated and more detailed knowledge was generated. For example, I started to notice the key benefits and challenges in coaching, and not simply that coaching is the appropriate instruction method in the course. This is when the action research appeared to lead to a GT-like investigation: conceptualizing experiential data by comparison between cases.

Ideas derived from this phase were published in (Isomöttönen et al., 2007) and (Isomöttönen and Kärkkäinen, 2008). These were still interim results.

Refinement: autumn 2008 –

During this phase, I reacted to student feedback by making small changes to teaching, and considering the effects of these changes. In some cases, the research moved backwards somewhat in order to clarify the ideas found earlier. Overall, the refinement phase was more one of critical evaluation on the actions and the associated project cases (a thought process) than taking further actions. While all of the phases allowed for theorizing, this was particularly the focus of the refinement phase: the findings still in a more or less implicit form had to be translated

[6] The project topic was not pre-packaged for the research purpose, but explored how to automate the construction of a conceptual model from use case specifications, and later resulted in a publication (Kärkkäinen et al., 2008).

into explicit knowledge. The output of the refinement phase is this dissertation (Chapter 5).

This phase (time period) was dominated by the use of grounded theory, a parallel research undertaking, which provided further insights into the conceptualizations generated by the action research. As outlined at the beginning of this chapter, some of the grounded theory results are presented and referred to within the action research results because they, as subject matters, fit into the conducted action research. One of the most important thing here is that the grounded theory research added an explicit student perspective to the action research through a systematic analysis of student course experiences over a long period of time. This resembles data and method triangulation (Denzin, 1978) widely acknowledged in action research, e.g. (Elliot, 1976). The student data was in accordance with the theorizing in the action taking context: the data indicated, both directly and indirectly, the same areas of concern that had been identified in the course development work. For example, the students had noticed some process overhead in their work, the applicability and inapplicability of certain practices (e.g. prototyping vs. detailed design), and a need for an efficient project set-up phase. The summary of the research methods in Section 3.8 will illustrate the relationship between the grounded theory themes discovered and the action research conducted.

3.6.5 Technical perspective

Technically, the action research process was about *iterative writing*, leading, first, to the course development document (Isomöttönen, 2006) during the initiation phase. During the major changes, reduction was made and the results were presented at a higher level of abstraction (Isomöttönen et al., 2007). Some of the viewpoints are also found in (Isomöttönen and Kärkkäinen, 2008). These interim results were finally elaborated for this dissertation during the theorizing-oriented refinement phase. A relevant comparison can be made here to what was presented in the GT section, particularly regarding the delimiting and writing of a theory. Thus, towards the end of action research process, the emphasis turned more and more towards reduction and explication, using writing and diagramming as a methodical means.

3.7 Data and data usage

The data applied in the research consist of several sources. These are:

1. teacher experiences
2. course development plan
3. questionnaire data on student experiences aimed at exploring teaching quality

4. course evaluation statements
5. students' documented course experiences

As outlined at the beginning of this chapter, three grounded theory processes and a larger-scale action research process were undertaken in the present research, and the two kinds of research methods involved (GT and action research) were employed in parallel. To make a rough division, it can be stated that the principal data source of the three grounded theory processes was the students' documented experiences (item 5) whereas the action research result is principally based on teaching experiences about the ten projects that were supervised during the research (item 1). Use of the data sources is described in more detail below.

1. *Teacher experiences* have accumulated into a shared experience during the history of the course. Of the three grounded theory processes (see Figure 6), principally the one on the students' communication challenges utilized teacher experiences. In this connection, the experiences (observations) originate from reflective discussions among the teachers or they are directly connected to a specific case (project). I tracked the origins of the teaching experiences that were used. I cumulatively added the tracking information into a text-file with information including the human source (a discussion with a particular colleague) or the project name, and additionally as much time-related information as possible (e.g. discussion with a colleague in the middle of the first pair-supervised project). This helped in managing the research. Within the action research process, teacher experiences principally relate to the ten projects that were supervised under the action research undertaking and they were incorporated into the iterative writing process (see Section 3.6.5).

2. *The course development plan* is a direct output of the initiation phase of the action research process (see Section 3.6). It is not attached to this dissertation as such because it was not written for a public purpose. It was not direct reference material for the students, but consisted of diagnosis and an improvement proposal for the department. It is based on teacher observations, and was reviewed by the departmental committee twice during the writing. It served as a data source for both the conceptualization effort within the action research (the conceptualizations are basically reduced and elaborated from the plan) and GT coding concerning the students' communication challenges.

3. *Questionnaire data concerning the students' feedback on teaching quality and the students' course experiences* was collected together with a student organization and project teachers. The student organization was in contact to the respondents, and the purpose was primarily to investigate teaching quality in the department and potentially use the results in the present research. The questionnaire was targeted to students who had participated in the project course earlier and only a minority of the respondents had just finished their

project. Questions were answered anonymously. The analysis of this data and associated student quotes can be found in (Isomöttönen and Kärkkäinen, 2008). In this dissertation, this data set principally served as a source that suggested a core attribute regarding the first results chapter of the dissertation, as was explained in Section 3.4.1. Some quotes from this data set are also given in Chapter 5 when referring to anonymous course feedback.

4. *The course evaluation statements written by project teachers for completed projects* were used in the "communication barrier" GT thread when evaluating the potential researcher's position in the teacher-as-researcher setting and the relevance of the core category that was being arrived at; see (Isomöttönen and Kärkkäinen, 2009). These are also referred to when considering communication as an operational key practice within the action research results in Chapter 5. The evaluation statements are publicly archived (in printed form) in the project premises of the department.

5. *Students' documented course experiences* can also be found in the publicly archived project reports. Student project reports were sampled from the years 2000–2007. The reports contain the experiences of 121 students. The length of each experience unit (a section in a report) ranges approximately from a third of a page (*A*4) to two and half pages of text. The projects were selected with the aim of providing a rich view on the projects. They involved both in-house and external customers as well as the participation of seven different teachers. This was the main data portion of the project reports, but the sampling was not restricted to these alone. For example, I occasionally studied the student experiences in the projects, where I assumed to identify problems (by project grade), to see what issues had been paid attention to and whether the students' overall tone had become positive or negative (cf. theoretical sampling).

   The project reports are public documents. They are inspected by the project stakeholders and publicly available for subsequent project course attendees as reference material. In all cases the documents are checked by the customer to ensure that they contain no confidential information. The project contract used in the projects with external customers includes an agreement to use project outcomes for research purposes. In the case of public sector projects without a contract, the university's research policy did not identify any constraints on their use in research: all the students have taken the course with similar expectations of the later archiving of the results. Altogether, this data set was made anonymous for any reporting during the research.

   As highlighted in Chapter 2, the course context often varies across project groups. For example, the size of the project organization varies due to the varying number of customer representatives and end users involved, and software domains and problem domains vary because the customers represent a wide variety of areas of life. This temporary nature of a single project is further emphasized by the fact that the students undertaking the course do not share previous projects. Usually the student do not know the people involved in the project beforehand. The data thus represent many meaningful comparison groups (variation) in a single substantive area (a variety of student project instances in software engineering education).

The analyses conducted in the context of the three grounded theory processes focused on the students' experiences found in the project reports. In particular, the grounded theory presented in Chapter 4 got started from the questionnaire data but was developed by focusing only on this data source. Quotes from this data source are also included in the action research chapter. This is because parts of the grounded theory coding were integrated into the action research results and because it was noticed that the students' experiences indicated issues similar to those observed under the action research undertaking.

It should be noted that the nature of theorizing (grounded theory) is not about working with a predefined data set but instead the analyst is, in a way, surrounded by data. This was also the case here. While the theory presented in Chapter 4 was primarily generated from the data written by the students, the teacher-as-researcher setting means that a plethora of experiential data (e.g. informal discussions with colleagues and students) are incorporated into the thought process of the analyst. Similarly, while the theory in Chapter 5 was mainly generated on the basis of the experiences gained from the projects supervised under the action research undertaking, it is inevitably affected by the other data sources.

3.8 Summary of the research process

Figure 11 depicts how the method and data usage were located in the research timeline. The action research phases are marked in the uppermost part, GT processes in the middle part, and the data sources in the lowermost part. The research started in the form of action research. The basis for the research was the word-of-mouth knowledge that was shared in the form of pair-supervision in the spring semester of 2005. This led to the planning of a course development strategy and formulation of the course development plan in autumn 2005 and the first half of 2006. Course development actions took place and reduction to the resultant conceptualization was made at the end of 2006. As a result, a conceptual framework was outlined in (Isomöttönen et al., 2007). As can be seen in the figure, the research emphasis then shifted to GT (in 2007). The first theme of GT-based research, "enduring value of realism", was started by the questionnaire designed to investigate the quality of the course (see Sections 3.4.1 and 3.7). After this, other GT themes were also arrived at and then elaborated, the focus alternating between three major GT themes discovered in the data. The themes were:

- the “enduring value of realism”, which was finally elaborated into the first results chapter of the dissertation.
- the “communication barrier”, which I used as a reference for my use of GT earlier in this chapter. In Figure 11, it is directed towards the present action research, as it addresses one important area of the action research results. I thus refer to it in the action research part of the dissertation (cf. method triangulation).
- the “one-track principle in teaching in a realistic environment”, which is presented in the action research part of the dissertation, because it further refines and is consistent with the overall conceptual framework given in that part of the dissertation (cf. method triangulation). In Figure 11, it is directed to the conceptual framework obtained by the action research process. I did not bring this to completion in terms of grounded theory.

The main outcomes of the research were finally achieved during autumn 2008 and spring 2009. The final research period, starting from spring 2009, was about the explication, reduction, re-analysis and writing of the conceptualizations given in Chapters 4 and 5. At this point, the literature was also further studied and compared to the results achieved.

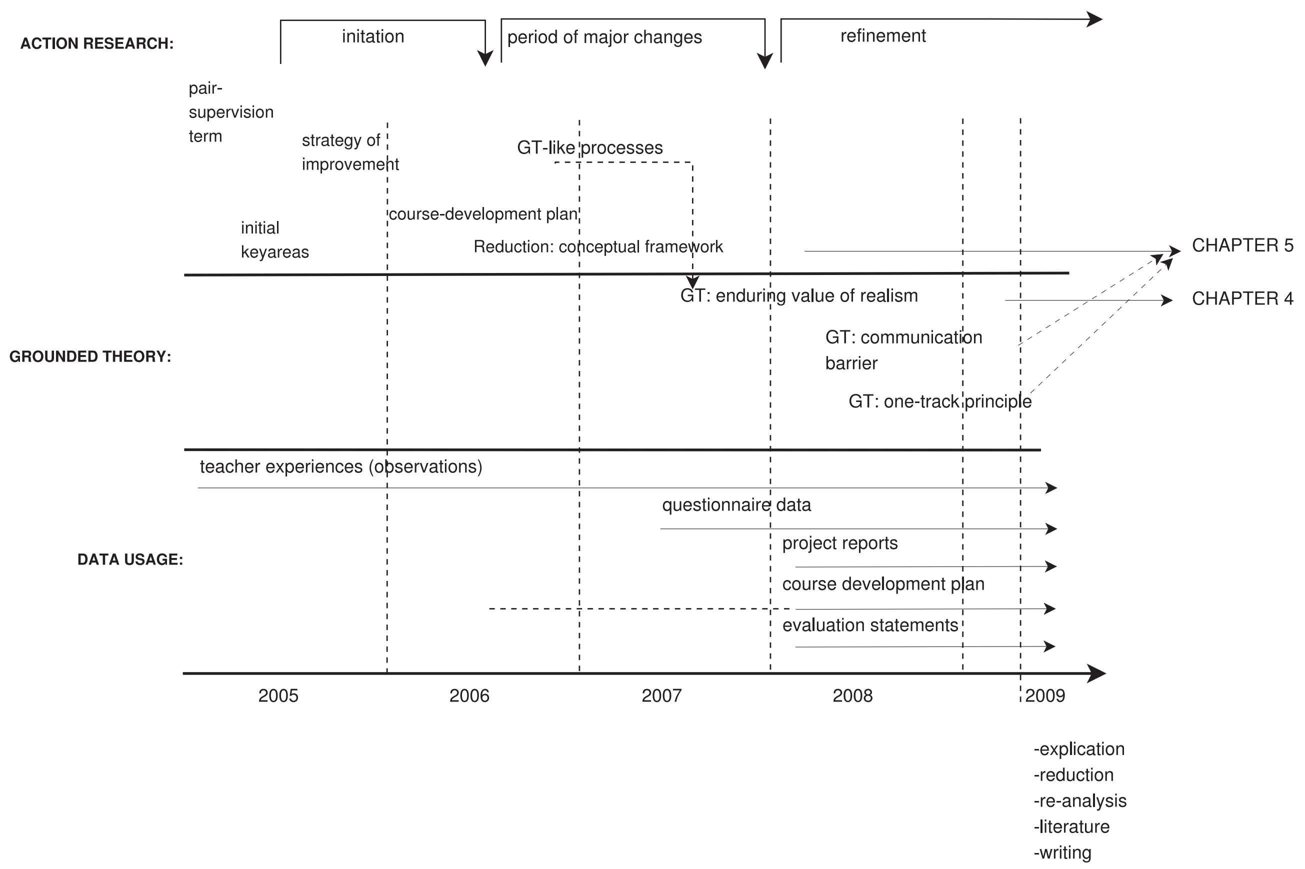

FIGURE 11 Research timeline

4 ON THE IMPACT OF THE ONE-SEMESTER REAL CUSTOMER PROJECT COURSE

The dissertation has so far outlined its motivation and approach (Introduction), research context (Chapter 2), and methods (Chapter 3). I now turn to the results. This chapter is the first of the two results chapters of the dissertation and will present a grounded theory. Here, a research theme having a holistic nature was arrived at through a grounded theory process. The grounded theory that will be presented is therefore a high level characterization of the one-semester real customer project course. It dissects the impact that the course has on students and was derived from the student data, as described in Section 3.4.1.

The grounded theory is first outlined around its core category ("enduring value of realism") in Section 4.1. This explains how realism has enduring value in the students' experiences, outweighing the problems the students encounter during the course. This (hopefully) provides an analytic viewpoint (see Section 3.1.4), as it *characterizes the impact* of the course instead of putting the entity "impact" itself into a dominant position. After the outline, the elements of the grounded theory are explained in subsections.

The present theory is reflected on in relation to the literature in Section 4.2 and then summarized in Section 4.3. The chapter concludes with methodological reflections in Section 4.4. This foregrounds the methodological considerations of the researcher and will serve as a discussion on validity and reliability.

4.1 Enduring value of realism

The grounded theory, titled "enduring value of realism", is outlined as follows:

> **The value of realism endures in the experiences of the course described by inexperienced[1] students, despite the number of problems encountered in the one-semester real customer project course.**

[1] The course has been the first curricular exposure to realistic projects for the students.

Realism creates an enjoyable experience, which is of a "run-time" positive nature, and a feeling of a necessary experience, which has a "future orientation"; these, the enjoyable and the necessary experience, outweigh the problems encountered. As a consequence, the students' overall course experiences[2] manifest positiveness.

The main concepts and the conceptual logic of the present theory are illustrated in Figure 12. The reader is advised to refer to this figure when reading the subsequent sections that explain the elements of the theory.

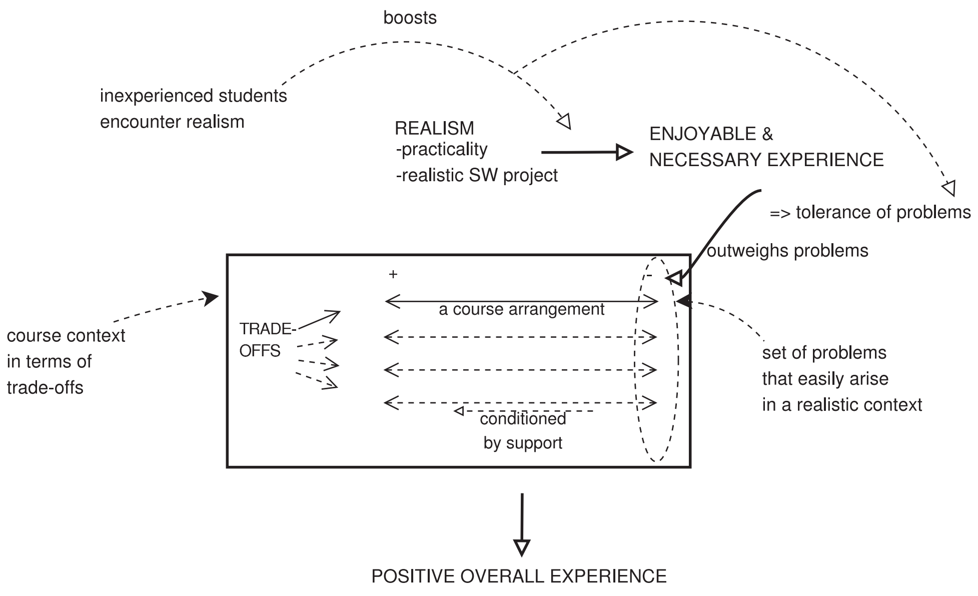

FIGURE 12 The grounded theory illustrated

4.1.1 Realism

To start with, the term "realism" needs to be defined in this context. Here, it is associated with 1) the practicality of the course and 2) the characteristics of a realistic software project: experience of group work with a real objective, dealing with real customer expectations in relation to a large and real project topic, intensive work with deadlines (cf. one-semester timeframe), and taking responsibility, as is necessitated in working life. While a realistic software project (the second item) is practical in itself, practicality (the first item) is also an issue for the students from another perspective. It has to do with how students experience different ways of teaching in the curriculum and hence has to be highlighted in its own right. A reduction is here made so that this viewpoint is also set under the term realism because the practical leaning context can be seen close to the real phenomenon here, i.e., real software development work.

[2] Notice the difference between "an experience" and "an overall course experience". The latter means a student's summing up of the course as a whole.

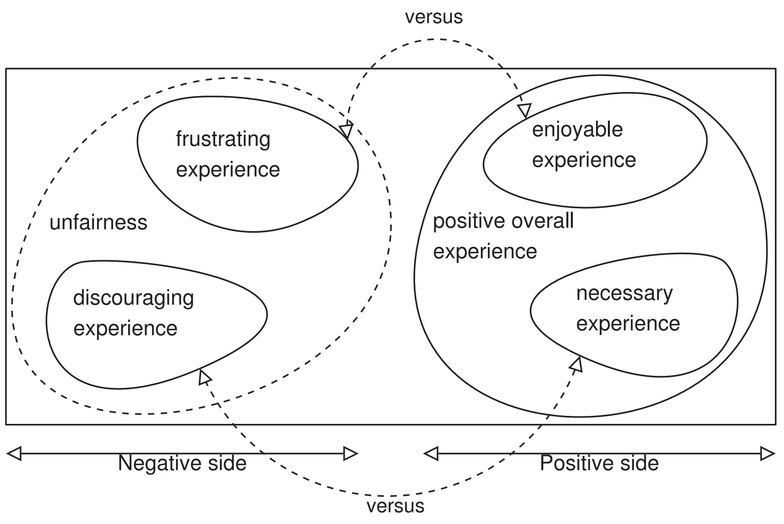

FIGURE 13 Types of student experiences

4.1.2 Course context in terms of trade-offs

With respect to Figure 12, the focus is now on the box titled “course context in terms of trade-offs”. This section illustrates the sensitivity of the realistic course context and will help in understanding the main flow of the present theory, i.e., the mechanism through which realism has enduring value and produces a positive overall course experience. The section will first present different types of students’ course experiences. After this it is illustrated how one and the same course arrangement can lead to different types of experiences, thus can be seen in terms of a “trade-off”, and how the students’ positive experiences are “conditioned by support” in the context of such trade-offs.

Experience categorization

In outlining the present theory above, the students’ positive experiences were reduced to an enjoyable and necessary experience. To help in communicating the rest of the present theory, it is necessary to present a reduction of the students’ experiences which also includes the students’ negative experiences of the course. Altogether, the course experiences map into an enjoyable, necessary, discouraging, and frustrating experience. These are illustrated in Figure 13.

At the positive end, two modes can be differentiated. “Enjoyable” refers to experiences that characterize the course as an enjoyable way of learning, as a run-time positive course experience. This enjoyable experience is triggered by group work with a “together-we-survive” spirit, attraction of realism, challenge, feeling of completeness, and practicality. Under real expectations enjoyable collaboration within a student group means having a *“together-we-survive”* spirit. The *attraction of realism* means that students are attracted by real problems and the associated technologies, responsibility for one’s own performance and towards real customers, and intensity. *Challenge* is perceived as motivation and this challenge-as-motivation association creates enjoyment. The *feeling of completeness* means here that students enjoy the project work as they get their deliverables and the project completed, and *practicality* means that the students enjoy the practical way of learning.

The other category (mode) at the positive end, "necessary experience", refers to the strong future orientation in the experiences, i.e., that the students experience growth and benefits for their subsequent careers. The feeling of a necessary experience arises because the project course reinforces students' occupational thought process, improves their self-conception, particularly self-confidence, and is experienced as very educative. The *reinforcement of occupational thought process* is indicated by the students' critical reflection on the performance and progress of the projects (a specific mode in the student data). The *students' self-conception* is improved as they get to know the different roles and tasks of real software development work and notice their weaknesses and strengths. A very important factor for the students seems to be the experience of survival in an authentic context and the associated improvement in their self-confidence[3]. *Educativeness* arises from the chance to internalize concepts in the practical course. This is experienced as an effective way of learning. The realistic context has a key role here as the students' conception of effective learning includes correspondence to a real software project.

At the negative end, "discouraging experience" is due to too much challenge and complexity, uncertainty in general, and difficulty in meeting one's personal learning objectives. The *too much challenge and complexity* means that students run into despair when the goals of the work are unrealistic. *Uncertainty* is associated with the aspect that a stationary target is not possible in a realistic project and students must work not knowing if they are able to fulfill all the initial goals of the project. *Difficulty in meeting personal learning objectives* arises from real expectations in a fixed time period: students tend to use their personal strengths to survive with the result that personal learning objectives are not taken into consideration when dividing up the work. Notice that discouraging experience is in a way a counterpoint to necessary experience (discouraging versus necessary, see Figure 13).

"Frustrating experience" arises in cases where the two audiences of the students' course work (academic vs. customer) create a contradiction, when the course is reflected on in relation to the other simultaneous and previous courses, and when unequal commitment occurs in group work. *Two audiences* means that while the academic side expects certain process deliverables from students, it may be the case that the most relevant tasks in respect of the project's goals (c.f. the customer's expectations), at a particular stage of the project, differ from these. The consequence is that students experience irrelevance in what they are expected to do. Frustration from *reflection on other studies* means two things here. Firstly, the students notice that the project course disturbs their simultaneous

[3] Thus, getting things completed in an authentic context means not only enjoyment but a future-oriented experience of survival.

studies, and secondly, they experience their credit compensation as insufficient, as they notice that they need to work more for a credit on the project course compared to previous courses. *Unequal commitment* is perceived as frustrating but the impression from the data is that students do not often directly bring it up. It has to be noted here that all the team members get the same number of credits, and this matches the lowest individual total work hours in the team. Thus, working more than others does not mean more credits. Notice, further, that 'frustrating" has a nuance of run-time experience as the counterpoint of enjoyable experience (frustration versus enjoyment, see Figure 13).

Negative experiences can be explained in terms of "unfairness" (see Figure 13). When a student strongly experiences some aspect of the course negatively his/her feelings are associated with the idea of unfairness, which also means that the positive overall experience is not that obvious. For example, when the course is perceived to considerably disturb other studies (and life), the overall course experience is dominated by the phrases such as the following:

[ST] *Occasionally there was no life outside the project.*

This negatives-as-unfairness viewpoint is pointed out here as it presents the negative case of the present theory. It also provides an important focus for teaching in this kind of course context: monitoring fairness/unfairness.

Trade-offs

After reviewing the different types of student course experiences, it can now be discussed how a single course arrangement plays a role in the students' experiences at both the positive and negative end of the experience scale, constituting trade-offs. The examples that are given concern the responsibility the students are given (and required to take), group work under real expectations, high workload, collaboration with real customers, and use of new technologies (programming languages, application platforms, APIs). These are given in full awareness of their differing nature: e.g., the responsibility is a more abstract concept than the use of new technologies.

Responsibility. `TIES405` is not structured as fixed lab sessions but is about a real situation in which students must take responsibility. This makes course work attractive (cf. attraction of realism leading to an enjoyable experience) and leaves space for students to think about their own performance (cf. reinforcement of occupational thought process leading to the feeling of a necessary experience). On the other hand, when students are given too much responsibility in an unrealistic situation regarding the project's goals, responsibility can create a discouraging experience. See Figure 14.

Group work. While group work is frequently mentioned as a source of enjoyment, it is a trade-off. It was stated earlier that enjoyable group work under real expectations can be characterized as a "together-we-survive" spirit. However, there are at least two ways in which group work can turn into a problem. On the one hand, it was noted earlier that frustration is caused by unequal commitment.

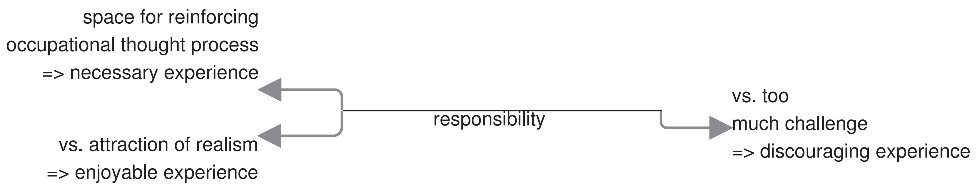

FIGURE 14 A low level attribute can create both a positive and a negative experience

Students notice that they really should make progress under real expectations, but, in the end, cannot take responsibility for their team-mates' commitment:

> [ST] *In this course we should practice correct ways of action, but still, in the end, each student can only take responsibility for his/her own actions. This complicates project management and progress. Perhaps the biggest failure was in group work and commitment. We discussed the issue within the group in the autumn. It nevertheless surprised me that all the group members did not have equal commitment to the project.*[4]

On the other hand, it was noted that under real expectations students cannot always meet their personal learning objectives, which can lead to a discouraging experience[5]:

> [ST] *My programming skills did not improve to the degree I wanted and expected before the project. This was perhaps the biggest disappointment in the project...I documented more and I surely learned a lot about it.*

High workload. Students explicitly and frequently bring up the high project workload in the sense of a trade-off. It is accepted as a feature of realistic project work and the associated intensity of the project course is experienced as attractive, contributing to an enjoyable experience. Also, the work load is understood as a part of the educativeness of the course, and then it is associated with the feeling of a necessary experience:

> [ST] *The course required much more work than lecture courses but was also much more educative.*

On the other hand, the workload is sometimes experienced as too much of a challenge, causing discouragement. It is felt both heavier than that of previous courses and seen to disturb other simultaneous studies, causing frustration.

[4] Here, the students' course feedback is focused on unequal commitment, which in a way steals the focus from other kinds of reflection. This is consistent with Tomayko's (Tomayko, 1996) argument discussed in the introduction of this dissertation. He argued that one improperly attending student can spoil the project if small groups are used. I would add that, while a project is likely to achieve some result in even such a situation, the learning (reflection) of other team members might be "disturbed".

[5] Notice that this problem is alleviated as the students working in small groups can at least observe all the project tasks from a close distance if not directly attend to some degree, and they have anyway meaningful tasks assigned to them.

Collaboration with real customers. Real customers with real project topics means open-ended and non-guaranteed work for the students. These contribute to a motivating challenge (cf. enjoyable experience) and experience of survival in a sufficiently realistic context (cf. relevant challenge and necessary experience). On the other hand, it is not axiomatic that the students' feeling of completeness and the associated enjoyment and feeling of survival will arise when all of the initial requirements cannot be met. Customer representatives may not know what they want and participate less than expected by the students. Students experience uncertainty and are stressed (cf. discouraging experience).

New technologies. Collaboration with real customers often necessitate the learning of new technologies (other than those used in the previous course assignments). Using and learning new technologies is attractive (cf. attraction of realism giving rise to an enjoyable experience) and contributes to self-confidence as the students notice that they can make progress using new technologies (cf. improved self-concept giving rise to a necessary experience). But all this easily turns the other way around. The students do not find that they have sufficient resources to learn a set of new technologies. New technologies impose too much challenge and complexity and attraction turns into a discouraging experience.

The high workload and the non-guaranteed nature of project work due to the collaboration with real customers are arrangements that are difficult to situate either in the negative or positive side of the students' experience scale to aptly characterize the students' experiences: whereas students' comments explicitly indicate that the workload is often an accepted character of the realistic course, uncertainty in the negative sense is often more about stress than discouragement. Thus, these factors cause a kind of accepted "heavy experience" for the students. This is added as an additional layer to the experience categorization in Figure 15. Notice still that whereas the workload is frequently situated in the middle of the experience scale, it can create a strong negative experience if it travels to the negative end: it is the dominant cause of a neutral or negative overall course experience[6].

While other trade-offs are also found in the data, and one can view these at different levels of abstraction (e.g., instead of speaking of a concrete course arrangement one can see a trade-off in the amount of challenge encountered by the students), the purpose here is only to communicate the conceptual idea of a trade-off with the given examples.

Positive experiences conditioned by support

As illustrated above, gaining an enjoyable and encouraging experience from group work is non-trivial. On top of this, a project topic may impose both a complex problem domain and a set of new technologies on inexperienced students. In many cases the course introduces several considerable challenges at the same time. The more new learning issues to be acquired the more easily a course arrangement travels from a positive cause to a negative cause. In such a sensitive

[6] An observation in the research notes, not calculated.

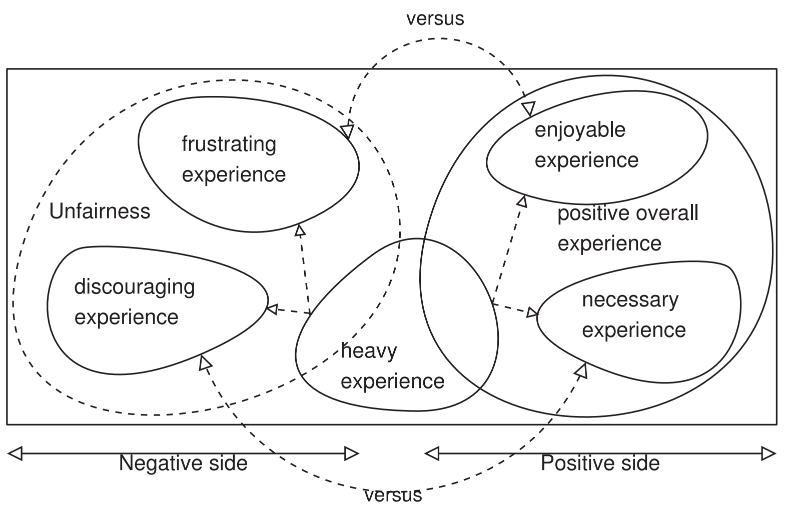

FIGURE 15 Types of student experiences: a heavy experience added

context, students' positive experiences are conditioned by support, which means different things depending on the course arrangement it relates to.

While the support is needed and can be directly received from any of the other stakeholders, the students' work is often indirectly conditioned by support from the teacher. Below, these two viewpoints are again illustrated along with the examples on responsibility, new technologies, group work, and collaboration with real customers.

Responsibility. An example of how a teacher must directly help students regarding responsibility is found in Section 5.2.2: a teacher must manage student expectations to avoid the emergence of a discouraging experience. Students must be helped to notice that not all of the challenge in non-guaranteed work is the students' responsibility. It is not a failure to not meet all of the requirements, which tend to change and become refined during the project.

New technologies. To avoid the use of new technologies turning into a discouraging experience students' need direct consultation. In the following example this is received from a fellow student and a technical supervisor:

> [ST] *In the beginning of the project I felt that I remembered almost nothing of programming, and I was doubtful whether my skills would be sufficient to complete this project. Nevertheless, during the project I noticed that I learned many new things on programming, and particularly John [a group member, name changed] was a great help in learning and doing ... The tutorial provided by Jack [a technical supervisor, name changed] at the beginning of the implementation phase clarified well my conceptions on using the particular library.*

Group work. This same quote (above) also provides an insight into how group work in terms of the "together-we-survive" spirit is conditioned: here a student gets help from a team-mate and is then able to get on with the programming tasks whereas the example in the previous section (see "trade-offs") illustrated how a student could not get into programming and therefore meet his/her personal learning objectives. Thus, support here means that the students in a group should allow everyone to become involved despite differences in initial skill levels. On the other hand, in many cases students thank their team-mates for a fair distribution of the work:

> [ST] *The other team members were kind and diligent, no one dawdled over their tasks.*

This means that each student should attend actively and challenge him/herself so as to prevent frustration on the part of the other team members, as illustrated with the data example in the previous section. Thus, support in group work means both *involving others* and *being in attendance with the others*:

> [ST] *I'm very pleased with the distribution of work within the team.* ***Everyone got to work properly*** *[everyone could get involved] and* ***no one was left in the lurch*** *[the team members attended actively].*

Collaboration with real customers. The need for indirect support from a teacher is well illustrated with the example of how the students' experience of survival (completing requirements with real topics) is conditioned by support from the customer: students evaluate the progress of their project and express the wish that the customers would have attended actively (e.g. given feedback) and been able to express their needs so that a shared understanding on the project topic could have been developed.

Taking into consideration that customers meet students frequently (agreed in the contract), support from the customer is likely to be a communication ability issue not only for customers but also for students. As foregrounded in (Isomöttönen and Kärkkäinen, 2009), to make progress in communicating with the customer, students often need help from the teacher. Thus, in order to receive support from the customer, students may first need support from the teacher[7]. Similarly, if teacher experiences are incorporated into the present analysis, it can be stated that meeting the challenges of group work is conditioned by the teacher's ability to intervene. Thus, to prevent a course arrangement (e.g. collaboration with real customers, group work under real expectations) from causing a negative experience, students' work is often indirectly conditioned by support from the teacher.

Here also, the purpose is not to provide an all-inclusive list of the different ways in which students' work is conditioned by support, but to communicate the conceptual idea with examples.

4.1.3 Boosting by inexperienced students' need for realism

The project course includes many features that are liable to create serious problems. This was illustrated already with the trade-offs. Moreover, such problems

[7] It is well known in the field that a teacher in customer projects must work in between the students and the customer, thus also help the customer (Fincher et al., 2001).

are difficult to manage, as was illustrated by how students' positive experiences are conditioned by support in a project course involving several stakeholders. One might expect to encounter negative overall course experiences, but this has not usually been the case.

In contrast, the project course fills students' need for realism, and realism fed to inexperienced students who are in need of such boosts the students' positive experiences. Because they are boosted this way, students tolerate negative experiences and the unfairness associated with them, and their overall experience becomes positive. Thus this is an important mechanism which explains why a positive overall course experience tends to arise.

Need for realism

It appears that inexperienced students *need* a chance to improve their skills in an authentic context, they *need* a chance to internalize concepts, they *need* a chance to experience survival. In short, they *need* to have a realistic experience during their education. An example is sourced from an informal discussion with the project students in autumn 2009 at the time of the project start-ups. I was not a project teacher during this semester. These students talked about how they looked forward to this course as they assumed it would give them more courage to apply for jobs after completing the course. This very aptly characterizes the students' need for realism: they had likely not gained a sufficiently realistic experience in the prior curriculum, and demonstrated a need to progress "from doubt to confidence".

On the other hand, students also prefer realism purely in the practical sense, as they compare the course work to lectures. As stated earlier, practicality is enjoyed and it allows internalization of concepts, but here the point is that the students experience something they have not had earlier in the curriculum. Before the project they look forward to getting involved in practical work:

> [ST] *I was excited about the project, it was interesting to get the chance to apply previously learned skills to practical problems.*

Boosting effect

The boosting effect (realism fed to students who are in the need of such) shines through the students' glowing summing-ups which reveal the uniqueness of the experience in the students' study history and also the value the student personally see in the experience.

> [ST] *The software project [the course] was in every way a rewarding and educative experience which I regard as **the most beneficial course in my study history.***
> [ST] *The software project [the course] is **a golden experience** of team work, communication, and system development and its problems.*
> [ST] *...now I know I have a chance to make it in working life.*

The boost given by a chance to do practical course work is also evident in the data:

> [ST] *Let's see how I feel about going back to lectures.*

Tolerance of negative experiences

Students tolerate negative experiences because their positive experiences are boosted by their unique exposure to realistic and practical work. This tolerance is indicated by a kind of dualism in the students' experiences. Negative experiences are mentioned but the value of the course is nevertheless highlighted in the students' concluding statements.

> [ST] *The project gave me more than I had thought it would, despite the amount of problems.*
> [ST] *A very heavy experience but I'm glad that I took part in the project.*

This tolerance is particularly indicated in the extreme cases where frustration, discouragement, and unfairness are clearly expressed and yet the students' summative statements about the course indicate positiveness[8]. The following extracts are from a delayed project:

> [ST] *All the members were amateurs, so I would have expected more support and instruction from the supervisors [cf. insufficient support leading to a discouraging experience].*[9]
> [ST] *The project focused too much on specification and secondary issues in documentation...We simply ran out of time because of this [cf. frustration and feeling of unfairness arising from the contradiction between the two audiences of the students' work: academic and customer].*
> [ST] *The software project [the course] was an educative experience for me personally. After many boring theory lectures the software project was the first chance to use all the learned knowledge in practical work...*
> [ST] *Even though mistakes were made in the project, more than that new things have been learned.*

It can be seen that the students bring up their inexperience and the course as the first exposure to practical software engineering work. The project was delayed by almost three months. Yet the students value the educative effect and enjoy the practicality of the course, and the impression given by the students' reports remains positive. Notice thus that even though one can identify discouragement and frustration, the value of realism endures. The value gained personally (*...was an educative experience for me personally...*) indicates the students' need for realism and that they anyway gain something.

[8] Examining such cases was about theoretical sampling; I wanted to know if this need for realism can create a positive overall course experience even in very problematic cases.

[9] Whereas the examples in the previous section illustrated how students need and receive support from their team-mates, technical supervisor, and customer, this example indicates how students' work is conditioned by support from the teacher.

It is important to conclude by noting and underlining the effect of the course's position in the curriculum as the students' first practical project course. This "first project course in the curriculum" is here associated with the students' inexperience, and in terms of GT, could be seen as an in vivo concept — a concept that can be directly derived from the action scene (Glaser, 1978, p. 70).

4.2 Comparison with the literature

This section examines the project course literature in the context of the grounded theory stated above. The literature is used as data so that the focus is on the experiences reported by students and teachers. It should be clearly noticed that the aim is to *reflect* and *discuss* the present grounded theory against the literature, as opposed to forcing the literature to fit it, and that this work with the literature took place only after developing the grounded theory above. I associate this manner of working with the on-going nature of grounded theory research, as discussed in Introduction of the dissertation.

Overall positive impact

Suri (2007) reports on a real-world laboratory project course, meant to take around 10 hours weekly, where students attend on-going real world projects, and are exposed to industry practices. They report that there is lots of overhead in the time spent, and the small number of weekly hours lead to a loss of student motivation and customer interest. Despite these problems, they report an overall positive course experience. This raises the question of whether here also the students' inexperience fed with some realism creates an enduring experience outweighing the problems. Based on the paper, it appears that the reported course is the students' first project study (cf. inexperience). The major component in the course is the learning of relevant practices (cf. *educativeness* as a component of a necessary experience in the present grounded theory).

The educative impact of the practical project course, where the students are able to internalize areas of project and SE work, is illustrated by the quote from the study by Smarkusky and Smith (2004). The student comment is written after project completion:

> *- I learned several new concepts in working on this project. I also gained a better understanding of several concepts taught to us within the IST curriculum. One of the benefits was a better understanding of the importance of user centered design and its use in project development. In addition, adhering to the concepts within the systems development life cycle allowed me to better structure my work on the site. All of these concepts were re-enforced through practical experience.*

Similar student reflection is found in the study by Rusu et al. (2009) study. Their project course introduced students with a local-remote team combination.

> *- I felt that I learned the software engineering theory more easily when I was able to apply it to a real project for a real customer. This allowed abstract ideas presented in the textbooks to become much easier to understand.*

Based on this kind of student comments, and the overall positiveness reported in Suri's paper, it seems that the educative impact of a practical project course is a strong component, and emphasizing this component alone is capable of making the value of realism endure if it adds to previously experienced realism.

Daniels and Asplund (1999) report on a one-semester industry-strength project course. The following extract is from their paper:

> *- The course is rewarding, since the change in most students during the course is both dramatic and positive. The route is certainly not smooth, but the feeling of accomplishment, even in the cases were the actual project "fails", is substantial, and students have certainly been given a chance to grow as computer science professionals in a way that, we believe, is lasting...*

Although it is difficult to interpret the literature without direct contact with the authors, this extract also raises a question in the context of the present theory. Strong growth and future orientation is here given while the "road is certainly not smooth". Whether and how much the statement arises from an analysis of student data is not given. Compared to Suri's report, the paper gives an impression of students' improved self-concept, and the difference in the course arrangements appears to be that this is a more intensive course with a frequent customer contact, real expectations, and one-semester timescale. In terms of the present theory, the feeling of a necessary experience due to an increase in self-confidence, again due to the experience of survival, is probably better enabled here.

A student quote from Zilora's (2004) paper perhaps explains a portion of the positive experience reported by Daniels and Asplund.

> *- I thought the project was great. I think it requires a lot more time than any of us are used to. I love "real world" projects. Even if you are only doing a small part, it gives you a better understanding of business processes and what the needs really are. Academic projects tend to be very small so you can do everything and get it done in the limited time available. In some cases it's good but sometimes it's kind of lacking real challenge.*

Here the student states that just getting something done is educative if done in a real world project, and one can compare this to the previous quote *...but the feeling of accomplishment, even in the cases were the actual project "fails", is substantial.* It might be that while a project may not achieve all its goals, Daniels and Asplund speak of "failure", compensation comes from partial survival and the possibility to internalize from practice in a realistic context.

It appears that Zilora's projects were of an on-going nature as were Suri's, but Zilora's paper reports on direct customer contact and that while the projects were on-going, the main target of a particular group was delimited (cf. a stand-alone project). Perhaps here also — as in Daniel's and Asplund's projects — the students are "closer" to the customer's expectations compared to Suri's lab in which one problem was the loss of student motivation due to slowly moving projects. In terms of the present theory, two viewpoints arise. Firstly, in Suri's slowly moving on-going projects, students may not be able to enjoy and experience survival from "completeness". Secondly, regarding the component of the attraction of realism[10] within the category of an enjoyable experience, responsibility towards customers and intensity of the course work appear to play an important role, as Suri's paper suggests that only having real world problems and infrequent customer collaboration is not a success.

Rover (2000) reports on a renewal of a computer engineering project course and includes many student quotes in her report. She reports that the students were influenced less positively in projects with external customers. The paper does not provide an explanation for this, but the impression given is that the customers have not been actively attending customers — the given student quotes do not mention a customer. The course is scheduled into class and lab sessions (cf. autonomous work expected from students). In line with the present theory, the students were satisfied with the course, and the quotes illustrate that the course was experienced as more enjoyable than the students' previous studies:

- The course was consistently more enjoyable than any course I've taken, on a day-to-day basis.
- It was different than other courses.

Interestingly, this is only a four-credit course compared for example, to the 10–15-credit `TIES405`. Despite this difference, many student comments in Rover's paper are in line with student comments on `TIES405`. For example, the Rover's students highlight "enjoyment from completeness":

- No one particular moment stands out, just a general sense of pride in what my team and I accomplished.

The importance of supportive group work (cf. the together-we-survive spirit in the present theory) is evident when such a sense of completeness is referred to:

- Going from the worst team ever to take the class to successful completion and presentation of the project.

Although the course is only a four-credit course and real customership is not emphasized, the students seem to experience survival. The paper does not explicate the course's position in the curriculum or how many hours students actually spend on the project, while the students' comments refer to a difficult high-workload course:

- It was a satisfying experience to see my group and other groups present a finished product after spending so many long hours in the lab together. That rewarding feeling is the best part of engineering. - It was one of the most

[10] Attraction of responsibility for one's own work and towards customers, attraction of intensity, and attraction of real world problems and associated technologies.

difficult and most enjoyable classes I've had at MSU. There were times when I hated it and times when I loved it.

One student's favorite memory is

-the long, late-night work sessions (and the chair races down the corridor!)

In terms of the present theory, the course is likely to be a heavy experience for students. This raises the issue that it is possible that the simple fact of surviving a difficult high-workload course (cf. heavy experience) creates a strong rewarding experience — whether actively attending real customers are involved or not. It might be that the positive experiences of inexperienced students, who are in need of progressing from a doubt to confidence — as formulated in the present theory — , are boosted by any substantial experience of survival on a study path. This importantly suggests that one should be cautious in concluding that an increase in students' self-confidence needs a realistic context and active customer collaboration, although these features seem to contribute to the experience of survival. Perhaps the important observation here is that Rover's course is a one-semester course and therefore students are likely to feel that they are working to a deadline, which may be a potential underlying explanation for reporting a heavy experience (cf. long lab hours) and the feeling of survival.

Hadfield and Jensen (2007) describe a two-semester capstone course sequence at the U.S Air Force Academy, which, lecture-like, consists of both class and lab hours. Their experiences derive from one offering in which five projects with real issues and customers were run. They report that the student end-of-course critiques were generally positive, with ratings consistently above department and institution averages. The authors state that from the students' point of view the course was regarded as both a positive and practical learning experience. It seems that practicality is enjoyed and favorable comparison made with other less enjoyable means of instruction. The paper gives an impression that the course was a challenge to inexperienced students, as it was also planned to really expose students to non-trivial problems — thus obviously more challenging for students than previous courses. Notice that disparity in team sizes was considered an equity problem by the students (cf. the problems in terms of unfairness in the present theory). Here, the practicality is associated with enjoyment, and the paper speaks of a generally positive outcome. This may indicate that practicality itself is also a strong component in creating a positive overall experience, if it is more than previously experienced.

The following extract found in Olsen's (2008) paper is interesting in the context of the present theory.

- Prior to the adoption of a real-world project, students taking Software Engineering frequently requested that their project be one for a real customer. In response to this request, an attempt was made several years ago to work with a customer that owned a small business. As the business was new, the owner had a real need for inexpensive software and was interested in participating in a project. There were several problems with this project. Most of the

> *problems were due to the difficulties of working with a business that had little time to work with students. The customer underestimated the time that was required on their part and expected students to take a minimal set of requirements and produce a complete product. Software was produced but, to our knowledge, it was never used as it only delivered some of the desired features. There were some benefits for the students. In spite of their frustrations, they reported that overall it was a good learning experience.*

She reports on the students' need for realism, problems encountered and the associated frustration, and students' overall positive learning experience. This description of a specific case quite closely matches the mechanism put forward in the present theory: inexperienced students who need realism tolerate problems, and overall experiences tend to turn positive despite the problems. The students' need for realism can readily be identified in the literature; for example, in Poger and Bailie's (2006) paper:

> *- Exit interviews with students revealed their desire to have experience with "real" problems as opposed to examples often found in textbooks.*

The need is also stated by project teachers in the literature (Tan and Jones, 2008):

> *- Students need real-world team project experience before they embark on their professional career in computing.*

A relevant reference paper is that by James (2005), on lessons learned from using an external sponsor. Relevance here means that the paper gives an impression of a degree of realism that corresponds to that in `TIES405`. Customers' expectations of the projects are serious. In this paper the basic problem is the open-endedness of real projects and ensuing difficulty to terminate the projects on time[11]. Regarding the course benefits, the authors state the following:

> *- Although these demands [refers to unpredictable workload] on the faculty are non-trivial, the positive learning outcomes resulting from participating on a sponsored project more than offset the burdens.*

There appears to be a problem of delayed projects so that the students have to continue their projects in the form of another course. The positive learning outcome is nevertheless highlighted, as it is by `TIES405` students. Among the benefits, James speaks of students' internalization of real world project experience and developing a closer relationship with their professors and each other. Interestingly, the latter is not included in the present theory derived from the `TIES405` context. In the course reported by James, students and faculty do projects together, which is not the case in `TIES405`, where teachers act from a teaching/supervising position. This may explain the difference, while it may also be due to a variety of lenses possible for a study and the limitations of the different data sources used in a study.

[11] The second results chapter of the dissertation addresses this problem.

Does overall positiveness always emerge?

The above literature references give an impression of overall positive course experiences while the positive impact of a project course is not always quite so obvious. Goulding and DiTrolio (2005) report on a project course experiment using a small 5-6 person team and realistic constraints (complex technical environment and realistic emphases such as priority on quality, cost & time, etc.). In such a small-team-size course, the authors note that students who are technically less skillful can nevertheless feel a sense of accomplishment with their contributions to the project. The same "alleviation" was included in the present theory in relation to the discouragement due to not being able to meet personal learning goals. Overall, Goulding and DiTrolio were not satisfied with the result of using realistic constraints. Here, the real customer interface is missing, the project topic is a pre-packaged boardgame, and the same topic is used for all the project groups. The realism provided here is artificially set up; it is about constraints according to which students are expected to prioritize their work (e.g. quality, costs). The realism provided is simulated realism and overall positiveness is not reported.

Poger and Bailie (2006) report on the experience of introducing a real world problem into a two-course capstone sequence. The students developed an assessment system for the university. Real customership appears to be less emphasized compared to having a real problem, and it appears that the projects are not stand-alone projects but can continue with new students (cf. feeling of success and survival from completion in the present theory and literature references). The authors mention that the course was well received by the students, if not overwhelmingly. Reading the paper discloses that the students who already had group work experience did not experience as much different the "real" project work. Two of the nine students did not feel any real benefit and did not recommend the course for others. Although it is difficult to form a reliable conclusion solely on the basis of the text, there is an indication that the realism provided needs to make a difference for students to boost their positive experiences.

Run-time and future oriented modes

The literature repeatedly notes that the students both enjoy and gain from project courses. In Clark's (2005) paper the conclusion begins with the statement that there are many reasons for incorporating real, unique industry projects: increased student motivation and confidence. Newman et al. (2003) conclude in a similar manner, by stating that the effectiveness of open-ended group projects was partially due to improvements in student confidence and motivation. Melin and Cornholm (2004) report on first-year project-oriented work in which the students should learn how to design and implement information systems. They report on overall positiveness because an authentic course prepares students for working life and is fun. These qualities correspond to future-oriented and run-time dimensions in students' positive experiences. Conceptualizing the description repeatedly found in the literature to these two modes helps to explain the positive effect of a project course and also what features (the run-time and future oriented mode) are in operation when inexperienced students' overall experience turns positive.

Trade-offs

The literature includes statements that explicitly describe trade-offs similar to those that were found in `TIES405`. Daniel and Asplund (1999) — cited earlier — report that *the team pressure into working with the project is in many cases a positive force, but it can turn into a problem if it gets too big*. Here the course context is a one-semester real customer course, yet not as intensive as `TIES405`. Melin and Cronholm (2004) report how several of the discovered benefits of their first-year project-oriented course can also be problems. They say that some of the conditions represented two sides of the same coin. Lowe (2000) states that the unknown is both a threat and an opportunity when he speaks of how students are initially concerned at the breadth of the work to be undertaken. His course is not a real customer course. The grounded theory of this chapter would stress that a project course can and should be characterized in terms of trade-offs. It appears that group work itself introduces trade-offs, and more such tensions come along by making a course context more realistic by using real topics and customers.

Support

Students' dependence on support is also evident in the literature. For example, Laware and Walters (2004) associate a project course with problem-based and problem-centered learning, and conclude that a course project can be extremely effective if students are provided with the right degree of guidance and mentoring. The factor shared with `TIES405` is that the course employed real customers and thus real customer expectations. Another example, a student report by Davis (2000), foregrounds how students' project work is conditioned by support from the customer.

> *- Most importantly, however, each customer will be required to commit time and resources to the project so that a team is not left in the cold developing for a phantom customer. This is far and away the best improvement that we see in the course. It helps ensure that future groups will have the support they need, and a specified amount of time from the customer.*

Inexperienced students thus require support in a variety of areas during the project course work but this is particularly important at least in realistic course contexts which are not pre-packaged and predictable. The present theory proposes that one of the contributors to increased confidence is having sufficient challenge to allow experience of survival, but on the condition that the amount of challenge is reasonable. More practically, challenge is conditioned by support, and this may arise at least in part from the course arrangements. While the project topics are carefully negotiated (see Section 2.4), it is my direct observation, in line with

James (2005), that a stationary target in projects with real customers and topics is not possible (cf. the note in Fincher's et al. taxonomy regarding a project with a real client in Chapter 2). Real topics cannot be pre-packaged, and this is why support has an essential role in addition to the negotiations before project commencement.

Pedagogical concepts

Finally, one could borrow concepts from the field of education, and reflect on the present theory, or parts of it, from that viewpoint. An example of a potentially relevant educational concept is self-direction. In terms of the present theory, if one considers the component of "occupational thought process reinforced" given under the category of necessary experience, the concept of self-direction appears to be somewhat analogous. According to Candy (1991), self-direction is about the learner's ability to critically evaluate and make decisions in a particular domain. Other evident reference concepts are problem-based learning, zone of proximal development (Vygotsky, 1978), and scaffolding Bruner (1977). Comparison with or potential transfer to educational concepts is not within the scope of this dissertation but acknowledged as a future direction for research.

4.3 Summary

The grounded theory in Section 4.1 revealed a mechanism explaining why a positive overall course experience arises despite the problems are easily encountered in a realistic project course context. More specifically, the boosting of positive course experiences due to students' inexperience was identified as the factor explaining overall positiveness. This mechanism can be shortly characterized as "enduring value of realism — boosted by inexperience". It appeared, also by continuing the comparison with the literature, that feeding students with a more practical course context and/or more realistic arrangements compared to their previous experiences contributes to a positive overall experience.

Interesting additions (hypotheses) arose from the literature. For a positive overall project course experience, it may be sufficient that only some of the components given under the categories of an enjoyable and a necessary experience are emphasized, if these add to previously experienced realism. For example, using real world problems and relevant practices alone can make overall experiences positive, as they are educative through allowing internalization. Perhaps the most interesting hypothesis is that simply completing a difficult course can create a feeling of survival and increased confidence without any particular emphasis on a realistic course context and collaboration with real customers. I mentioned within the present theory that there has to be a genuine challenge to experience survival, but did not really consider such a challenge independently of real customer course context, that is, as a viewpoint in its own right.

The curricular position of a project course appeared to have a key role in the analysis, as it was associated with the students' inexperience. According to Clear et al. (2001) considering a course's position in the curriculum is important regarding the setting of its goals. They also find giving students more complexity than they have previously experienced to be one alternative when stipulating project course characteristics. The present theory integrates these two viewpoints into a mechanism that explains the overall positive student experience gained from a realistic project course.

The related work referred to in Section 4.2 represents several curricula levels and areas of computing. On the basis of this observation, the mechanism the present theory puts forward, **enduring value of realism — boosted by inexperience**, might be a relevant viewpoint in any curriculum where realistic learning contexts are included.

4.4 Methodological reflection

Theoretical sensitivity

Piantanida et al. (2004) conclude that the validity of a grounded theory study arises from theoretical sensitivity — meaning qualities like concentration, patience, and ability to conceptualize — as opposed to strategic research actions. It is suggested here that this can be roughly evaluated by observing the level of abstraction, the associated amount of reduction, and the unity of the results. In other words, this can be observed in the completeness of a grounded theory study. If the theory is explicated around an emergent and analytic core category, thus furthered from what is still an implicit stage of theorizing, it is likely that theoretical sensitivity is in place. The grounded theory presented in this chapter is given in the context of a single core.

Relevance to those involved

Laymen can face familiar situations more purposefully with a relevant substantive grounded theory (Glaser, 1978, p. 14). Thus, the important implication, including here, with the present theory, is that when knowing the students' tolerance of problems and the tendency to positive reports when fed with more realism than previously, educators can be better aware of, and more careful in analyzing, the nature of the problems students encounter. The present theory thus underlines that it must be clarified whether the difficulties encountered by students are challenges that carry educational value or problems that should be resolved to improve education. For example, Coppit and Haddox-Schatz (2005) have devised large team projects and noticed that students suggested the use of smaller teams to ease management overhead, but without considering the educational goal that was to be accomplished by using large teams. Here the use of large teams is not likely to be a problem. However, the lesson to be drawn from the present theory is that the problems encountered may be severe but not paid attention to due to an overall positive learning outcome which arises not only from the course arrangements but also from students' inexperience.

Thus, the present theory provides a viewpoint that helps to objectively interpret the impact of a project course and it is argued as relevant not only for teachers and students, but also for teachers-as-researchers. This is similar to how Glaser and Strauss (1964) see relevance in their early work on how nurses experience social loss when a patient is dying. Their study reveals the underlying causes of such a loss experience, i.e., how different social characteristics have an effect on the nurses' loss experience and what kind of treatment is received by a patient. Glaser and Strauss state that the study results help nurses to function in their daily work so that they are able to treat dying patients more objectively. The conclusion drawn here is that grounded theory has an emancipatory character.

Was it induction?

Suddaby (2006) states that a fundamentalist grounded theory view on either induction or deduction is incorrect, and by referring to Peirce's conception on abduction[12] he arrives at saying that a grounded theory analysis anyway involves some deduction, as suggested by Strauss and Corbin (1990). To my mind, the difference between the two approaches (Glaser vs. Strauss and Corbin) must be clarified, and I explain it to myself by noting that Strauss and Corbin posit deduction as a conscious strategic act at the methodological level — or so it can easily be interpreted: "*While coding, we might see an incident of pain management and propose that conditions of intense pain will be followed by measures taken to relieve pain. We then check each incident of intense pain that we come across in the data to determine if this is so, verifying inductively what we proposed deductively*" (Strauss and Corbin, 1990, p. 111), whereas Glaser speaks of making sense of data. Glaser (1978, pp. 132, 135) states that an emergent theory is likely not to be correct at first and needs to be re-formulated over and over again. In my view, such re-formulation emphasizes the thought process needed to arrive at a grounded emergent theory instead of deduction as a conscious strategic part of the method. My conception is that I've followed Glaser's view — induction in the sense of the GT literature.

[12] According to Peirce (1998, pp. 106-107) abduction is a process where consideration of facts suggests hypotheses, and no new truth can arise from induction or deduction, but abduction, and abduction is, after all, nothing but guessing. This may actually resemble what Glaser calls induction at a methodological level, and one must therefore be careful with the lens used when reading the GT literature. With *methodological level* I refer to questions that characterize a research paradigm, methodological level being about means by which knowledge can be obtained, as opposed to epistemological one which is about the relationship between a knower and to-be-known; see Guba and Lincoln (1994).

Role of the researcher

Green (2003) states, in his description of developing a capstone project course, that one stress factor for students that needed to be improved was the lack of a software life-cycle. The students had previously worked as programmers on an instructor-defined problem and design. When I read the paper I first considered that perhaps the major solution is not coverage of the software life-cycle, but allowing the students to create ownership of the project topic. This is just a small example of how reading data is affected by the sensitives of the researcher. With respect to grounded theory, the researcher's sensitivities are inevitable and a benefit in getting coding started in the first place. However, sensitivities are not to be taken as a preconception or strategy that would define a specific research question and then limit the study to a verification of that question.

It has been argued that a researcher needs to explicate his/her position in a GT process (Suddaby, 2006). The sensitivities I identify in conducting the study presented in this chapter are my basic interest in whether this kind of one-semester real customer project model would be applicable in the computing curriculum, and if so, why it is like that. I knew that some authors in the field recommend having a full software life-cycle including maintenance, and I knew from experience that this is not usually the case in a one-semester course with real-life project topics. I also knew that realistic course contexts have been argued as problematic (Aygün, 2004). I went on ("got involved") to study the impact of the course, and the end result is not a statement for or against this kind of project model, but hopefully helps in making objective considerations on the impact of such a course.

There is an on-going debate on the foundations of grounded theory. Annells (1996) has posited an association between classical grounded theory (DGT and Glaser) and critical realism (Bhaskar, 1978). Critical realism holds a stratified ontological view, e.g. (Outhwaite, 1998, p. 282). The results from the two (full) GT processes I have conducted so far ((Isomöttönen and Kärkkäinen, 2009) and this chapter) tend to reminiscent of the "explanatory mechanisms" found in such a stratified ontological view. Urquhart (2002) notes that while GT can be placed into the domain of post-positivism (cf. Lincoln and Cuba (1994)), it still tries to be an inductive method, which is not in line with the deduction and verification of the post-positivist affair. From this viewpoint, GT appears to be a multi-paradigm method, and accordingly, there are different views on the amount of subjectivity/objectivity involved. Interestingly, Urquhart proposes that the amount of subjectivity or objectivity might arise from the philosophical position of the GT researcher instead of the method itself.

I feel at home with the association between GT and critical realism, applying DGT and Glaser's approach, and being in the role of an observer, the *mindset* I consider is naturally given by the conceptualization process when using constant comparison [read: conceptualization from similarities and differences]. I personally feel at home emphasizing an objective mindset with GT, its interest in finding latent patterns (cf. generative mechanisms), more than I wish to describe it as strongly interpretivist. In other words, I recognize that a conceptualization process is inevitably about interpretation and the selection of viewpoints, but for me, however, the interest in finding causes for what constructs the observable takes precedence over the interpretivist mindset. I recognize that the objective mindset to which I refer here is not about objectivity in terms of controllability and repeatability: the research process with (classical) GT is very implicit.

Systematicity

I ran into a problem of feeling that I was leaving out too much, particularly when I tried to understand the components in the experience categories given in Figure 13. I felt I had to force reduction. This may be due to what Glaser (1978, p. 58) warns about: if you tend to conceptualize by (just) refining a holistic overview, you might lose a reliable insight gained from continuous systematic comparison. Or, reduction simply is very difficult because of the possibility of addressing discoveries from so many viewpoints. A good example of this is the components under the category of an enjoyable experience. They are of different level of abstraction (e.g. group work vs. challenge) and I still do not know whether the "managing" of the concepts in this particular way was a good idea. Anyhow, I cannot and even do not feel a need to claim that the underlying components of the experience categories are all-inclusive, but something that stood out from the data and was influenced by the sensitivities of the researcher. This type of research was perceived as an implicit process in which I suddenly arrived at a discovery I saw as explanatory (how the students' need for realism, arising from their inexperience, provided a cause — not easily discoverable — for their overall positiveness), and it is difficult to explicate in retrospect between the "keep comparing in a systematic way" vs. "refine a holistic overview" modes.

Literature as data

I continued with the comparison using the literature as data. The papers that gave an impression of overall positive student experiences were compared and underlying causes were discussed. Similarly, two papers were included in which the overall positiveness was not quite so obvious. These were compared and some underlying causes were indicated (simulating realism and students' previous experience of group work).

The use of the literature as a data source is acceptable in GT which is essentially a comparison process, and as such, does not constrain data sources: in seeking to uncover latent patterns, anything is data (Glaser, 2002). The project course studies in the field often describe course arrangements and conclude with experiences, typically with teachers' experiences accompanied with student quotes. There are lots of reported experiences that are in line with each other but this data is not systematized and reduced to easy-to-remember mechanisms of the studied object (into theories). From this perspective, the literature in this descriptive form appears to be promising for generating grounded theory. In Section 4.2, I regarded the course project descriptions found in the literature as close-enough contexts with `TIES405` to be incorporated into a discussional continuation of the local theory. Here, contextual factors (differences between the course contexts) became a part of the comparison process. For example, I considered how the *intensity* of a project course might have played a role in students' perceptions.

On the other hand, it is difficult to know how the content of a paper is selected and from what kind of data sources it makes its statements. As the comparison with the literature in Section 4.2 suggests, nearly all aspects of the present theory are separately *described* in the literature, but I doubt that I could have arrived at the theory (the mechanism: enduring value of realism — boosted by inexperience) directly from the literature. The contents of the related work most probably focused only on some issues, or perspectives, not least because of page limits, and thus the interpretation in Section 4.2 has a speculative nature.

Generalizability

As discussed in Introduction, theory in this dissertation refers to middle-ground theories as opposed to grand generalizations. In terms of grounded theory, I see the present theory as a substantive theory, having realistic project courses in computer science project work as the substantive area of study (see Section 3.1.5).

I see the mechanism foregrounded in the present theory (enduring value of realism — boosted by inexperience) as a potential start-up for developing a more formal theory. That is to say, one that would not anymore be anchored to one specific area of study. For example, further comparison could start by analyzing realism in education implemented either in the form of few capstones or as a gradual inclusion throughout a curriculum — in relation to students' positive/negative feedback. This could lead to a study without a connection to one specific content area (e.g. computer science project work).

In connection to generalizability, it should be noted here that the reflective comparison between the local theory and the literature in Section 4.2 does not mean forcing a generalization, but is about advancing the local theory in a hypothetical manner.

5 HOW TO MANAGE A ONE-SEMESTER REAL CUSTOMER STUDENT PROJECT COURSE

This is the latter of the two result chapters of the dissertation and relates to the course development work and the associated action research as introduced in the Introduction and Chapter 3. The reader is assumed to be familiar with the context of the action research — `TIES405` project course — as described in Chapter 2. The starting situation, the main problem, the objective, and the strategy of the action research are given in Chapter 3. The objective of the action research is here restated as an introduction to the results:

> The objective of the action research was **to find an operational approach that enables the production of software for project customers in the fixed one-semester timescale not forgetting student learning.**

The chapter starts with an outline of the theory that was developed through the present action research, titled "operational foundation", in Section 5.1. This states the basic operational idea of how to manage the course with the constraints of real customer expectations and the fixed timescale, together with the educational responsibility to address student learning. The operational foundation is based on understanding the nature of the applicable operational knowledge and what the epistemological positions of project stakeholders are in terms of this operational knowledge. *Operational knowledge* means here how to develop software in the studied educational context: it is understanding what processes, practices, roles, etc., should be in place with this course model, i.e., with the fixed timescale and real customer expectations.

After the outline section, the key elements of the present theory will be revisited in more detail: While the outline of the theory (Section 5.1) focuses on the nature of operational knowledge, Section 5.2 provides a conceptualization of operational knowledge in terms of a meta-process. Section 5.3 again explicates the relationship between the teacher and the students in the present theory regarding the contentual focus and interaction in teaching.

Section 5.4 discusses the main themes of the present theory in the context of the literature. The contents of the chapter are summarized in Section 5.5. Method-

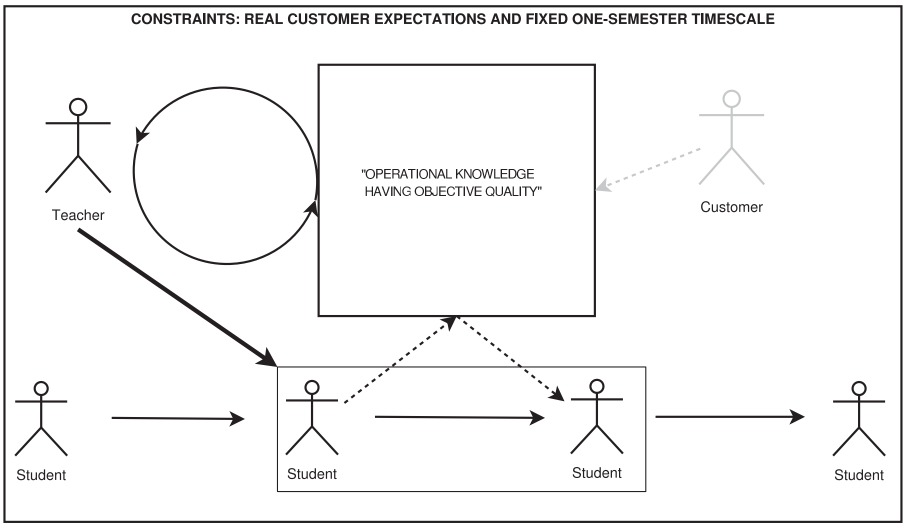
CONSTRAINTS: REAL CUSTOMER EXPECTATIONS AND FIXED ONE-SEMESTER TIMESCALE
"OPERATIONAL KNOWLEDGE HAVING OBJECTIVE QUALITY"
Teacher
Customer
Student
Student
Student
Student

FIGURE 16 An operational foundation for a one-semester (fixed time) real customer project course

ological considerations are foregrounded in Section 5.6, which will, as in the first results chapter, serve as a discussion on validity and reliability.

5.1 Operational foundation

The model in Figure 16 states the operational approach discovered with the stated action research objective. The model provides a basic operational idea of a holistic nature for the course, and is therefore analytically called *operational foundation*. It is outlined as follows:

Operational knowledge with a one-semester real customer course context needs to be derived from the teaching context itself and teachers are in the best position to foreground this knowledge by means of reflection. This knowledge has an objective quality as in real situations operational knowledge is not anybody's claim that something works. Teachers must share this knowledge with students to enhance student learning. The learning is thus addressed by considering the relevancies in operation together with the students.

The components of the model and how the components interrelate are explained below.

5.1.1 Stakeholders and operational knowledge

Teachers are in a specific position in the model. Projects following one another enable teachers to compare between projects and abstract out causalities. The teachers are able to see the consequences of the actions taken, both their own and those of the other stakeholders'. They have a continuous opportunity for reflection in one and the same course context. For this to be realized, they need the will to reflect.

Students, on the other hand, enter the project course at the beginning of a semester and leave after their project is finished. If they don't have earlier software project experiences — which they do not usually have — a comparison between projects, which would aid their reflection, is not possible. The setting is hence quite different for teachers and students in this respect.

Customers also affect the operational knowledge, but in this theory their role is viewed more as a constraint on the action research and the associated theory formation: the whole operational foundation given in Figure 16 was derived from actions which made the course a feasible educational arrangement regarding the constraints of real customer expectations and the fixed one-semester timescale.

Operational knowledge must be derived from the course context. In order to characterize such context-dependent operational knowledge, the term *objective knowledge* is employed. The term is sourced from Popper (1979) who contemplated an objective quality of knowledge, a certain kind of epistemology without a knowing subject[1]. It means the knowledge in a thought that is different from the "I think" mode. Accordingly, there is knowledge outside our subjective way of thinking. Popper uses the term "third world" for the objective quality of knowledge and "second world" for the subjective one. The following example illustrates the difference between the second world and the third world knowledge (Popper, 1979, p. 110):

> Second world (subjective knowledge): *"I know that Fermat's last theorem has not been proved, but I believe it will be proved one day"*
> Third world (objective knowledge): *"Taking account of the present state of metamathematical knowledge, it seems possible that Fermat's last theorem may be undecidable."*

Two further examples are included from Popper (1979, pp. 107–108), which he gives as thought experiments illustrating the position of third-world knowledge:

> Experiment 1: *All our machines and tools are destroyed, and all our subjective learning, including our subjective knowledge of machines and tools, and how to use them. But libraries and our capacity to learn from them survive. Clearly, after much suffering, our world may get going again.*
> Experiment 2: *As before, machines and tools are destroyed, and subjective*

[1] Notice clearly that the references to Popper are only a means to communicate the findings. They are analogies within the results as opposed to the delivery of any ontological or epistemological claims in a general sense.

learning, including our subjective knowledge of machines and tools, and how to use them. But this time, all the libraries are destroyed also, so that our capacity to learn from books becomes useless.

Popper then notes that in the latter case (Experiment 2) there would be no re-emergence of our civilization for many millennia. With these examples he illustrates the independent existence of the third world.

Now, in `TIES405`, customers, project groups, problem domains, and software domains change when projects end and the next ones get started. The course context is thus unstable in many ways, which means that the applicable operation cannot be totally stable, fixed. Project instances shape pattern-like knowledge of what will work in given circumstances, and what will not. Such knowledge is of an objective quality, being shaped by itself in the course context. While the teachers are in the best position to interact with this knowledge, the project stakeholders — primarily the teacher, the students, and the customer[2] — all shape that knowledge through interaction.

5.1.2 Acquiring and delivering the knowledge

The project stakeholders attempt to apply the solutions they currently know, and can come up with, to the problems they encounter, and new problems to be solved arise[3]. In this process operational knowledge is being shaped. While this knowledge becomes documented into project deliverables (e.g. the student course experiences in project reports), the teachers can efficiently deliver the knowledge to subsequent projects. The teachers' reflection is necessary and the resulting knowledge needs to be shared with the other stakeholders, particularly with the students, to provide content for the students' learning. This is the basic idea of how operational knowledge is acquired and delivered (see the relations between the teacher and operational knowledge and the teacher and the students in Figure 16; an example is given in Section 5.1.4).

With the intensive one-semester course model, all of the student effort easily goes into survival from the tasks and adopting at least a somehow disciplined manner of working. This was indicated by the analysis conducted for Chapter 4: it was noticed that students tend to focus their feedback on those issues/tasks that have been close to their part in the project. If the teachers do not intentionally share considerations on applicable operation (e.g. applicable software process and engineering practices) with the students, the students' learning potential is not properly used. Notice that the situation is likely to differ, for example, if a project course is devised to teach predefined engineering practices instead of survival from real customer expectations. In this case, the teacher could rely on the definitions of the practices, which themselves would then be the main content of the learning.

[2] End users attend the projects occasionally and the department's computer support is available for the students.

[3] Perhaps this can take an analogy from Popper's (1979, p. 144) idea that error elimination of tentative theories on a problem tends to create new problems.

What is then the role of industry practice? The teachers' reflection, referred to above, is much about finding an applicable form for the acknowledged contents of software engineering, see (IEEE, 1990, software process definition). A particular attention must be given to contextual factors here, i.e., the real situation and the results expected from these kinds of student projects. Because of these factors, external operational knowledge (e.g. a software process definition sourced from the literature) cannot be adopted without justifying how it fits the context. External conceptions (industry practice) serve as an important practice pool, but justificatory reflection is still needed. This is very important, because, for example, a customer company may suggest use of a particular software process which is entirely over-sized for the course context. Adopting such a process would burden the real progress of the project.

With the operational approach presented here, there is no hindrance to the university in redefining and innovating practices in projects because the students' learning is anchored to shared reflection instead of learning particular prescribed industry practices.

5.1.3 Characterizing knowledge acquisition: reflection with criticism

The analogy to the Popper's third world provides an apt characterization for the nature of teaching knowledge with a one-semester real customer project model. Objective knowledge needs to be interacted with in order to avoid the severe problem of a delayed project. From this perspective, the one-semester real customer model which focuses on both the product (working software) and the process is a frightening teaching context because projects should make progress and the teachers do not know for sure the solutions to problematic situations. Using Popper's words again: *knowledge in this objective sense is totally independent of anyone's claim to know* (Popper, 1979, p. 109). The teachers' reflection must involve critical on-going analysis and conceptualization. *They are at work to improve the objective thought contents by way of criticism* (Popper, 1978, p. 156).

This may appear to make enormous demands on teaching ability regarding how to develop software. It is to be understood here that teachers are not seeking the truth, but consciously attempting to critically encounter new projects and analyze what is applicable operation in them. The challenge is not only in the ability but the attitude of the teachers. The teachers need to be able to change their prevailing conceptions and thus admit the nature of operational knowledge that differs from the "I think" mode.

5.1.4 Example

In the following, an example is given to demonstrate the conceptual idea of the operational foundation in practice, as, according to Kemmis (2001), practical action research often includes self-reflective reports involving story telling. The example relates to the challenge of including some system testing in the tight one-semester timescale. The challenge of getting students to test software has been reported in the literature. For example, Lowe's (2000) strategy was to introduce a fake project completion date to students and then dedicate the time after the faked date solely to testing. The answer arising from this study indicates accepting the planned time constraints, then considering in what form system testing can be conducted in the given project context, thus casting aside the notion that one needs to use a certain prescribed practice, and finally focusing more on the critical evaluation of applicable operation.

I have calculated the number of defects found when a student team prepares a test plan that includes a large set of test cases for system testing, and when a customer uses the software after each short iteration. Testing of the first kind resulted in almost zero value (0–2 defects, several test rounds) compared to the customer using the software frequently (8 defects). An inexperienced developer preparing test cases most likely follows the same cognitive paths that he or she utilized in the requirements specification phase and ad hoc usage of the software during the development, so that test cases tend to be just a repetition of one and the same intuition — as Brooks (1995, p. 142) refers to a statement by Vyssotsky regarding the testing of a specification: *They [developers] won't tell you they don't understand it; they will happily invent their way through the gaps and obscurities.*

Utilizing a customer or students from other project groups to test software seems to show up more defects. The most applicable way of testing is in any case a project-specific issue, it is of situational importance. The real customer projects at JYU/MIT vary in their nature (e.g. from scratch vs. enhancements and maintenance) which also has an effect on what kind of system testing is reasonable with the existing time constraints[4]. The reason for undertaking a certain kind of testing must be explained to the students, and preferably, an applicable way of testing is figured out together with the students. This allows for students' learning when learning about a particular prescribed system testing is not possible with the given constraints.

The example demonstrates on-going process tailoring which is built upon teacher reflection, but also necessitates student participation to focus not only on efficacy but learning (cf. the objective of the action research highlighted at the beginning of the chapter).

5.2 Operational knowledge in terms of a meta-process

While it was stated in Section 5.1 that operational knowledge is based on ongoing reflection on the stakeholders' endeavors in the course, it is obvious that in terms of software process this means some kind of process tailoring. This does not, however, guide a practitioner regarding how to construct an applicable process. Section 5.2.1 presents a meta-process on how to construct an applicable process, further refining the understanding of operational knowledge, as in Fig-

[4] Pfleeger and Atlee (2006, p. 372) identify function tests, performance tests, acceptance tests, and installation tests belonging to system testing.

ure 16. Sections 5.2.2 and 5.2.3 communicate this meta-process in the context of the studied course.

Here, some discussion is needed regarding the terminology on how to develop software. The IEEE (1990) definition already mentioned in Section 5.1.2 says that a software process is a process by which user needs are translated into a software product, including translating the user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operation use. Humphrey (1995) describes the software process similarly, simply as the sequence of steps required to develop or maintain software. Additionally Humphrey speaks of a software process definition as a description of this process. According to Humphrey's view, the software process sets out the technical and management framework for applying methods, tools, and people to the software tasks, while the process definition identifies roles and specifies tasks. Sommerville (2007, pp. 8-9) uses the term software process in a sense close to the IEEE's definition. He covers the software life-cycle with four process activities: software specification, software development, software validation, and software evolution. He additionally uses the term software process model as a simplified description of a particular domain of the software process (e.g. a role/action model) and identifies three generic process models (paradigms): the waterfall, iterative development, and component-based software engineering. Pressman (1994) does not use the term software process but also speaks of paradigms and generic phases. It seems that the widely adopted terms are not strictly standardized between textbooks. In the following I aim to avoid confusion between the different nuances and use only the term *software process* or, more briefly, *process* in speaking about how to develop software. In the educational context this also includes aspects with a mainly educational rationale.

5.2.1 The meta-process

By the end of the action research phase of major changes (see Section 3.6), I discovered that an emphasis on communication, teaching as coaching, and short iterations fitted the course. After using this initial framework for a while, I *realized* that the concept of operational knowledge could be explicated in a more communicative form. I could speak of a meta-process. Figure 17 depicts the meta-process, which is explained as follows. I *realized* that since we had talked about context-driven roles, goals, and practices (Isomöttönen et al., 2007), the context could be understood as consisting of two parts: a stable part and an unstable part. The stable part comprises those characteristics of the development context that occur often, e.g., student inexperience. The unstable part is something which is difficult to predict and changes often, e.g., customer type (how technically oriented the customer is etc.). The software process is first based on the stable part of the context, which creates a stable part of the process. This provides a loose process skeleton assumed to fit the context, thus helping students to produce a reasonable outcome in the fixed one-semester timescale. The skeleton

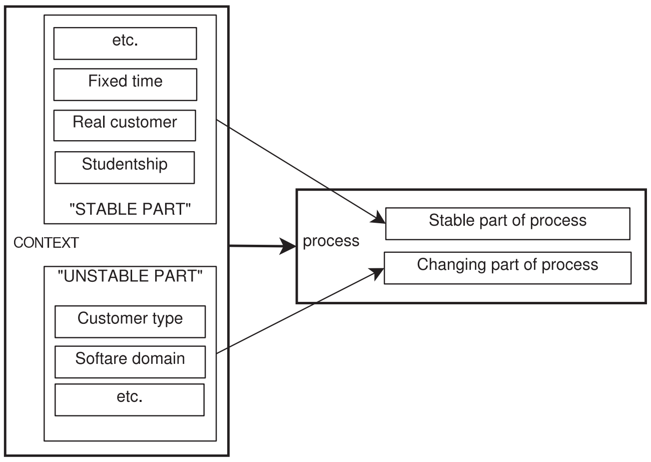

FIGURE 17 Meta-process

is complemented on the basis of the unstable part of the context, from a situational perspective. For example, the project type, customer type and customers' preferences for certain low-level practices necessitate process tailoring. This creates a part of the process susceptible to change. A teacher is in a key position to communicate the skeleton to the other stakeholders, while situational factors are added to the skeleton through the collaboration between the stakeholders of a single project.

Rolland (1998) states that *a "process meta-model" is a description at the type level of a process model. A process model is, thus, an instantiation of a process meta-model. Obviously, a meta-model can be instantiated several times in order to define various process models.* This is the meaning in which I speak of the meta-process. The meta-process in Figure 17 guides how the applicable software process is constructed in the course context but does not describe any particular process. For example, I do not speak of a particular agile process even though I have found issues emphasized in the agile manifesto (Beck et al., 2001) very valuable, the use of short iterations being a concrete example of the practice sourced from the agile pool. This would not reveal the idea of understanding the applicable process here, as given in Figure 17. Instead of listing practices found useful, a greater value comes from theorizing what the process is composed of, as it also becomes an objective in deciding what a software process should be composed of. In short, the conceptualization becomes a practical tool for a teacher.

The loose process skeleton that fits the context means having the right amount of discipline, still leaving space for the empowerment of the stakeholders. This is particularly important for enhancing the students' sense of the ownership of their project work. It is also important as one cannot assume that an external customer of a student project would start to act as the university stakeholders dictate, that is, with a certain process prescription in mind. From the customers' perspective, the whole point of using the skeleton, discovered applicable in the sense that the student group is able to develop a reasonable output for the customer in the given time, means taking account of the customer.

In Section 5.1 it was noted that teaching that interacts with objective knowledge is challenging. The loose process skeleton stated here with the meta-process mitigates this challenge. With the skeleton in mind, teachers do not start from scratch but have an operational skeleton they can trust and which is based on their reflection. While the skeleton presents elements in the software process that are assumed to "always" fit, the skeleton must be observed to fit and the attitude of the teacher must allow for a change of skeleton if necessary.

5.2.2 Stable part of the meta-process

At JYU/MIT, as a result of the present action research efforts, the stable part of the software process consisted of the use of short iterations, with the emphasis on face to face communication, and teaching as coaching. These are reviewed in the following.

Short iterations

Over the course of time, the process model adopted in `TIES405` usually come to be a single-pass waterfall sequence or two to three large development increments. The trend is found by analyzing the handouts that have been given to students at the beginning of the course. One reason for the use of the waterfall among the project teachers has been that the single-pass waterfall sequence is the easiest for inexperienced students to perceive and start work with. Under this waterfall setting, the course has been phase-driven so that the documentation associated with the project phases (project planning, analysis, design, etc...) has provided milestones and set the overall rhythm for students' work.

On reflection, I first considered two projects that made excellent progress. One used a single-pass waterfall sequence (during my first supervision term spring 2005), and the other a set-up phase with two ensuing large increments for actual product development (when moving towards the use of short iterations, autumn 2005). This was how the groups were guided to conduct the work, but was not what the groups actually did in practice. While they delivered documentation consistent with the waterfall phases and the use of two large increments, in practice they proceeded by prototyping in short iterations and carefully managing and reacting to the customer's feedback. They used short iterations and managed requirements carefully, iteration by iteration, while we teachers provided them with another track which was too much an irrelevant external process prescription. A lot of the teaching effort was put into reviewing the documentation delivered by the students. For example, the students of the waterfall project were expected to deliver an extensive design plan, but this deliverable was late. The students lost the motivation to complete the plan since the documentation at that point of the one-semester timescale was not effective but introduced irrelevant work. The design was already in place and had been reviewed by the technical supervisor assigned to the project. Thus, writing a design plan became a simu-

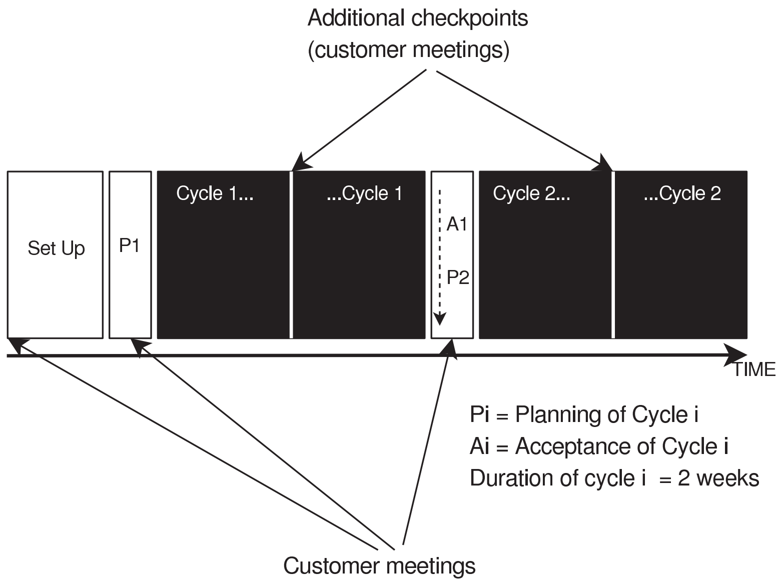

FIGURE 18 A reference for the use of short iterations

lation of realism: "This must now be completed because this is how it is done in real projects". On reflection, the success of the groups appeared to be based on requirements management and implementation in short iterations, and importantly, autonomous communication with the customer as an integral part of the iterative development.

I also considered other two projects in which three increments were used for implementation after a project set-up phase (when moving towards short iterations, autumn 2005 and spring 2006). This was not much different from a single-pass waterfall sequence. Using large increments did not bring home the focus and the transparency the above two projects had demonstrated. By comparison I thus concluded that short iterations work well and are worth going into more explicitly in the course. I then started to use one-week iterations in the project set-up phase and two-week iterations thereafter. This seemed to have a positive effect. It changed the nature of the work. The first such project was carried out in autumn 2006. The use of short iterations now meant that project tasks are planned at the beginning of each iteration and reviewed with the customer at the end of the iteration. Figure 18 depicts the handout given to the student groups after the pilot project as a simple illustration of working in short iterations. Along with the subsequent projects, I identified a number of reasons why short iterations constitute a key practice in the process suitable for the one-semester real customer model.

- They decrease complexity and confusion by giving *focus*. One of the most frequently stated challenges raised in the student data is a complex problem domain and the associated difficulty of understanding the project as a whole.

  [ST] *At the beginning of the project I had difficulties in understanding*

> *the project as a whole, how to distribute the work, and the scale of the work.*

Introducing the explicit use of short iterations is in line with coping with this challenge that can become a problem when the project also necessitates the learning of a set of new technologies from the students.

- They increase the *transparency* of the project state. All the stakeholders — not only the students — know where the project stands as they know what features are under construction at the moment. A personal teacher experience is that this makes it easier to monitor the students' work. For example, it is easier to monitor the task division within a small set of — per iteration — tasks, and hence monitor fairness in group work. The students assign themselves to the tasks that are explicitly agreed at the beginning of each iteration and then no-one is "kind of doing something" which would be difficult to monitor.

- They thus ease the task of project management. Using short iterations so that the content of an iteration is planned, discussed, and agreed with the customer, and the results of each iteration are reviewed, means continuous and careful project management. Using short iterations in this way creates a *communication framework* between the students and the customer. This is important as it may be difficult for the students to see the importance of communication among the many new things they encounter (Isomöttönen and Kärkkäinen, 2009).

- They help in enhancing the students' *confidence* early. This can be compared to the first results chapter of the dissertation in which it was stated that completion in the project work creates enjoyment and an experience of survival, the latter of which contributing to confidence. Interestingly, it has been stated that any method of software development works with a small team size.[5]. I consider that the educational context makes some difference so that using a small team size alone does not imply that any process will probably work: developing confidence is a very important factor in the educational context[6] and short iterations with systematized and frequent customer feedback provide an explicit framework to enhance the students' confidence early and throughout the project.

- They drive *early implementation*. The students entering the course are more inexperienced in designing real software than in programming. This may

5 Stated by James O' Coplien in INFWEST seminar on agile software development, April 2–4, 2007, Oulu. This "small teams work" perspective is also found in an online interview with Coplien: `http://developer.symbian.com/main/downloads/papers/jim\_coplien/jim\_coplien\_the\_agile\_heartv1.0.pdf`.

6 In (Isomöttönen and Kärkkäinen, 2009), it is presented how students confidence contributes to their ability to communicate. This is in line with Williams et al. (2007) who also discovered that confidence has effect on students' performance.

be due to the fact that they have mostly been exposed to assignments with limited scope (cf. (Gal-Ezer and Zeldes, 2000)), which is why I prefer early programming, thus using code as design[7]. Students often encounter new technologies (other than those used in their previous courses) in projects and the code-is-design manner of working enables them to learn the requisite technologies along with the initial, feasibility study-like iterations. This *reduces the risk arising from the "everything is new" starting point*. A common instruction from teachers in waterfall projects has been that one of the students should start to study the programming environment early in the project. This is also in line with the shift towards short iterations.

There are many comments in the student data that are consistent with the need for early implementation. The following is an apt example:

> [ST] *The project would have succeeded even better, if we had got going more quickly. In addition, making some kind of prototype in the very early phase, before completing the design document, could have helped. We couldn't take into consideration all the implementation issues in the design document, and it remained something between a user interface design and a complete design plan. I know I produced some questionable solutions in the code, and I believe these could have been avoided if I had thought the practical aspects when designing. I could have started learning to use the tool right at the beginning of the project, not only when the design work was already undertaken, this would have been beneficial regarding the design plan.*

Fornaro et al. (2007) are concerned about the students' will to code immediately in a one-semester course where students collaborate with external sponsors. I see a fit between the code-is-design manner of working and the context where inexperienced students should be able to produce a result in a stand-alone project. Fornaro et al. continue projects with a new team if necessary. This is not the intent in `TIES405`, and the difference in the conclusion may stem from this contextual difference, and also, from what conceptions of software engineering play a role in the teaching.

Notice here the relation to the section on the operational foundation. While here I propose that the stable part of the process, the practices that fit, seems to be partly based on emergence, such practices must nevertheless be identified for them to be made use of, and the teachers' reflection, as highlighted in that section, is in a key position in identifying the relevant mechanisms of the social object. Emergence refers here to two different senses: one of which is how applicable practices have emerged independently of teacher observation (ontological/epistemological perspective), and the other to how a teacher tries to make sense of observations, and thus how understanding emerges (epistemological/methodological perspective).

[7] At JYU/MIT the students often take a theoretical course on software architectures after the project course.

The challenges encountered with short iterations

The following are the challenges faced when using short iterations in the course:

- When I discovered short iterations as a key practice, I noticed that the waterfall sequence, starting from up-front project planning and requirements elicitation and specification activities, is softer for the students to assimilate and start work with. Autonomous use of short iterations early in the project necessitates a disciplined and focused project start-up on the part of the students. However, this can be achieved if the students are coached at the beginning of the project, and this is important to bring home the benefits listed above.

- How long it takes a student group to adopt short iterations depends on the students' communication ability. The students must both understand the value of effective communication and use it. The work in short iterations is based on direct communication rather than documentation. Without communication awareness, courage, and skills (see (Isomöttönen and Kärkkäinen, 2009)), short iterations remain just lines drawn into a presentation of the project timeline. Then, the work resembles a waterfall sequence that is not under control because it is still consciously trying to adhere to the use of short iterations.

- The value of short iterations depends on the nature of the project. For example, when the project developed software that replaced an older system and the major challenge concerned technical details instead of figuring out the requirements together with the customer, short iterations did not appear to be that crucial. For example, there was no necessity to provide a focus or enhance the communication framework with short iterations. However, I considered the explicit use of short iterations was nevertheless beneficial as it provided the transparency, helping to monitor the students' work as described above.

- Basing a process on short iterations has sometimes necessitated explaining to the customer why short iterations and the associated code-is-design manner of working is more applicable than a dedicated upfront specification phase. Using real examples of previous projects and explaining how the course constraints play a role in the operation helps. If customers have experience of software engineering, they might have their own conceptions on how to best develop software but they do not necessarily consider contextual differences, thus whether their conception fits the educational context. If a student group demonstrates a communication barrier towards the customer (Isomöttönen and Kärkkäinen, 2009), short iterations may not appear beneficial until the team is able to communicate efficiently. The iterative development may then first appear an inapplicable work method from the customer's perspective.

Communication

As foregrounded above, communication is a condition for the efficient use of short iterations. A corresponding role of communication will be illustrated with the example on project start-up practices in Section 5.2.3. Students' communication ability is thus in many ways a primary concern and therefore independently highlighted within the stable part of the process.

Communication must efficiently take place (1) within the student team, (2) between the customer and the student team, and (3) between the teacher and the student team.

(1) Teams that spend time together in the project room, and are thus able to communicate efficiently (face to face), tend to be successful, as opposed to those that do not adopt a shared work schedule. This observation is confirmed by reviewing the evaluation statements about projects in retrospect. For example in the set of ten projects supervised during the course development work, there is an indication that problems in communication within the group correlates with low grades. In some teams, despite their use of email, the team members were engaged on the same tasks or were unaware of each others' progress and thus unable to react to deviations from the plan. By observation, this slows down the progress of a project.

(2) I already mentioned above that effective student communication with the customer is a success factor in projects, as observed, for example, in the successful projects, that by themselves, adopted short iterations. The challenges students face in their communication with customers was studied in (Isomöttönen and Kärkkäinen, 2009). This is a highly relevant issue because problems in this communication direction may mean that students focus their efforts on the wrong software functionalities (see the student quote in Section 3.4.2).

(3) From a teaching perspective, face-to-face communication is needed because it is much more efficient than emailing or document reviews (cf. the fixed timescale) and enables the building of close pedagogical relationships. A personal teaching experience is that this also made the supervision work much more fun and appealing.

Teaching as coaching

Teaching as coaching means here an active and supportive teacher role. Below, I illustrate the necessity of this teaching approach with examples that will foreground the aspects I associate with it in the present course context.

Making the variety of project tasks explicit to students: Our study on students' communication ability Isomöttönen and Kärkkäinen (2009) provides a good example of how inexperienced students, encountering several learning challenges when the project starts, have difficulties in paying attention to all the important areas of software project work, such as communication with customers. They must be helped to identify the different tasks and get rid of the initial confusion.

Helping students to get started with tasks: I illustrate this with project planning.

One problem identified in the course is that project planning and the documentation of it take too long and sometimes get never finished. In this action research setting this problem was avoided by "initializing" the students' understanding of the tasks involved. A discussion with a student, which clarifies the objectives of project planning and gives examples of options for the various areas of project planning, initializes the student's knowledge so that the challenge encountered is reasonable: a student gets started. This process of initialization was used to explicate the variety of tasks in the projects. All students in the course are also provided with document templates and guidelines to ease documentation.

Understanding individual capabilities: I have had to consider what constitutes reasonable achievement, based on observing the individuals' skill level and learning/social challenges. I have taken the position that it is better to support individuals in completing their tasks at a tolerable level than to assume that each individual fulfills exactly the same teacher expectations regarding a certain process deliverable: by so doing the students can develop confidence and are invited to improve themselves as learners during the project.

Responsibility regarding students' rights: A certain responsibility that the teacher has comes from being more experienced compared to the students and being a representative of the department providing the course. It is important to understand this when providing students with realism in the form of a mandatory course. Responsibility means that the teacher must monitor the students rights, take the students' side when there is tension in project completion, and demonstrate the courage to intervene and negotiate with the customer. Usually the course does not encounter severe problems with customers, but some tension often arises in project completion — as noted by James (2005) whose projects were of similar degree of realism. The position taken in this dissertation is that the teacher's intervention drives the teacher's professional learning and indirectly also organizational learning. The following student quote from the anonymous questionnaire data concerns this problem.

> [STQ] *The supervisor could have terminated the project when the time ran out. A project can guarantee at most a prototype. We had to change the whole business of the customer in a new direction. So it was a rather oversized project.*

Managing student expectations: To ease the responsibility of protecting student rights and to improve the students' learning experience, both customer and student expectations must be managed. The literature (Judith et al., 2003; Laware and Walters, 2004; Hadfield and Jensen, 2007; Isomöttönen and Kärkkäinen, 2008) notes that a project teacher of a real client project course needs to manage customer expectations. Customers need to be prompted to recognize the scope of the project and told the constraints of the educational setting. In practice, I try to prepare the student group for negotiations with the customer, saying: "you should negotiate with the customer on this issue and if need be I will also bring this up"[8]. By prompting the students to take up this challenge of communicating

[8] Active coaching creates sensitive pedagogical relationships and, here, the teacher must be

a difficult issue, they can learn negotiation and develop communication courage. Practical advice to the student group is also to make their operation as transparent as possible by giving honest and undelayed information about both progress and problems (this is in line with using short iterations). In the end, it is the teacher's responsibility to intervene because the students are inexperienced and may accept the customer's expectations for the feeling of duty on a mandatory course. My experience is that some sort of teacher intervention is often needed.

It is important to communicate the constraints of the course face-to-face in the first meeting where all the stakeholders are present[9]. Customers should similarly be informed of the constraints — face to face or voice to voice — during the initial project topic negotiations (see Figure 2). This direct face to face contact helps when returning to the issue later. Then, when the project approaches closure and the customer starts asking for new features, the teacher can communicate the constraints of the project in polite phrases: "I notice that the group has less than 100 hours of resources available, so it seems that we need to carefully prioritize what we will do during the rest of the project". I have never needed to refer explicitly to the signed contract, but recall that an ex-colleague had to directly ask a customer "Is it your opinion that we have not fulfilled the responsibilities agreed by the contract?".

The GT analysis presented in Chapter 4 revealed that students associate completion with project success. Completion creates both enjoyment and an experience of survival, enhancing the students' confidence. It is for this reason why teachers must also manage the students' expectations. When a project appears to be too large in scope for a single project, and a portion of it has to be agreed as future development, the teacher should explicitly state that this is not a failure but due to the course arrangements where projects cannot have a pre-packaged stationary target. The following student quote from the anonymous questionnaire data illustrates the changing requirements and that the student has probably not experienced success.

> [STQ] *The customer's interests and objectives should be carefully defined so that the project's success is not dependent on the customer's notions. Of course the group should be able to manage this to some degree, but in my opinion the software project [the course] should be such that the success is mainly dependent on the group's [the students'] work.*

Hagan et al. (1999) have stated that problems arising from changing requirements are not to be considered a failure.

Avoiding a customer role: The teacher must principally follow only one role: a coach who supports the students' collaboration with customers. With attractive real-life topics, it is easy for the teacher to slip into thinking up functionalities that could also be included in the software. If students have to evaluate the

careful in guiding the students. It is important to not create a culture in which the students directly answer no without any considerations of the resources a customer's proposal would actually need.

9 A practice first used by a former colleague.

feasibility of the teacher's ideas on top of those of the customer, it takes inevitably more time and confuses the focus of the students' work. I was careful about this and it saved time (common sense observation). Outside this research, the course has included usability-focused testing conducted by the teacher, and this is when it's easy to slip into the customer role. I also discussed this tendency with some technical supervisors.

Providing sufficient feedback: The necessity of active participation became evident by the following. When I first took an active role as a teacher I occasionally intervened too much at a low level of operation. Basically, I assume this was due to personal excitement when shifting towards short iterations. Some of the students commented on this informally. I then took a slightly less active role during the year 2008, but problems followed and I noticed that the principal issue is to focus on how to interact with the students during the active teaching — not to reduce this activity (Section 5.3.2 focuses on how to interact with the students). With the teacher in a less active role the students started to comment that they would appreciate getting more feedback during the project to know how they are doing. Students are inexperienced and wish to hear during the project what issues they should pay attention to and improve on instead of receiving feedback as a final judgement in the form of a course grade. Students are then able to improve their operation during the project and have a chance to affect their grade. Moore and Potts (1994) made a similar reflection: *it might be helpful to tell a student how he/she is doing at midterms*.

The examples above illustrate that teaching as coaching is about understanding and accepting the constraints of the course. An active and supportive teacher role is associated with fairness from the students' viewpoint. If this is compared to the first results chapter of the dissertation, particularly the observation that students' negative course experiences have to do with the feeling of unfairness, teaching as coaching can be argued to be very important in the course, as shown in the student data. Notice also that it was foregrounded in the same chapter how students' positive experiences are conditioned by support.

Regarding these examples, a challenge and shortcoming during the research was the fact that I did not use any explicit and frequent intervention practice which would have always ensured sufficient feedback on the issues to be improved during the projects.

Relations between practices in the stable part

In the above, I every now and then referred to how the practices of the stable part interrelate. Figure 19 summarizes these relations, which are bidirectional. The connections marked with numbers are explained as follows:

1. Communication skills, communication courage, and awareness of the importance of communicating are needed to be able to autonomously work in short iterations with the customer.

2. Short iterations create a communication framework.

3. Explicit demonstration (coaching) is needed to support the adoption of short iterations. The students' understanding must be "initialized".

4. Short iterations help to monitor the students' work and assess the need for coaching.

5. Face to face communication is an efficient way of teaching, and is dictated by the one-semester time constraint.

6. A teacher must help the students to understand the necessity of communicating.

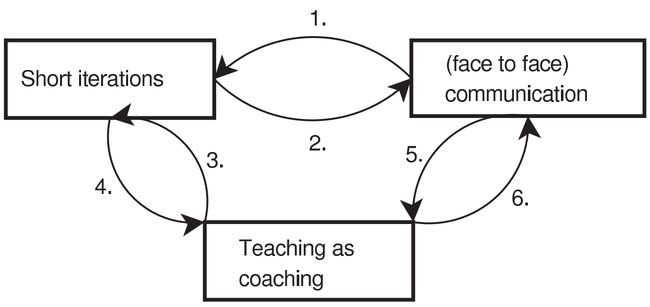

FIGURE 19 Relations between practices in the stable part

5.2.3 Unstable part of the meta-process

As summarized in Section 2.8, there is variation in the course settings of `TIES405`. The meta-process takes account of this, stating that a part of the software process is derived from the unstable part of the course context. This changing part of the software process is associated with on-going process tailoring. The example given in Section 5.1 has already illustrated the necessity of process tailoring. In the following, two more examples are given. The first relates to the practice of estimation and the second to that of how to support the students' project start-up phase.

Estimation. I tried XP's planning poker[10] with the students. The first session, in which relative effort points were given to the initial set of requirements elicited with the customer, was of value. The participants were communicating. Design aspects were discussed and open questions concerning the requirements were identified. However, when I tried to get the students to track and learn their team velocity (what is the relative effort they can manage per iteration) problems were encountered. The students did not see any further sense in the relative estimation after the first round and they neglected the whole thing. The interpretation was that because the students were anyway required to log their working hours in

[10] An estimation practice where participants (team and additionally the customer) vote on a relative implementation effort for each specified customer requirement.

the course[11], the relative estimation introduced something they experienced as a duplicate practice. Also, the estimates of inexperienced students in a from-scratch project were observed to be more or less guessing. It is difficult to learn about velocity during the single project with a newly formed student team that does not have an established organizational culture as its background. It is also inevitable that students who study other courses in parallel with the project course cannot adhere to fixed weekly hours (cf. 40 hours week in XP) which makes the learning of velocity unrealistic. The practice became that of simulating certain software development practice ("simulating realism") and I did not push the students into what I observed as too much overhead.

I had earlier tried a simpler estimation where I informally prompted estimations with questions "How much time do you think it will take to finish the task" at the beginning of each iteration. By asking the same question of all the team members, communication was triggered and some rationale was achieved, only using work hours as opposed to relative measurement. The students were not pushed to learn their velocity. I considered that it was also better to ask these questions in the meetings with the customer so that the customer got a realistic picture of the potential pace of progress and the students got used to transparent working. This was a less formal way to estimate the workload and appeared to fit better, "fit" meaning here that the practice of estimation could be communicated to the students but not in the form of an irrelevant forced practice (cf. the objective of the action research at the beginning of the chapter).

These experiments suggested that the planning poker sessions are a good alternative in projects where there seems to be a challenge regarding the students' communication courage and requirements elicitation. The less formal estimation procedure gave students a hint of time estimation and could be used throughout the project. Altogether, the conclusion from these experiments was that estimation is guessing in the educational context where the students do not have significant previous experience in software engineering. I thus do not regard estimation as a condition for a successful project. The question whether to use an estimation practice and, if so, what type of estimation practice should be used remain open issues of situational importance.

Project start-up. I have asked student groups to develop a conceptual drawing[12] from the topic description they are given in the starting lecture. Such drawings illustrate how the students make sense of the project purpose. The aim is to make students aware of what they know and do not know and prepare them for efficient communication with the customer (cf. meta-cognition (Flavell, 1976) and constructivism (Ben-Ari, 2001)). This practice turned out to be very useful, resulting in a first version of customer requirements within two to three customer meetings at the beginning of the projects, but I came to notice that there were exceptions.

---

11 Credits are given according to spent work hours, see Section 2.5.

12 Compare this, for example, to the accelerated analysis in the feature-driven development method where a conceptual overview (object model) is first prepared and discussed to achieve a shared understanding (Palmer and Felsing, 2002, pp. 63-64).

Namely, with a project topic that was readily understandable as a whole, the challenge being in the technical sophistication needed to update existing software, I noticed that the practice would be valueless if the preparation of a conceptual model focused on the customer problem of interest, i.e., what should be done in the project. Instead, visualizing the technical inner workings of existing software that was to be updated was more valuable, as it promoted understanding of the actual problem that needed to be addressed to start working efficiently. Encountering this kind of project topic reinforced my conception that the process in the course is to be based on high level practices that seem to "always" fit the context, while the rest tends to depend on the situation at hand — more or less.

The more interesting point here concerns the students' communication ability, as studied in (Isomöttönen and Kärkkäinen, 2009). That is to say, success in giving students the kind of start-up practice that would help them to make progress with the customer depends on the students' communication courage, communication skills, and their awareness of the importance of communicating. Students do not take properly advantage of the practice involving communication if they are challenged in their communication ability; e.g. they do not see the point of efficient communication. The practice of preparing an overall conceptualization of the project purpose is about inviting the students to engage in knowledge acquisition process together with the customer, and without the requisite communication abilities this invitation is difficult for a teacher to trigger. Although the start-up practice was found to be very effective, and thus could belong to the stable part of the process, the more primary issue here is that the teacher helps students to overcome their communication challenges (cf. stable part practices and their relations, shown in Figure 19).

These examples illustrate that there is a changing part in the software development method applicable in the course. This kind of process tailoring based on reflection yields pattern-like knowledge that could be maintained as an easy-access experience factory (Wiki) accessible for all project stakeholders; see (Ras et al., 2007). Such an arrangement would make the tacit knowledge that is generated in the course more easily accessible and maintainable. Knowledge that is currently embedded in a large set of archived project documents is implicit and therefore difficult to utilize efficiently.

5.3 Relationship between teacher and students

This section elaborates the relationship between the teacher and the students (see Figure 16). Section 5.3.1 translates the operational foundation given in Section 5.1 into a teaching principle, arguing on the basis of student data for the contentual focus of teaching to take into account of activities that fit the course context. Section 5.3.2 translates the operational foundation into a characterization of interaction between the teacher and the students.

5.3.1 One-track principle

The operational foundation outlined in Section 5.1 states that operation in the course must be based on activities that fit the context, principally sourced by teacher reflection. “Fit” means that students should be able to produce a reasonable result for the customer. Translated into a teaching principle this means that the project stakeholders need to have sufficiently shared objectives so that the teacher principally supports the students in their effort to produce a reasonable result for the customer. This is called here the one-track principle. It means that students should not encounter teacher expectations that annoyingly contradict the progress of the project. This is no problem regarding the learning objectives: as put forward in Section 5.1, the learning objective for the students is to consider relevancies in the course operation.

This principle can be argued by reference to the problems that arise when things have gone differently. The teacher-as-researcher experiences and student data are here in line with each other. The following student quotes along with research interpretations illustrate three areas of software development in which students have noticed irrelevancies. These are an up-front design phase disconnected from programming activities, project-wide workload estimation at the beginning of the projects, and too much emphasis on documentation. These tend to be *simulation of realism*, activities that do not fit the context but are nevertheless practiced, as I similarly noted with the estimation example in the previous section:

> [ST] *Once the design document was completed, the design changed almost on a weekly basis.*
>
> For the inexperienced students the “code is design” tends to better bring out design issues than the upfront design in the paper format.
>
> [ST] *We especially had problems with the design document. Well, this was partly due to the fact that the actual design was still incomplete.*
>
> The design document had no value in relation to the work. Design without implementation — or final approval of software functionality — is difficult.
>
> [ST] *The only estimate that was wrong right away, was the estimation on the project’s workload.*
>
> There is no means for producing a reasonable project-wide estimation at the project outset as the students are starting with a new team, and facing a new project topic, a new domain, etc. (cf. the “everything-is-new” starting point summarized in Chapter 2).
>
> [ST] *The workload estimation failed, and we were in a hurry towards the end, but I’ve heard that it’s not unusual in the field.*
>
> The students spend time on “guessing” at the beginning of the project which is one reason for the hurry at the end.

> [ST] *Paper work took too many resources from the other work.*
>
> An emphasis on documentation (amount, timing) slows down the actual development effort.

It could be argued here that students learn from mistakes and are therefore capable of reflection, but further analysis of the student data indicated that it is risky in this kind of realistic context to teach on the basis of conceptions that are not grounded in the context itself, and thus have teacher/course expectations that contradict the project goals/progress. It is risky because[13]

1. only some of the students are able to see an interrelationship between the project outcome and task irrelevance, and students may feel unsuccessful even though in part the underlying reasons are independent of their efforts.

   Some students simply mention that they didn't expect an emphasis on a certain area in the project work. For example, they say that their preceding studies did not prepare them for such careful technical documentation. They pay so much attention to something unexpected that it becomes included in their course experiences. Some students wonder if certain tasks were really necessary. For example, the students felt that all that mattered was the documentation. Or, they wonder if the granularity of the resource allocation was too detailed. At the same time the students consider that the project course gave them a realistic picture of software engineering work. Thus, the students accept what they are expected to do, even if they may wonder about it. Some of the students value the skills learned on the "second track" despite its conflict with project progress. This particularly takes place if they notice an improvement in their second track skills — for example, an improvement in their writing skills.

   This should be compared to Chapter 4, and the component "occupational thought process is reinforced". Students are able to reflect on the project progress but their view may be problematic when they do not notice the second track.

2. those that notice the problem are frustrated (cf. Chapter 4).

   Some students take it easy and their frustration is translated into amusement. Some students are highly critical: they experience the arrangement or emphases on certain practices as absurd. They compare the relevance of the project tasks to the resources available and then report severe frustration. Some of the students' frustration explicitly indicates the two-track problem:

   > [ST] *We felt that we were squeezed between the supervisor and the customer.*

[13] I use the term "second track" to speak of activities and expectations that do not sufficiently support the students' work towards the goals of the project.

An excessive emphasis on documentation seems to create most of the student frustration and it seems that here the students' criticism of the course is more hard-hitting in the anonymous questionnaire data than in the public project reports (see data sources in Section 3.7).

> [STQ] *In working life the tweaking of documents for weeks on end would lead to firings.*

3. inequity emerges.

   Student groups notice the problematic second track when they compare their work to that done in their neighbor project groups. They observe inequity. There are also direct comments in the students' project reports indicating that it was the teachers' and the customers' focus on the relevant project tasks that enabled them to finish the project on time.

   > [ST] *Moreover, the project documentation focused on what is relevant and neither the teacher nor the customer considered irrelevant issues. In this way the project ran to time.*

   Similar comments were also found in the questionnaire data. The students brought up the issue when asked how to improve the course.

   > [STQ] *Fairness: different project groups were not in an equal position regarding, for example, the requirements they received from the supervisor.*

The second track issue is observable in the student data and indicates a concern with respect to the teaching focus. As the theory of the first results chapter suggests, potential problems in realistic course contexts should be carefully analyzed and reacted to, even where the overall course experiences are positive.

However, the second track problem is not a black-or-white matter. When the resources are available, the teacher can add content (challenge) to the project by focusing on practicing certain SE tasks in detail. This is what Burge and Gannod have noted (see Chapter 2). I have myself experienced this by adding detailed documentation regarding system testing into a project, and I find projects in the course history where documentation-oriented projects have been finished on time, probably at least partly due to the small size of the product (number of code lines).

Practicing the one-track principle — the challenges encountered

The major challenges concern how students' reflection can be supported, and this will provide a link to the next section concerning the teacher's interaction with the students. The challenges can be summarized as follows:

1. a fair share of the students' resources goes into the learning of a systematic way of working (whether such a way fits or does not fit the context) and it is

likely that this reduces their possibility to reflect at the process level. Focusing the teaching on process level understanding resulted in some students commenting on their learning of a software process in their written course feedback.

> [ST] *As a team leader, in particular absorbing the agile process model was interesting.*
> [ST] *... in the team leader role, the process model became very familiar to me.*
> [ST] *The most of the learning comes from the process model, project work, meetings, and other social interaction.*

The quotes are from the projects that used short iterations. Basically, the students are happy to learn the process, but do not really "examine" it in their course feedback. Here I came to consider the students' course work in the context of Maslow's (1943) hierarchy of needs, and by searching for a joint reference to "project management and Maslow" I found that this analogy had been presented earlier[14]. Accordingly, if the students first learn basic skills in software engineering and group work, they are likely to have more resources for reflection at the process level in a customer project. In any event, there is an indication of a change in student responses, with students increasingly explicitly using the term "process" in their responses, which may mean that they have internalized what process means in a software project.

2. in any case, there are considerable challenges in this kind of course context, and such individual challenges can steal the students' focus away from generic process level reflection. The students' course experiences of projects aiming at a one-track principle (finished on time) do not indicate "second track frustration" but disclose that there is enough challenge. The trade-offs presented in Chapter 4 remain. There are new technologies, group work challenges, communication challenge, etc. Focusing teaching on process level issues does not guarantee success, because it is conditioned by these inevitable challenges.

3. not all the students have sufficiently been invited to engage in process level reflection within the present research effort. As the above quotes illustrate, student team leaders focusing on project management are in a good position in the student group to reflect on process level issues. Inexperienced students tend to reflect on issues that are "close" to their role and activities in the project. From this perspective, special attention needs to be given on how to invite all the students of a group to engage in such process level reflection.

[14] `http://www.chacocanyon.com/essays/hierarchyofneeds.shtml`

5.3.2 Interaction: dynamic dialog vs. authority

This section discusses the difficulty of interacting with the students when seeking to ensure their learning. I came to notice that this is one of the major concerns in teaching in a realistic context in which students' positive experiences are conditioned by giving them both responsibility and support (see Chapter 4). It is a great practical challenge for a teacher to act in an appropriate way on this continuum. By undertaking project teaching in the form of coaching, I ended up with the following characterization on appropriate interaction: teacher interaction with students is *dynamic dialog anchored to negotiable competence-based authority*. This is associated with the operational foundation stated in Section 5.1: space has to be left in the interaction to address student learning and acknowledge that students also affect what counts as applicable operation, while some sort of authority (instructiveness in practical sense) arises from the teacher's epistemological position. In the following, I discuss the themes found under this characterization. Although the chapter has a specific section for a literature comparison (Section 5.4), I include some literature in the presentation as I noticed close analogies that help in communicating the results and serve as argumentation.

Dynamic dialog

I found the term *dynamic dialog* to be an appropriate synonym for the liberty needed in interaction between teacher and students in the `TIES405` context. The term is sourced from Robinson (1994) who highlights the dynamic nature of dialog in the context of empowerment. This refinement of empowerment has a strong element of liberty in it. Participants in the dialog need to feel free to increase their competence and they have changing roles. Both the students and the teacher teach, and they both learn. Dialog is also emphasized in constructivist teaching; Peavy (1999) situates dialog in central place in constructivist teaching that stresses social interaction between a supervisor and the person supervised. The nature of the dialog is such that neither of the participants aims to rise above the other. Listening and empathy are the important supervision skills in this dialog.

In the `TIES405` context, dynamic dialog in interaction is needed because student groups working on their project topics on a daily basis become more familiar with the customer's problem domain and the software than the teacher. This is why students have insights as to what is relevant at a particular point in the project and what order their problem solving processes should take. This is an important single observation, a fact to be accepted by the teacher. This is why the interaction between teacher and students has to be dynamic in nature. Without such a dynamism, there is an immediate risk of two tracks in the students' work, one in which deliverables are prepared for the academic track, in the worst case without a rationale, and another in which software is developed for the customer. It is thus important that the teacher always engages in dynamic dialog when giving feedback to students on a process deliverable and lets the students affect the operational knowledge. A shared understanding of the appropriate strategy to re-work the deliverable is needed.

The need to have dynamism in order to address student learning can be illustrated with an example of the lack of dynamism. I return here to the issue of an overly active teacher role (see Section 5.2.2). Along with the strategy of active teacher participation undertaken under the action research setting, I started to receive direct comments from the students about my role being too instructive. Instead of prompting students to consider relevancies, I interrupted and took over their problem solving processes which was likely to contradict with their possibility to experience ownership of the problem. Maehr's (1984) theorizing on meaning and motivation provides an explanation. He states that the meaning of a situation and the associated motivation is determined by how an individual experiences his or her action possibilities in the situation. This is in line with my experiences: too much control at a low level of operation constrains the students' perception of action possibilities, thus reducing motivation.

Having stated that dynamic dialog is an appropriate characterization for interaction in coaching, it must be stated further that dynamism is also inevitable because of the presence and changing requirements of real customers. The teacher has occasionally to act without a chance to lean on any prescriptive principle. This is an important observation as it illuminates why following a structured teaching method, e.g., a group counseling method suggested by Borgen et al. (1989), which assumes designing of activities for the group, is not straight-forward.

The content of the dialog is also important. Based on my experiences, concentrating too much on being congenial may result in difficulties in focusing on serious project matters. This may happen when the teacher spends a lot of time with the group and particularly when the conversations are often off-task. As a consequence, an atmosphere other than learning-oriented one can emerge in customer meetings, where the results of the projects should be inspected in a mature way. This example demonstrates the sensitivity recognized in dynamic dialog and teaching-as-coaching: the teacher affects and can affect the work culture of the project (values and norms). The literature warns about the risks of a "blind" humanistic view in teaching/learning (concentration on nice) (Järvinen et al., 2000). It may cause failure in respect of learning objectives. Notice that constructivist teaching, which underlines the socio aspect of supervision, also breaks away from a naive focus on the overly positive teaching mode (Peavy, 1999, p. 163).

Inevitable and applicable authority

There is a long tradition of instructive and scheduled classroom teaching in academia. One reason for this could be that instructiveness — as opposed to the situational approach to teaching — guarantees equal treatment of learners. All the students are treated with a same prescribed set of requirements. In `TIES405`, instructiveness is needed because of the contextual constraints. The fixed timescale, real customers, and inexperience of students imply discipline and instructiveness in interaction. Instructiveness is particularly needed during project set-up. Rapid progress is immediately needed instead of an explorative manner of learning, such as discovery learning (Bruner, 1961). If inexperienced students just come up with their own operative model in the one-semester context with real customer expectations, progress starts in such a slow fashion that it is difficult to provide real value for the customer within the time constraint. The project model is then not feasible.

A key consideration here is that when a teacher needs to instruct, this implies some authority over those to be instructed. This authority has to be based on a certain kind of competence which is discussed below.

The most important property of competence is directly what was stated in connection with the operational foundation in Section 5.1. The teachers' competence has to arise from reflection on the project course context. Such competence means that instruction and supervision are relevant. If competence is sourced from external preconceptions, for example, based on process prescriptions not carefully rationalized in the local context, there is a risk of irrelevance in teaching: second-track issues are introduced (see Section 5.3.1). One can also compare this to the preference for emergent concepts in Grounded theories. Concepts that are generated from the research context, not borrowed, fit, they are relevant. Similarly here, competence derived from the course context itself, not borrowed, fits.

This kind of *competence-based authority* can attract students because of its relevance. Strike (1982, p. 52) points out that a learner must willingly submit to the expertise of the master, and I have found that the relevance of instruction is a major determinant of expertise that is willingly submitted to[15]. However, in addition to relevance, it is necessary to consider how to communicate the teacher's competence to the students. There are two aspects to this, one of which is suggested by the constructivist literature: the teacher needs to become familiar with the students' prior knowledge level to be able to actualize his or her competence in an accessible and attractive way, as argued, for example, by Feldman (1980), who points out how a child is attracted to learn from an older sibling if the gap to be filled is of a sufficient size. Being close enough to the learner's current knowledge structures creates a need (motivation) for the learner to restructure his/her cognitive structures to the tempted level. The other relates directly to the objective nature of operational knowledge: I interact with the students by transmitting the relationship between the teacher and operational knowledge, i.e., by showing that the teacher's proposals are not opinions but based on critical observations that can be argued empirically (see Popper's examples of second and third world knowledge in Section 5.1.1 and the discussion in Section 5.1.3).

Another practical perspective on competence-based authority is not hiding areas (and moments) of incompetence. This requires honesty and courage from the teacher. In unpredictable situations[16], to avoid annoying situations, the

[15] See the example of process tailoring in Section 5.2.3; When trying to get the students to evaluate group velocity, they ignored instruction on what appeared to become simulation of realism.

[16] As I noted earlier, in a real customer context a teacher is likely to encounter situations that

teacher has to be able to say “I really don’t know, we have to figure it out”. Honesty leads to trust[17], and importantly, seems to decrease the student stress emerging from the fairly serious course work. The teacher does not “hide behind a structure”, and the students notice that it is not unusual to not have a ready-made answer in a real situation. It is not poor operation if they do not know how to operate. This again is an example of how a teacher can affect the work culture of the student group, here to accept and cope with uncertainty. In terms of Peavy’s socio-dynamic counseling, this is about “outsourcing” problems away from persons so that they become external and can be managed (Peavy, 1999, pp. 107-108).

Finally, competence-based authority has always to be negotiable; Even where the social object, a realistic project course work in software engineering, is considered so stable that conceptualizations are meaningful, the conceptualizations are not put forward as fixed conceptualizations. “Theory as a process”, “ever-developing”, Popper’s statement on how explanations of problems tend to create new problems, etc., are here the underlying paradigm level assumption that necessitates negotiability.

Linkage between dynamism and instructiveness

Consider the following example. A student group enters the project course. As a teacher I wish to meet the group after the starting lecture. The aim is to become familiar with the group and begin discussing the areas or project work that the students will first encounter. I start by asking how much the students know about processes and proceed by explaining the possibilities regarding the process to be used in their project. It is not unusual that some students do not attend to the dialog, they answer only when prompted to. Is such a student unmotivated, too timid to communicate, or too unfamiliar with the terminology used so that there is too much challenge right away[18]? The question illustrates the multidimensionality in supervising a group, and how it arises from individual differences. The student sitting quietly may wish only to have direct instructions and feels uncomfortable if pushed into dialog. The literature suggests that the person’s previous experiences in learning affect the way he/she wants to be taught. Rogers (1983, p. 154) notes that there should be provision for those who do not want freedom and prefer to be instructed. Thus, dynamism (liberty) for a student does not necessarily mean dynamic dialog but receiving instruction.

This challenge means that the teacher should pay attention to both group and personal characteristics. This, however, is difficult because observable behavior is unlikely to reveal underlying cognitive processes (Gibbs et al., 1980). To address this challenge I aimed to build a pedagogical relationship with each student in a group. In practice, the method I advanced — first learned from a former

cannot be pre-planned.

17 According to Strike (1982) trust is a constituent of successful pedagogical relationship.

18 Peavy (1999, p. 162) states in the context of socio-dynamic counseling that the language used should be familiar for all participants.

colleague — is to go to meet the group informally without a set appointment. I prefer to do this on almost a daily basis and often without any prescription in mind. The key issue here is that such meetings are of an informal nature, and they become such if the teacher does not plan them. Such visits to a student group starts with "How are you doing with the project?" and attention is given to all the students in the group. My experience is that this appears to reduce the initial teacher-student hierarchical tension and gradually enhances dialog with all the students.

Challenges encountered

This Section 5.3.2 has been about challenge: when the emphasis in learning objectives is transferred from the learning of individual skills to reflection, the interaction between the teacher and the students becomes the major issue. The following can be added to the discussion in the section:

- The group remains a black box due to a dominant yet strong team leader: it is then difficult for the teacher to build a pedagogical relationship with each student in the group. This complicates the assessment of individuals as the team leader takes lots of responsibility but leaves the rest of the team a black box.

- Teams with communication problems due to conflicting opinions: the teacher must try to support and motivate both camps and prompt the students to resolve the issues. Instructiveness may help so that the teacher clearly sets the direction for the project based on his/her experience and justifies this with examples for both camps (thus by showing students empirical evidence).

- Unequal commitment and team members not working together: These affect the mood of the group, making it difficult for the teacher to have positive and learning-oriented interaction with the students (see the student quote in Section 4.1 at the point "Trade-offs").

5.4 Comparison to literature

In this section I discuss the major themes of the theory presented here in relation to the literature. Similarly to Section 4.2, the nature of this section is reflectional and discussional, as opposed to an aim to force the literature to fit the present theory.

Identifying operational foundation

The present theory answers the concerns stated by Parker et al. (1999) in the context of real client student projects:

> *We find it a considerable challenge to our skills as educators to draw out relevant methods, approaches and techniques from the software engineering literature, in response to particular problems raised by the students in discussions.*

According to the present theory, the SE literature is a pool from where to source practices, but essentially a teacher observes emergent practices and needs to critically evaluate how practices that are imported into the course context fit that context. This means observation and conceptualization from observation. Usually, project courses yield process deliverables that can be analyzed to support the observation and conceptualization. For example, `TIES405` projects have been fully documented and there are plenty of empirical data available for a teacher to check his or her observations.

Objective knowledge vs. empirical modeling

In Section 5.1, I used Popper's concept of third world as an analogy in communicating the results. A similar idea and analogy can be found in the empirical modeling, so I interpret it here. Beynon et al. (2009) speak of empirical modeling, seen as a constructivist pedagogical approach to computing which from an epistemological point of view admits that knowledge is generated by experience. It is knowledge that has to precede established knowledge in a propositional form (cf. domain understanding vs. program code). Regarding its epistemology, Beynon et al. source an analogy from radical empiricism. They refer to William James's (1996, p. xxii) conception that knowing is rooted in direct experience, and his thesis that relationships between experiences are themselves given in experience. They arrive at saying that though such knowledge is of its essence a personal matter, this is no obstacle to its potential classification as having an objective quality, if one's own experience is experienced as cohering with that of another person experiencing the same situation. It is suggested that Beynon et al. are speaking of knowledge similar in nature (its objective quality), but arrive at it through another route.

Interestingly, the idea of empirical modeling has been used in an unpredictable learning context, embedded in a software product that collects information from the learning context for a teacher, who can then act just-in-time (Jormanainen et al., 2009). In the present study, a related idea was conceptualized through action taking in a capstone project context.

Meta-process: emergence of short iterations

The shift towards using short iterations was not an action randomly derived from contemporary software development, but was indicated by observation, as described in Section 5.2.2. In this instance, taking actions was about explicating the use of short iterations. They were no longer used more or less consciously, and occasionally, as a "closet" process with a waterfall sequence. As suggested in Section 5.2.2, one could speak of emergence. In this connection, an interesting statement is found in Fornaro's et al. (2007) paper concerning one-semester student projects with external sponsors. The paper provides experiences gathered over a ten-year period:

> *CSC 492 imposes no specific demands on the format of a software development methodology, other than that there must be some well-defined process in place. The exact form is normally negotiated between student teams and the sponsor.*

This resembles what has been stated with the meta-process above, i.e., that an applicable process consists of a stable part and a changing part. The same paper also states:

> *The iterative approach [2–4 week cycles] is by far the most popular; it allows students to develop smaller portions of systems and thus receive more frequent feedback from mentors/advisors. Timing constraints imposed by the 16-week semester time-frame often steer teams to this iterative format.*

These two quotes indicate not only that a stable part could be identified in an applicable software process but that short iterations are an option in the stable part in the course contexts of this kind. Here also, the latter quote gives an impression of emergence, with iterative development (Basili and Turner, 1975) finding its way into this kind of course context. Similarly, Brügge (1994) concludes that iterative development fit an unpredictable real-life course context.

It seems that by diagnosing and taking actions the present research have arrived at what Brooks (1987) concluded regarding the importance of iterative development in the face of inevitably complex software. Brooks identifies that specifying what should be done is the most challenging single issue in software engineering, and that early prototyping, and developing software in an iterative fashion is necessary to meet this challenge. Note that when Brooks speaks for iterative development he is referring to the problem of customer inability precisely to state the requirements of inevitably complex software. The context of the writing is thus that of a customer project, as is the case with TIES405, and the papers by Fornaro et al. and Bruegge.

Notice here that short iterations, belonging to key agile practices, were not chosen in order to teach best practices of contemporary software development but because they appeared to arise from and fit the context.

Inductive process improvement and contextuality of software development

The software engineering literature has identified two approaches to process improvement. These are the inductive and the prescriptive (Gorschek, 2006). The inductive means that improvement is based on an understanding of the current situation in a project organization, whereas prescriptive means that it is based on best practices in industry. By referring to the CMMI-model (CMMI Product Team, 2002) and an ISO standard (ISO/IEC, 2002), Napier et al. (2009) state that the prescriptive approach is the dominant of the two. The inductive approach to process improvement most obviously overlaps with the acknowledged issue of contextuality of software development; see e.g. (Bern et al., 2007).

The operational foundation states that the applicable process is derived from the development context, in which case the approach is inductive/contextual. However, process *improvement* as a title framework for this chapter is somewhat misleading as the changes do not necessarily mean improvement but a better fit only in the current context and situation.

From a curricular perspective, the process approach of the dissertation provides a means to communicate this inductiveness/contextuality to the students during their study path.

Agile software development and knowledge acquisition

Clearly, the stated stable part practices can be associated with the practices highlighted in agile software development such as Scrum (Schwaber, 1995). Scrum is characterized as a project management framework that does not define the actual development procedures (Abrahamsson et al., 2003). This corresponds to the present theory that suggests a process skeleton that should be counted on. The present theory is, however, more specific, suggesting that the rest of the process must be continuously tailored and, in particular, that teacher reflection with criticism is the key to this tailoring activity. To my knowledge this aspect of the teacher role does not correspond to the Scrum master role, which assumes self-organizing teams. I also identify that the facilitating role of the coach has a clear educational emphasis in the present theory compared to Scrum. For example, teaching as coaching must consider the individual abilities of inexperienced students (“tolerate blocks”) to invite all the students to engage in active participation and learning; see “teaching as coaching” in Section 5.2.2.

From a more abstract viewpoint, I assume I have arrived at a view of software development as a knowledge acquisition process (Armour, 2003). The emphasis on short iterations, communication, and teaching as coaching all enhance knowledge acquisition concerning what is developed, and needed between the customer and the student team. I could thus state that the course model is managed by viewing software development as a knowledge acquisition process.

Altogether, I perceive that the research setting has not been one of verifying a particular software development approach in the studied course context but one of coming to know what fits the local context.

Importance of communication

Communication skills are stressed in the current view of engineering work (Norback and Hardin, 2005), employers rank them high (Teles and de Oliveira, 2003; Lamp et al., 1996), and they are recognized as a condition to advance one’s career (Polack-Wahl, 2000). Students’ communication skills have received a great deal of attention during the history of `TIES405`. For example, the two project presentation rehearsals aimed at preparing students for the final public presentation (see Section 2.5) provide a safe environment for the students to practice communication, and have been very well received by the students. The contentual emphases of the present theory (communication was included in the stable part of the meta-process in Section 5.2.2) continue the focus on the students' communication ability, but the theory incorporates communication in the sense of making the course a feasible arrangement instead of teaching particular skills.

The present theory foregrounded that students' communication ability is in many ways "primary", and therefore earns a position in the stable part of the process. Brügge (1994) concluded that there was a need for iterative development in the real customer project context, but noted that this can lead to communication problems and breakdowns, leading in turn to unsatisfactory end results. He argued the need for the inclusion of social, organizational, and communication issues, which seems to be in line with the present theory — spoken of in the same sense.

The present theory suggested that short iterations create an explicit communication framework. One could thus speculate that partly because of the need to communicate efficiently in the fixed time one-semester real customer course context, short iterations have emerged in that course context.

Teaching as coaching

Teaching as tutoring/mentoring/coaching has long been acknowledged in education (Wood et al., 1976) but a focused paper on coaching in computer science/software engineering education is difficult to find. Dubisnky and Hazzan (2003) present one that is grounded in extreme programming. Hedin et al. (2005) report on a specific coaching course run in parallel with an extreme programming project course. Mentoring/tutoring/coaching is, however, often included as a topic or at least briefly mentioned in a descriptive student project paper, e.g. (Törngren et al., 2007). It can be concluded from these references that coaching means guiding and active role, communication triggering, intervention, and balancing between educational and customer objectives. It is found challenging because of the close contact with the students and because it requires effort, consideration, experience, and training. These coaching qualities are in line with the present theory, except that in the present theory teaching-as-coaching was identified as a component making the course feasible with respect to the action research objective. The present theory illustrated coaching with examples to ensure a context-specific view of coaching, and the common theme of the examples was fairness (see Section 5.2.2). To my knowledge, the CSE literature does not emphasize the conceptualization of fairness in coaching practice.

The examples given of coaching practice (making project tasks explicit to students, helping them to get started by ensuring their have sufficient understanding of the tasks and giving them guidelines such as documentation templates, understanding their abilities individually in relation to a reasonable achievement, and managing of student expectations and rights) can be viewed as conforming to constructivist teaching with a phenomenological humanist flavor.

More precisely, both the constructivist implication, that a teacher must ensure that a learner has sufficient prior knowledge for a task (Ben-Ari, 2001), and scaffolding, in which a teacher first gives guidelines and then lessens his/her guiding role (cf. zone of proximal development (Vygotsky, 1978)), are applicable in seeking to explain the constructivist component. The attention on individuals, their ability, expectations, and rights, gives the phenomenological humanist flavor interestingly demanded by Dahlin (2001).

Here the constructivist implication and scaffolding are, in a way, linked to the conscious aim to accelerate student learning and then contradict with other constructivist constructs such as discovery learning (Bruner, 1961). However, this is not seen here as neglecting an explorative manner of learning, but is rather about understanding the teaching context to make it feasible.

Learning objectives; contentual focus in teaching

I observe in the literature that real customer projects often specify learning goals as giving students a realistic experience and enhancing skills, e.g. (Knoke, 1991; Johansson and Molin, 1996; Bothe, 2001; Christensen and Rundus, 2003; Mochol and Tolksdorf, 2009). This "taste of the real-world" was also identified in the taxonomy by Fincher et al., as the objective of real client projects (see Section 2.8). As given in Section 2.2, realistic experiences and skills have also been defined as the learning goal in the context of `TIES405`.

These observations suggest that explicit support of students' meaning making (reflection) may have not received much attention in the context of intensive real customer projects. A reason for this may be the fact that student resources are easily expended on surviving the project assignments, as discussed within the present theory. An extract from the paper by Parker et al. (1999) well identifies this challenge that comes with intense real customer projects:

> *Some of the issues concerning professional software engineering practice that preoccupy us as academics are probably lost on many of our second-year undergraduates in their dedication and determination to build working, finished products.*

The present theory suggests that enhancing students' holistic understanding so that a teacher aims to share his/her operational knowledge with the students should be emphasized. This is here dictated by the course constraints: a high focus on specific skills easily contradicts with the goals of a project and therefore learning must be devised in another way[19]. Learning objectives in the context of

[19] This relates to the concept of two audiences/tracks/targets in students' work, which is found in the literature. Clear et al. (2001) state that contradictions between educational and customer objectives must be continuously monitored and resolved. Similarly, Fincher et al. (2001) find that real client projects involve balancing between academic objectives and customer requirements (see Section 2.8). In Chapter 2, and within the present theory, I also referred to Gannod and Burge (2009), who have noted that content with an educational purpose can be added to projects if they lack challenge. Notice that two audiences

the present theory (relevancies in operation) are directly consistent with the customer's objectives and a potential academic "second track" is considered a problem. Here, one can compare the example in Section 5.1.4 with the paper by Shafer (2002). In the latter, project studies appear to have high focus on process deliverables (documentation) and it is observed that over one thousand student hours is not enough for a reasonable project in one semester. In contrast, the example in Section 5.1.4 illustrates how software engineering practices are tailored to fit the context (real expectations and one-semester timescale), and that the learning objective is thus to consider the tailoring activities together with the students.

While shared reflection is seen as the primary way of learning in the present theory, it was noticed that without an explicit teaching method it is difficult to invite all the students to generic process level reflection. I found that some students started to explicitly mention "process" in the course responses, which is the right direction regarding the objectives of the research conducted here.

The present theory stated a holistic framework in which student learning objectives were understood as an integral part of applicable operation in the course. In this connection, Dubinsky and Hazzan (2005) have developed a framework for teaching software development methods and also define the goals of the course within a holistic conceptual framework which identifies and defines roles for the stakeholders. They base their course on extreme programming (Beck, 1999), and the setting of the course, and its operational foundation, is thus different than that suggested by the present theory.

5.5 Summary

In this chapter, a theory on how to manage a one-semester real customer project model was presented (see Figure 16). This identifies an applicable way to ground the operation of the course. More specifically, the suggested way was based on an understanding of operational knowledge and then the epistemological positions of the project stakeholders in terms of that knowledge.

A meta-process was derived which illustrated how an applicable process in the course consists of a stable part and a changing part. This provides both a process skeleton that can be counted on (a stable part that fits) and sufficient space to empower the project stakeholders (changing part: what phases of the software life-cycle are stressed, does customer attend technical reviews, how requirements are estimated, what kind of testing is performed, etc.).

The present theory stated that the contentual focus of teaching should follow a single track in which students develop software for customers, so that the

or targets in students' work can be viewed from two viewpoints. The first was discussed in Section 2.8 at "Degree of realism": the students must notice the need for mature product (future development) despite the customer's immediate requests. The second relates to the present theory: academic emphases may become irrelevant in the working life-like teaching context.

learning objective is to consider the relevancies in operation. The student data indicated that `TIES405` has basically stressed both the end product and process deliverables, and this was what I observed when reviewing typical deliverables of a `TIES405` project in Section 2.6. Regarding the contentual focus of teaching, the present chapter suggests that a shift should be made from process deliverables (skill focus) to enhancing students' holistic process level understanding (reflection). Challenges were also identified, in particular inviting all the students to engage in process level considerations had not been sufficient with the action research effort. Thus, the shift away from the skill focus, towards reflectional learning, requires careful considerations on how to support reflection. I return to this challenge in Section 6.4.

With the given learning objective, it is highly important to know how to interact with the students. I referred to dynamic dialog and also stated that some authority is inevitable. The dynamic dialog should be in place because students develop expertise in the problem domain, and their views must therefore be considered when discussing the direction of the project. Moreover, in a real situation, operation is not based on fixed conceptualizations, but dynamism becomes inevitable. The teacher's authority is of an epistemological nature, and arises naturally from the teacher's possibility to observe successive projects and gain knowledge by reflection.

Finally, it is pointed out that the present chapter omits a case description (details). This is due to three things. Firstly, the goal of the work was theorizing as opposed to description. Secondly, the research theme that was "run into" here had a holistic nature as opposed to an interest in a single isolated question of student software development. Thirdly, the conceptualization (result) itself maintains that there is a changing part in the applicable software process, which means that a detailed description would not foreground the mechanisms of how to manage the course.

5.6 Methodological reflection

Addressing a real issue

Addressing real tensions and problems in people's lives itself demonstrates the validity of action research (Waterman, 1998). The action research of the dissertation addressed a real issue (instead of verifying a researcher-deducted preconception in practice). There was a call for a change due to tension in project completion and the associated problem of delayed projects, and the research conducted provides knowledge that the organization can utilize. Waterman (1998) suggests that the sources of tensions should be explored. I explored the problem only to the degree I felt appropriate: the principal aim was to foreground knowledge that helps improve things.

In claiming that a real issue was addressed, the change involved in the action research undertaking must also be discussed. The change is most appropriately discussed by referring to the contentual focus in teaching, as discussed in Section 5.6 and summarized above: the present theory suggested a shift away from a skill focus to reflectional learning, but this was noticed to be difficult, as any structured method for supporting student reflection was not adopted. I return here to Section 3.5, where it was stated that it is not clear what change constitutes an improvement. In this connection, teaching undertakings under the present action research process demonstrate that it is possible to satisfy the course constraints (real expectations and the fixed time scale), but the methods of how to support student reflection requires further considerations. Overall, I incorporated this kind of open-endedness into the theory of the present chapter through the sections "challenges encountered".

Critical validity

As noted in Section 3.5, one of the many lines of action research is that of emancipatory action research (critical theory). The validity of action research can also be viewed through the critical lens (Waterman, 1998). More precisely, action research aims to improve justice in people's lives; this demonstrates its validity (Waterman, 1998; Kemmis and McTaggart, 1988). The action research of the dissertation addressed an issue that closely relates to student rights.

In the context of critical validity, it is important to identify what the change has meant to the people involved (Waterman, 1998). The dissertation work put student groups in unequal positions as the approach taken with these groups differed from what other groups practiced. As students have experienced inequity due to the different approaches of project teachers independent of this research (as indicated by the data), this dissertation work inevitably injected more differences into the course practices. However, this work did not directly harm or overload students and aimed at foregrounding important knowledge.

Several discussions on student rights occurred with colleagues during the research process, particularly with regard to the limits of the course and contracting. I see all of this as about challenging the each other's occupational views. As noted in Section 3.6.3, such critical discussions helped in acknowledging personal bias.

Theoretical sensitivity

The second part of the dissertation is the result of continuous teacher reflection and reduction of that reflection to few conceptualizations (theory): see Figures 16 and 17, the teaching principle in Section 5.3.1, and the characterization of interaction in Section 5.3.2. The final interest was thus in theorizing and then in informing others of the understanding gained.

At many points I ended up using a comparison similar to GT. An illustrative example of this is Section 5.2, particularly the point at which I review the different project groups and conclude about the applicability of short iterations. In the similar manner, I could compare between an active and a less active teacher role. I also did systematic coding and note writing when I reviewed student experiences regarding the one-track principle in Section 5.3.1. This arose from the inclusion of an incomplete grounded theory process in the action research results. As stated in Chapter 3, the grounded theory-like comparison procedure intuitively entered the action research when theorizing the findings.

It should thus be appropriate to refer to validity in terms of grounded theory. Similarly to the grounded theory chapter (see Section 4.4), theoretical sensitivity is argued here as a validity aspect. Any theorizing process most likely involves some reduction to be able to put forward a consistent while explanatory outline of the theory. In this regard, the theory of this chapter was centered around a very basic and reduced conceptual idea, as illustrated in Figure 16 (cf. the core category in grounded theory).

Generalizability

As stated in Section 3.6, theory (conceptualization) becomes free from time, place, and people. In this connection, the theorizing aim of the present action research process is likely to increase generalizability and transferability: conceptualizations mean raising the level of abstraction and are easy to remember. Therefore, those planning or using the similar course constraints *may* benefit from the stated conceptualizations.

This *free from time, place, and people* is referred to from the methodological perspective here, thus in the sense of what kind of knowledge is developed: conceptualization vs. description. It is not assumed that (and it is not the interest of this dissertation to examine whether) the research result here is context-free, generalizable in the classic sense of the word, see (Lincoln and Guba, 1985, pp. 110–128), but can principally concern intensive real customer projects. This is in line with the grounded theory notion *substantive theory*, as discussed regarding the generalizability of the first results chapter (see Section 4.4)[20]. Only on the account of the messiness of social systems (Sayer, 2000), the dissertation neither claims that the stated operational approach is the only applicable approach with the studied course model.

Similar to the first results chapter of the dissertation, it must be noted here that the reflectional comparison between the local theory and the literature in Section 5.4 does not imply an attempt to force a generalization. Here also, the comparison arose from the on-going nature of the theory-oriented research and can advance the local theory in a hypothetical manner. To give an example, in Section 5.4 I further characterize the research results through a *hypothesis* which says that the course studied can be managed by viewing software development

[20] While grounded theory literature speaks of substantive grounded theories, the notion of "naturalistic generalization", which may associate with the "substantive theory", has been introduced in the literature by Stake (1978). As opposed to a grand generalization, naturalistic generalization is intuitive, empirical, and based on personal direct and vicarious experience; see discussion in (Lincoln and Guba, 1985, pp. 110–128).

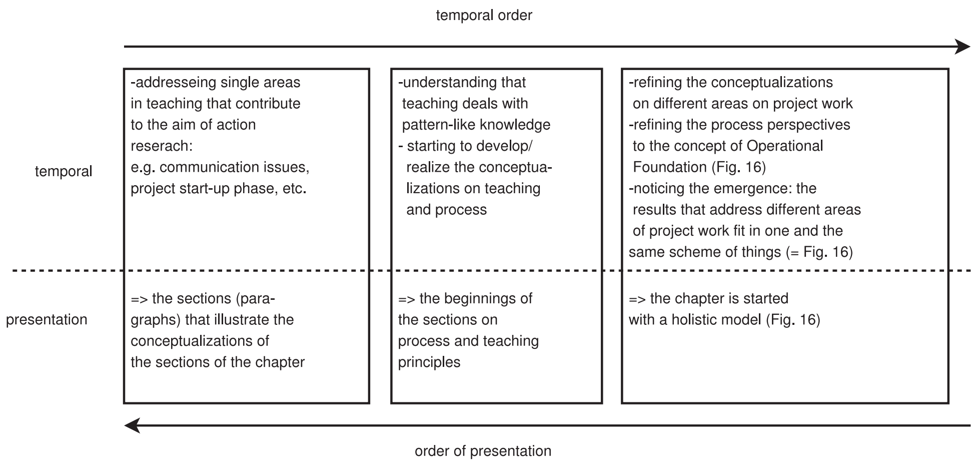
temporal order
temporal
-addresseing single areas in teaching that contribute to the aim of action reserach: e.g. communication issues, project start-up phase, etc.
-understanding that teaching deals with pattern-like knowledge
- starting to develop/ realize the conceptualizations on teaching and process
-refining the conceptualizations on different areas on project work
-refining the process perspectives to the concept of Operational Foundation (Fig. 16)
-noticing the emergence: the results that address different areas of project work fit in one and the same scheme of things (= Fig. 16)
presentation
=> the sections (paragraphs) that illustrate the conceptualizations of the sections of the chapter
=> the beginnings of the sections on process and teaching principles
=> the chapter is started with a holistic model (Fig. 16)
order of presentation

FIGURE 20 Temporal ordering (decisions and realization) vs. order of presentation

as a knowledge acquisition process.

Process of coming to know

Turnock and Gibson (2001) have examined observation-based action research and conclude that it may be a good idea to try to illustrate validity so that the reader can judge it. Clark (2000, p. 191) proposes that the researcher should present an analysis of the decisions made. To answer these validity issues it is noted that the results were given in a reflective manner and connected with teaching situations. For example, when a preference for short iterations was claimed, it was illustrated how such a conclusion was arrived at; when a changing part was proposed to be included in the software process, examples of the process tailoring were provided; when the teaching context was claimed to be sensitive, specific teaching situations were referred to. The process of coming to know was explicated within the results.

It was given in Section 3.6 that due to the theorizing aim, the temporal order of the present action research (the temporal order of the process of coming to know) is hardly foregrounded in the results. It is nevertheless possible to approximate how the layout of the results relates to the three larger scale phases given in Section 3.6. Namely, the results are given in reverse order compared to the temporal ordering of the research. This is illustrated in Figure 20. For example, the illustrative description of the occasion in Section 5.1.4 (which is a reflective research observation) later contributed to the realization of a meta-process that proposes that the applicable process has a changing part in it, and it is given to illustrate the holistic idea of the main conceptualization (Figure 16). I here follow Glaser's recommendations on theoretical writing (Glaser, 1978, Chapter on theoretical writing) so that conceptualizations come first and illustrative description follows.

Emergence

An interesting issue from the methodological viewpoint relates to the concept of emergence in the meaning familiar from grounded theory: how do concepts and theory emerge? Namely, I noticed that I had arrived at one and the same pattern whether I had examined the process approach, contentual focus of teaching, or interaction between teacher and students: the teacher has a specific position regarding reflection on relevant process and practices (see the outline of the present theory in Section 5.1). In line with this, the contentual focus of teaching is based on understanding the relevant practices that fit the context (see Section 5.3.1), and some kind of authority is inevitable in the interaction between the teacher and students (see Section 5.3.2). I therefore noticed that I can state and start with one holistic view of the studied course, as given in Figure 16. I assume this "parallel emergence" increases the validity of the results in terms of relevance.

Data collection by observation

The present action research did not directly affect the course in the sense that students would have consciously taken part in the research. My role in the ten projects supervised was clearly a teacher role, and reflective analysis of observations took place in the background. While I felt that this was appropriate for the research target to remain a teaching context, I would do things differently in retrospect. The afterthought is that continuous note writing closer to the teaching situations would have helped theorizing. As I have stated, technically the action research was iterative writing, but this was hampered without the on-going writing of analytic notes. Note writing would have helped in retaining a connection between thought and data.

I thus identify a validity concern which is the lack of continuous note writing. However, I would state that this diminishes because I became aware of it when performing the grounded theory research (Chapter 4), which underlines the importance of note writing, because the student data analyzed in that research indicated the same issues as the teacher observations (cf. triangulation), and because I finally stated the results so that I maintained a connection with empirical teaching situations (see "Process of coming to know" above). I also found it difficult to place myself as a researcher compared to the first part of the dissertation, in which I principally utilized data in the written form. Compared to the first part, it is not that straightforward to identify with the objective mindset given by comparative analysis (see Section 4.4), which may be due to the emancipatory characteristics of the present action research.

Role of the literature

The issues picked up in the second part of the dissertation principally arise from theorizing teaching experiences, thus not from the literature, although I consider

I have, for example, largely arrived at the view of a modern teacher[21]. I consider that first omitting the literature increases the validity of the study in the sense that I assume that I have intuitively extracted the most relevant issues from the process of teacher reflection, so that the result is very contextual, which is what this chapter was meant to be about: how to manage a one-semester real customer project model.

It was a conscious choice to present the analogy to objective knowledge already at the start of the present theory as I considered that this helped in communicating the holistic idea of the results. Similarly, I employed concepts from the educational literature, in particular when speaking of interaction in Section 5.3.2. This was because some of the issues that I ran into related to such general knowledge of modern teaching that I did not want to delay the citation to an isolated section (Section 5.4). This inclusion of the literature within the results is thus a presentational issue.

[21] Hacker and Niederhauser (2000) state that active participation, effective use of examples, collaborative problem solving, effective use of feedback, and motivational components underline the modern view of creating deep and durable learning as a teacher.

6 DISCUSSION AND CONCLUSIONS

This dissertation theorized a one-semester real customer student project model. The first results chapter posited a grounded theory that reveals a mechanism that played a role in students' overall experience of the studied course. The second results chapter identified an operational approach that can be used with the studied course model. In addition, there was a strong focus on theory development.

In the following sections I briefly discuss the results of the dissertation and state my conclusions. Section 6.1 summarizes the results chapters in the light of the attribute of realism in education. Section 6.2 returns to the problem of delayed projects. Section 6.3 argues why it is important to identify an operational approach that can be used with the studied course model. Section 6.4 illustrates the on-going nature of present non-positivist theorizing-oriented research: as supporting each student's reflective (conceptual) learning was found challenging, a word on group work from the viewpoint of individual learning is given to indicate how this line of research could be furthered in the future. The contribution of the dissertation is discussed in Section 6.5 and concluding remarks are given in Section 6.6.

6.1 Realism

`TIES405` has generally been very well received by the students. Solely on the basis of the overall positiveness of student experiences, reported in Chapter 4, the dissertation suggests that realism is an important attribute in education with respect to both the enjoyability of learning and the learner's growth. The frequent presence of realism in a curriculum is a good starting point both to drive students' interest and to disclose problems. The latter means here that when students are sufficiently frequently exposed to realism their response is perhaps not colored by enthusiasm about realism.

Putting realism into practice requires effort. In particular, I hope that the mechanism underlying the positive student overall course experience — proposed in Chapter 4 — helps to critically evaluate such efforts. Moreover, I hope that the second part of the dissertation provides a good illustration of the effort needed (on-going course development work with conceptualization) when embedding realism in an educational context. I could formulate something like this:

> *You wouldn't leave a child to cook by him/herself with an electric oven as it is likely that the child would burn him/herself. But you can cook together and then the child can use the oven, being overseen by someone who takes the responsibility of the child's authentic use of the oven.*

The major message and conclusion of the dissertation follows the same idea: **realism must be managed.**

6.2 Problem of delayed projects

Having stated that one must be careful in how to embed realism in education, I return to the problem of delayed projects. This is first discussed from a critical perspective and then in relation to the data sources used and the aim of theorizing.

Chapter 4 presented a grounded theory which states that the positive impact of realism endures in students' experiences irrespective of the problems that arise during a project. The underlying reason is that inexperienced students who are in need of realism tolerate problems when fed with realism, which in turn explains why their overall course experiences tend to turn positive. However, the overall positiveness of students' experiences does not justify actions that are risky from the perspective of students' rights. Overall positiveness is found in the students' experiences in delayed projects, but this positiveness does not justify using students as free labor. Students' inexperience has to be acknowledged, as indicated by the student data and also by the curricular position of the course (no prior projects). Due to this inexperience, students are not likely to speak up for themselves in problematic situations. An indication of this silence is also the observation that the students' criticism of the course was observed to be more hard-hitting in the anonymous questionnaire data than in the public project reports. Note also that for most of the attendees the course has been mandatory. This discussion refers back to the emancipatory starting point of this dissertation and underlines the tenet that **students' "silence" due to their inexperience and obligation does not justify ethically questionable acts in realistic project courses.**

The above discussion can be continued in the context of Parlett's (1977) work. His discussion concerns how an academic environment (e.g. a department) inevitably creates certain norms (a "background factor") that may have a strong and lasting effect on students' perceptions. He states that with educational inquiries that are concurrent rather than retrospective, there is the difficulty of students' views at the time not being their final ones; those sought long after the event may deserve greater notice. He also points to the difficulty that students may be responding to the milieu in ways they may not wish to disclose; and of which at the time they may be only partially aware. In the light of these assertions, it could be asked whether the local motivation of the dissertation is justified: is the view of a project delay as a problem anchored in student data that represents the students' final views?

In Section 3.6 I pointed out that the very nature of a customer project and the large number of process deliverables and expectations of them are likely to cause tension in project completion. The latter was discussed in the results (Chapter 5) through the concept "second track". There, data quotes were provided from public project reports (concurrent data) and from retrospectively collected anonymous questionnaire data. This partially answers the concerns raised by Parlett above. Moreover, the whole of Chapter 4 is basically consistent with Parlett's assertions, as the grounded theory of that chapter calls for careful analyses on student feedback although a positive overall response to the course is in place (see discussion in Section 4.4 under the title "Relevance to those involved"). The grounded theory suggests that the students' responses are connected to their inexperience. In this light, I see that discussion of project delays is well initiated through a research undertaking.

Moreover, I do not see that focusing on students' project-time responses limits theorizing. Theories in the sense of this dissertation are abstract representations that can be modified (as opposed to fixed generalizations). Re-gathering student data, for example, ten years after project completion, would enable the inclusion of a timeline in relation to the students' perceptions in the theory, but would not reduce the theorizing interest on the basis of the students' initial perceptions.

6.3 Why an operational foundation must be in place

The second results chapter started with a holistic operational idea found to be applicable in the course of the research. This had to do with understanding the nature of applicable operational knowledge within the studied course model. This section discusses why this is considered important.

Understanding the nature of operational knowledge, how it is acquired and delivered, is crucial with a one-semester real customer project model. If an appropriate contract is used in such a course, a teacher can and should refer to the contract when necessary and terminate projects on time. However, this is not the only answer to the task of finding a successful arrangement for such a course model, as it is unlikely that paying customers will continue to engage in student projects if their outcome is nothing more than a set of unfinished process deliverables. For this reason a basic understanding of the operational approach, which directs projects to efficient operation, but also explicates the students' and the teachers' epistemological positions in relation to the students' learning, is required.

On the other hand, one could argue that the project topics need to be stationary and down-sized to remove the tension in project completion. As discussed earlier in the dissertation, working with a truly stationary topic with a real customer arrangement is problematic due to the very nature of software development. Down-sizing means moving towards pre-packaging, which is likely to reduce the challenge perceived by students, and thus also their motivation (the challenge-is-motivation association was foregrounded in Chapter 4). From this perspective also, it is the operational approach should receive attention in course contexts of this kind.

One could also argue that projects can always be continued with another group in the next semester, as a way of removing the tension in completion. I return here to Chapter 2. I stated there that the project outcome depends on the nature of the project, and gave an example of how sometimes, when the project stakeholders notice the actual size of the project after project commencement, a project is continued during the following semester with a new student group. This is also the option given by Forner et al. (2007): a new group continues the project. However, in `TIES405`, the customer may have the expectation that the project should be doable in one semester, because the project proposal was accepted by the departmental committee in the first place. Small companies may not be willing to pay for work where a new group has to learn the problem domain, extending delivery by half a year. Thus, while assigning unfinished project work to a new group is one option when a project must be continued, it is certainly not a guaranteed option, as customers are of different size, have different schedules for the product and collaboration resources, and come with a varying degree of need. Furthermore, in the light of the student experiences reported in Chapter 4, i.e., students' enjoyment and feeling of survival due to completion in an authentic context, it seems valuable to provide students with stand-alone projects instead of turning the project into an on-going project during the semester. From this perspective also, the operational approach stated in Chapter 5 is important.

6.4 A word on group work

It was found that student learning in a group is non-trivial (see Chapter 4) and that supporting student learning in an intensive group work context is challenging (see Sections 5.3.1 and 5.3.2). I concluded, among the results, that student learning could have been better addressed with frequent and structured teacher interventions, that is, by introducing focus into the students' reflection. In the following I discuss how group work could be better taught in a context such as `TIES405`, and at the same time illustrate the on-going nature of the present research.

Those parts of the grounded theory in Chapter 4 that relate to the learning possibilities of individuals in the context of group work could be formulated as self-evaluation questions. The grounded theory proposed that students' positive experiences are conditioned by support, and in respect of group work this meant that students must both attend actively and involve their group mates actively. The grounded theory also proposed that students' negative experiences tend to be associated with feelings of unfairness. In these connections, students could be asked throughout their projects (e.g. every other week) whether they are attending actively, whether they are involving their team mates actively, and how they see their own performance from the viewpoint of justice in group work. Here the concepts of *supportiveness* and *justice* would be used as a teaching tool. This would give students explicit ideas about how to view group work in relation to the learning possibilities of individuals, and give them a chance for self-regulation during the projects.

Such self-evaluations would also help in studying how much individuals' learning objectives can be addressed in the presence of real customer expectations. Meeting individuals' learning objectives is one of the most difficult challenges facing group work under real customer expectations: as the grounded theory in Chapter 4 foregrounded, students tend to divide work according to their individual strengths. On the other hand, the data indicated that using a small group size is likely to mitigate this problem. Moreover, shared reflection, which was proposed as the principal way of learning in the theory presented in Chapter 5, is something all the students can do whatever their role in a group. While these, small team size and the learning through reflection, may help in providing a meaningful experience for all the students in a group, the challenge of meeting personal learning objectives remains and merits study.

Overall, these future directions on teaching in the context of group work underline the importance of supporting students' meaning making when they are engaged in intensive project work. It can easily happen that a realistic course gives students a chance to accumulate tacit knowledge through a realistic experience, without making more explicit what is being taught (see Section 5.4 under the heading "Learning objectives"). To help remedy this situation, ways of introducing focus into students' reflection need to be found, and this would concern not only group work. This goal can be characterized as controlling the loss of meaning in educational communication, as discussed by Roos and Hamilton (2004).

I also conclude that introducing students to realism does not always need to be realized in the form of group work. The ability to work independently is a valuable working life skill, and based on the challenges facing group work, encountered in the course of this dissertation, it makes sense that students have also been introduced to realism in individual projects in computer science education (see Section 2.8).

6.5 Contribution

In the introduction to the dissertation it was stated that a research field with a long history presents the researcher with a challenge with respect to making a genuine contribution. The research approach taken here evolved into theorizing, the motivation for which was the possibility of finding fresh viewpoints in, and giving structure to, the long-established research field, in which studies are often realized in the form of description. Overall, both the results chapters of the dissertation arrived at a holistic mechanism. In Chapter 4 this explains students' course experiences and in Chapter 5 how actions took place. This collection of experiences and the conducting of the research on the basis of critical realism form the central contribution of the dissertation. Critical realism states that the explanation of social events, processes, and states of affairs requires an account of the real mechanisms that bring them about (Nash, 2005). I would summarize the results of the dissertation by considering that while many of the findings are described or touched on in the literature, it is the integrity and the conceptual level of the analysis that contributes most to the field. Thus I view the results as having a close relation to the research approach, i.e., theorizing on the grounds of critical realism.

I will give examples starting from Chapter 4. It is repeatedly reported in the literature that students working on a realistic project course enjoy it and think that their project experience has been useful, e.g. (Clark, 2005; Newman et al., 2003; Melin and Cronholm, 2004). If such extracts are conceptualized into two dimensions, run-time (they enjoy) and future-oriented (useful), as is done within the theory in Chapter 4, one can start saying that the run-time and future-oriented dimensions are in place and move on. In a similar manner, it is repeatedly reported in the literature that project students need support, e.g. (Davis, 2000; Laware and Walters, 2004). In this dissertation, this subject matter is formulated into a theoretical proposition: students' positive experiences are conditioned by support, the form of which depends on the topic of the project work. Further, it can be frequently observed in the literature how students summarize their project experience as the best course they have taken, e.g. (Brown, 2000). My position is that it is valuable to come to know why this is the case. Thus I am pointing to the real mechanisms that underlie such strikingly similar responses; for example the mechanism of the theory presented in Chapter 4: enduring value of realism — boosted by inexperience, as I briefly refer to it.

Similar illustrations can be drawn from Chapter 5. For instance, it is reported in the literature that iterative and incremental development means less documentation and can thereby reduce the workload of students (Brown, 2000). In Chapter 5, I arrive at a higher level of abstraction, by pointing to the inductiveness of the operational processes and to the teaching challenge with an analogy to the objective nature of knowledge. Within the same framework, I further refine that operation has a stable part, and that due to variation in the course context, it also has an unstable part. With respect to the stable part, I refer to iterative development.

In my view, it is this level of abstraction (conceptual level) and integrity that is valuable. It is valuable to conceptualize and have lower level conceptualizations integrated into a single conceptual scheme, as illustrated at the beginnings of the results chapters. To explain this, I will repeat the quote from Glaser (1978) I gave in Chapter 3, where I argued for theorizing in the action-taking context: *Men in the know easily become locked-in or status quo-oriented, as their knowledge becomes stable, consistent and consolidate their position. With theory their perceptions are more amenable to change since they can begin to see the processes making for change and can modify their ideas to handle the new knowledge. They begin to transcend what was seen as inviolate, by seeing precious happenings as merely elements of patterns in process. They can work with familiar occasions purposefully.*

I feel it necessary to point out that how one sees the contribution of this dissertation may depend on the lens one uses, that is, whether one notices the presence of middle-ground theory and sees value in theory that touches many of the issues which are repeatedly found in the literature. This means viewing the results as theory instead of "a set of findings". The grounded theory debate in the literature demonstrates how emergence-based theory development is sometimes subordinated to a traditional research approach, such as the one described by Jenkins (1985): idea ⇒ library research ⇒ research topic ⇒ research strategy ⇒ experimental design ⇒ data capture ⇒ data analysis ⇒ publish results. Suddaby (2006), for example, states that *the idea that reasonable research can be conducted without a clear research question and absent theory simply defies logic*. He also states that *in pure form, grounded theory research would be presented as a jumble of literature consultation, data collection, and analysis conducted in ongoing iterations that produce many relatively fuzzy categories that, over time, reduce to fewer, clearer conceptual structures. Theory would be presented last.* Suddaby directs his critique towards grounded theorists who take their method as an excuse to ignore literature. In my view, these assertions point to the traditional research process described, for example, by Jenkins and conflict with the idea of emergence. These assertions conflict with my experience that latent patterns resulting from theory development can be implicitly present in the literature but not identifiable until they emerge for a researcher in a local context. Assertions such as those stated by Suddaby are an interesting point in the grounded theory literature and make me conclude that seeing the contribution made by this dissertation may depend on the lens.

As further contribution of this work, I would also note the presence of a critical voice from inside academia, when I refer to student rights; see for example Sections 6.1 and 6.2. Moreover, it is noted that the holistic research approach adopted in the dissertation gives rise to many themes that with further work can become stand-alone research results. For example, in Section 5.3.1 it is pointed out that when inexperienced students are given wrong practices (ones that do not fit the context), they do not necessarily identify this as a cause of problems, and may think that they are gaining a realistic view of a software project. This can be further analyzed in connection with course assessment, i.e., what if students in such situations are assessed at least partly on the basis of achievement? This constitutes an unfortunate latent pattern in project education, the disclosing of which is about emancipation.

6.6 Concluding remarks

Based on the present research, I regard the one-semester real customer project model as a feasible educational arrangement, even if it is prone to serious problems. It is a stressful teaching context, but the challenging situations that arise from close contact with students and using real customers drive the teachers' professional learning and thus also organizational learning. The dissertation proposes theory that helps teachers to cope with their daily practice. The dissertation is inevitably a snapshot of the understanding yielded by the act of theorizing in a specific project teaching context.

REFERENCES

Abrahamsson, P., Warsta, J., Siponen, M. T. & Ronkainen, J. 2003. New directions on agile methods: A comparative analysis. In ICSE ’03: Proceedings of the 25th International Conference on Software Engineering. Washington, DC: IEEE Computer Society, 244–254.

Adolph, S., Hall, W. & Kruchten, P. 2008. A methodological leg to stand on: Lessons learned using grounded theory to study software development. In CASCON ’08: Proceedings of the 2008 conference of the center for advanced studies on collaborative research. New York, NY: ACM, 166–178.

Allan, G. 2003. A critique of using grounded theory as a research method. Electronic Journal of Business Research Methods 2 (1), 1–10.

Almstrum, V. L. & Last, M. Z. 2005. What attracts women to CS? SIGCSE Bull. 37 (3), 378–378.

Almstrum, V. L. & Last, M. Z. 2006. Men are from toys: women are from tools. In ITICSE ’06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education. New York, NY: ACM, 313–313.

Annells, M. 1996. Grounded theory method: Philosophical perspectives, paradigm of inquiry, and postmodernism. Qualitative Health Research 6 (3), 379–393.

Armour, P. G. 2003. The Laws of the Software Process: A New Model for the Production and Management of Software. Boca Raton, Florida: Auerbach Publications.

Atchison, W. F., Conte, S. D., Hamblen, J. W., Hull, T. E., Keenan, T. A., Kehl, W. B., McCluskey, E. J., Navarro, S. O., Rheinboldt, W. C., Schweppe, E. J., Viavant, W. & Young, J. D. M. 1968. Curriculum 68: Recommendations for academic programs in computer science: A report of the ACM curriculum committee on computer science. Commun. ACM 11 (3), 151–197.

Aygün, B. 2004. A platform for software engineering course projects. Turkish Journal of Electrical Engineering 12 (2), 107–116.

Basili, V. R. & Turner, A. J. 1975. Iterative enhancement: A practical technique for software development. IEEE Trans. Software Eng. 1 (4), 390-396.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. 2001. Manifesto for Agile Software Development. Retrieved May 29, 2007, from http://www.agilemanifesto.org/.

Beck, K. 1999. Extreme Programming Explained: Embrace Change. (First edition) Boston: Addison-Wesley Professional, 224.

Begel, A. & Simon, B. 2008. Novice software developers, all over again. In ICER '08: Proceeding of the fourth international workshop on Computing education research. New York, NY: ACM, 3–14.

Ben-Ari, M. 2001. Constructivism in computer science education. Journal of Computers in Mathematics and Science Teaching 20 (1), 45–73.

Bern, A., Pasi, S. J. A., Nikula, U. & Smolander, K. 2007. Contextual factors affecting the software development process—an initial view. In The Second AIS SIGSAND: European Symposium on Systems Analysis and Design. Gdansk, Poland: University of Gdansk Press.

Beynon, M., Myers, R. & Harfield, A. 2009. Web Eden: Support for computing as construction. In A. Pears & C. Schulte (Eds.) 9th Koli Calling: International Conference on Computing Education Research. Uppsala, Sweden: Uppsala University, 54–57.

Bhaskar, R. 1978. A Realist Theory of Science. (Second edition) Sussex, UK: Harvester Press.

Blum, F. H. 1955. Action research–A scientific approach? Philosophy of Science 22 (1), 1–7.

Borgen, W., Pollard, D., Amundson, N. E. & Westwood, M. 1989. Employment Groups: The Counselling Connections. Toronto: Lugus.

Bothe, K. 2001. Reverse engineering: The challenge of large-scale real-world educational projects. In Software Engineering Education and Training. Proceedings. 14th Conference on. Los Alamitos, CA: IEEE Computer Society, 115-126.

Bridgeman, N. 2008. Capstone projects — an NACCQ retrospective. In S. Mann & M. Lopez (Eds.) NACCQ '08: 21st Annual Conference of the National Advisory Committee on Computing Qualifications. Auckland, New Zealand: NACCQ, 195–199.

Brooks, F. P. 1987. No silver bullet: Essence and accidents of software engineering. Computer 20 (4), 10–19.

Brooks, F. P. 1995. The Mythical Man-Month. (Anniversary edition) Boston, MA: Addison-Wesley.

Brown, J. 2000. Bloodshot eyes: Workload issues in computer science project courses. In Software Engineering Conference. APSEC 2000. Proceedings. Seventh Asia-Pacific. IEEE Computer Society, 46–52.

Bruner, J. S. 1977. Early social interaction and language development. In H. R. Schaffer (Ed.) Studies in Mother-Child Interaction. London: Academic Press, 271–289.

Bruner, J. 1961. The act of discovery. Harvard Educational Review 31 (1), 21–32.

Brügge, B. 1994. From toy systems to real systems: Improvements in software engineering education. In SEUH, Vol. 43. Stuttgart: Teubner Verlag. Workshop of the German Chapter of the ACM, 62–72.

Burge, J. & Gannod, G. 2009. Dimensions for categorizing capstone projects. In Software Engineering Education and Training. Proceedings. 22nd Conference on. Los Alamitos, CA: IEEE Computer Society, 166–173.

CMMI Product Team 2002. CMMI for Systems Engineering, Software Engineering, Integrated. Product and Process Development, and Supplier Sourcing. Software Engineering Institute, Carnegie Mellon University. Technical report CMU/SEI-2002-TR-011.

Candy, P. C. 1991. Self-Direction for Life-Long Learning: A Comprehensive Guide to Theory and Practice. San Franscisco, CA: Jossey-Bass.

Carr, W. & Kemmis, S. 1986. Becomming Critical: Education, Knowledge and Action Research. London: The Falmer Press.

Chase, J. D., Oakes, E. & Ramsey, S. 2007. Using live projects without pain: The development of the small project support center at radford university. SIGCSE Bull. 39, 469–473.

Chen, T.-Y., Lewandowski, G., McCartney, R., Sanders, K. & Simon, B. 2007. Commonsense computing: Using student sorting abilities to improve instruction. SIGCSE Bull. 39 (1), 276–280.

Cheng, C. S. & Beaumont, C. 2004. Evaluating the effectiveness of ICT to support globally distributed PBL teams. In ITiCSE ’04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education. New York, NY: ACM, 47–51.

Chinn, D., Spencer, C. & Martin, K. 2007. Problem solving and student performance in data structures and algorithms. In ITiCSE ’07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education. New York, NY: ACM, 241–245.

Christensen, K. & Rundus, D. 2003. The capstone senior design course: An initiative in partnering with industry. In ASEE/IEEE Frontiers in Education, 33rd Annual. IEEE, S2B - 12-17 vol.3.

Clark, J. E. 2000. Action research. In D. Cormack (Ed.) The Research Process in Nursing. (Fourth edition) Oxford: Blackwell Publishing, 183–197.

Clark, N. 2005. Evaluating student teams developing unique industry projects. In ACE ’05: Proceedings of the 7th Australasian conference on Computing education. Darlinghurst, Australia: Australian Computer Society, 21–30.

Clear, T., Goldweber, M., Young, F. H., Leidig, P. M. & Scott, K. 2001. Resources for instructors of capstone courses in computing. In ITiCSE-WGR '01: Working group reports from ITiCSE on Innovation and technology in computer science education. New York, NY: ACM, 93–113.

Coleman, G. & O'Connor, R. 2008. Investigating software process in practice: A grounded theory perspective. Journal of Systems and Software 81 (5), 772 - 784.

Collofello, J. & Ng, C. H. 1999. Assessing the process maturity utilized in software engineering team project courses. In ASEE/IEEE Frontiers in Education Conference, 29th Annual. IEEE, 12A9/5-12A9/9 vol.1.

Coplien, J. O. & Devos, M. 2000. Architecture as metaphor. In Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics. Orlando, Florida: Institute of Informatics and Systemics, 737–742.

Coppit, D. & Haddox-Schatz, J. M. 2005. Large team projects in software engineering courses. In SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education. New York, NY: ACM, 137–141.

Dahlin, B. 2001. Critique of the schema concept. Scandinavian Journal of Educational Research 45 (3), 287–300.

Daniels, M. & Asplund, L. 1999. Full scale industrial project work, a one semester course. In ASEE/IEEE Frontiers in Education Conference, 29th Annual. IEEE, 11B2/7-11B2/9.

Davis, M. C. 2000. A student's perspective of a capstone course. In CCSC '00: Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference. USA: Consortium for Computing Sciences in Colleges, 151–167.

Deibel, K. 2007. Studying our inclusive practices: Course experiences of students with disabilities. SIGCSE Bull. 39 (3), 266–270.

Deibel, K. 2008. Course experiences of computing students with disabilities: Four case studies. In SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education. New York, NY: ACM, 454–458.

Denzin, N. 1978. The Research Act: A Theoretical Introduction to Sociological Methods. New York, NY: McGraw-Hill.

Dick, B., Stringer, E. & Huxham, C. 2009. Theory in action research. Action Research 7 (1), 5–12.

Dick, M., Sheard, J., Bareiss, C., Carter, J., Joyce, D., Harding, T. & Laxer, C. 2003. Addressing student cheating: Definitions and solutions. SIGCSE Bull. 35 (2), 172–184.

Dubinsky, Y. & Hazzan, O. 2003. eXtreme programming as a framework for student-project coaching in computer science capstone courses. In Software: Science, Technology and Engineering. Proceedings. International Conference on. IEEE, 53–59.

Dubinsky, Y. & Hazzan, O. 2002. Improvement of software quality: Introducing extreme programming into a project-based course. In 14th International Conference of the Israel Society for Quality. Jerusalem, Israel: , 8-12.

Dubinsky, Y. & Hazzan, O. 2005. A framework for teaching software development methods. Computer Science Education 15 (4), 275-296.

Eden, C. & Huxham, C. 1996. Action research for management research. British Journal of Management 7, 75–86.

Elliot, J. 1976. Developing hypotheses about classroom from teacher's practical constructs: An account of the work of the Ford Teaching Project. In S. Kemmis & R. McTaggart (Eds.) The Action Research Reader. (Third edition) Geelong: Deakin University Press, 915–213. Reprinted from Elliot, J. (1976), Developing Hypotheses about Classroom from Teacher's Practical Constructs: An Account of the Work of the Ford Teaching Project. Interchange, 7(2), 2–22.

Ellis, J. & Kiely, J. 2000. Action inquiry strategies: Taking stock and moving forward. Journal of Applied Management Studies 9 (1), 83–94.

Feldman, D. H. 1980. Beyond Universals in Cognitive Development. Norwood, New Jersey: Ablex.

Fincher, S., Petre, M. & Clark, M. (Eds.) 2001. Computer Science Project Work: Principles and Pragmatics. London: Springer-Verlag.

Fitzgerald, S., Simon, B. & Thomas, L. 2005. Strategies that students use to trace code: an analysis based in grounded theory. In ICER '05: Proceedings of the first international workshop on Computing education research. New York, NY: ACM, 69–80.

Flavell, J. H. 1976. Metacognitive aspects of problem solving. In L. B. Resnick (Ed.) The nature of Intelligence. Hillsdale, NJ: Erlbaum, 231–236.

Fornaro, R. J., Heil, M. R. & Tharp, A. L. 2007. Reflections on 10 years of sponsored senior design projects: Students win-clients win! Journal of Systems and Software 80 (8), 1209 - 1216.

Freeman, P. 1976. Realism, style, and design: Packing it into a constrained course. In Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education. New York, NY: ACM, 150–157.

Friedman, V. J. & Rogers, T. 2009. There is nothing so theoretical as good action research. Action Research 7 (1), 31–47.

Gal-Ezer, J. & Zeldes, A. 2000. Teaching software designing skills. Computer Science Education 10 (1), 25–38.

Genat, B. 2009. Building emergent situated knowledges in participatory action research. Action Research 7 (1), 101–115.

Gibbs, G., Morgan, A. & Taylor, L. 1980. Understanding why students don't learn. In Study Methods Group Report No. 5. Milton Keynes, England: Institute of Educational Technology, Open University.

Glaser, B. G. 1978. Theoretical Sensitivity: Advances in the Methodology of Grounded Theory. San Francisco, CA: Sociology Press.

Glaser, B. G. & Strauss, A. L. 1964. The social loss of dying patients. The American Journal of Nursing 64 (6), 119–121.

Glaser, B. G. & Strauss, A. L. 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research. New York, NY: Aldine de Gruyter.

Glaser, B. G. 1992. Emergence vs. Forcing: Basics of Grounded Theory Analysis. Mill Valley, USA: Sociology Press.

Glaser, B. G. 2002a. Conceptualization: On theory and theorizing using grounded theory. International Journal of Qualitative Methods 1 (2).

Glaser, B. G. 2002b. Constructivist grounded theory? [47 paragraphs]. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research [On-line Journal] 3 (3).

Glaser, B. G. 2004. Remodeling grounded theory. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research [On-line Journal] 5 (2). With the assistance of Judith Holton.

Gorschek, T. 2006. Requirements Engineering Supporting Technical Product Management. Department of Systems and Software Engineering, Blekinge Institute of Technology. Ph. D. Thesis.

Goulding, T. & DiTrolio, R. 2005. Incorporating realistic constraints into a student team software project. SIGCSE Bull. 37 (4), 54–58.

Grant, M. M. 2002. Getting a grip on project-based learning: Theory, cases and recommendations. Meridian: A Middle School Computer Technologies Journal 5 (1).

Green, L. 2003. Projecting IT education into the real world. In CITC4 '03: Proceedings of the 4th conference on Information technology curriculum. New York, NY: ACM, 111–114.

Grundy, S. 1990. Three models of action research. In S. Kemmis & R. McTaggart (Eds.) The Action Research Reader. (Third edition) Geelong: Deakin University Press, 353–364. Reprinted from Grundy, S. (1982), Three Models of Action Research. Curriculum Perspectives, 2(3), 23–34.

Guba, E. G. & Lincoln, Y. S. 1994. Competing paradigms in qualitative research. In N. K. Denzin & Y. S. Lincoln (Eds.) Handbook of Qualitative Research. Thousand Oaks, CA: Sage, 105–117.

Habermas, J. 1979. Communication and the Evolution of Society (Translated by Thomas McCarthy). Boston: Beacon Press.

Hacker, D. J. & Niederhauser, D. S. 2000. Promoting deep and durable learning in the online classroom. New Directions for Teaching and Learning 2000 (84), 53-63.

Hadfield, S. M. & Jensen, N. A. 2007. Crafting a software engineering capstone project course. J. Comput. Small Coll. 23 (1), 190–197.

Hagan, D., Tucker, S. & Ceddia, J. 1999. Industrial experience projects: A balance of process and product. Computer Science Education 9 (3), 215 - 229.

Haig, B. 1995. Grounded theory as scientific method. Philosophy of education (on-line). `http://jan.ucc.nau.edu/~pms/cj355/readings/Haig%20Grounded%20Theory%20as%20Scientific%20Method.pdf` Retrieved on March 2011.

Hanks, B., Murphy, L., Simon, B., McCauley, R. & Zander, C. 2009. CS1 students speak: Advice for students by students. In SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education. New York, NY: ACM, 19–23.

Hansen, B. H. & Kautz, K. 2005. Grounded theory applied - studying information systems development methodologies in practice. In HICSS '05: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 8. Washington, DC: IEEE Computer Society, 264.2.

Hart, M. 1990. Critical theory and beyond: Further perspectives on emancipatory education. Adult Education Quarterly 40 (3), 125–138.

Hedin, G., Bendix, L. & Magnusson, B. 2005. Teaching extreme programming to large groups of students. Journal of Systems and Software 74 (2), 133-146.

Herman, G. L., Kaczmarczyk, L., Loui, M. C. & Zilles, C. 2008. Proof by incomplete enumeration and other logical misconceptions. In ICER '08: Proceeding of the fourth international workshop on Computing education research. New York, NY: ACM, 59–70.

Herr, K. & Anderson, G. L. 2005. The Action Research Dissertation: A Guide for Students and Faculty. Thousand Oaks, CA: Sage publications.

Hewner, M. & Guzdial, M. 2008. Attitudes about computing in postsecondary graduates. In ICER '08: Proceeding of the fourth international workshop on Computing education research. New York, NY: ACM, 71–78.

Hiltunen, L. 2010. Enhancing web course design using action research. In Jyväskylä Studies in Computing, Vol. 125. Jyväskylä, Finland: University of Jyväskylä.

Ho, C.-W., Slaten, K., Williams, L. & Berenson, S. 2004. Work in progress—unexpected student outcome from collaborative agile software development practices and paired programming in a software engineering course. In ASEE/IEEE Frontiers in Education, 34th Annual. IEEE, F2C-15-16 Vol. 2.

Huggins, J. K. 2009. Engaging computer science students through cooperative education. SIGCSE Bull. 41 (4), 90–94.

Hughes, J. & Jones, S. 2003. Reflections on the use of grounded theory in interpretive information systems research. In Proceedings of the ECIS'2003 Conference, Naples, Italy.

Humphrey, W. S. 1995. A Discipline for Software Engineering. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Huxham, C. & Vangen, S. 2003. Researching organizational practice through action research: Case studies and design choices. Organizational Research Methods 6 (3), 383–403.

Huxham, C. 2003. Action research as a methodology for theory development. Policy & Politics 31 (2), 239–248.

IEEE 1990. IEEE standard glossary of software engineering terminology. IEEE Std 610.12-1990.

ISO/IEC 2002. ISO/IEC 15288:2002 Systems Engineering – Systems Life Cycle Processes.

Isomöttönen, V. & Kärkkäinen, T. 2009. Communicating with customers in student projects: Experimenting with grounded theory. In A. Pears & C. Schulte (Eds.) 9th Koli Calling: International Conference on Computing Education Research. Uppsala, Sweden: Uppsala University, 84–93.

Isomöttönen, V., Korhonen, V. & Kärkkäinen, T. 2007. Power-of-recognition: A conceptual framework for capstone project in academic environment. In G. Concas, E. Damiani, M. Scotto & G. Succi (Eds.) 8th International Conference on Agile Processes in Software Engineering and Extreme Programming, Vol. 4536. Berlin Heidelberg: Springer-Verlag. LNCS, 145-148.

Isomöttönen, V. & Kärkkäinen, T. 2008. The value of a real customer in a capstone project. In Software Engineering Education and Training. Proceedings. 21st Conference on. Los Alamitos, CA: IEEE Computer Society, 85–92.

Isomöttönen, V. 2006. Projektiopintojen perusteet. Department of Mathematical Information Technology, University of Jyväskylä, Finland. (In Finnish).

Jadud, M. C. 2006. Methods and tools for exploring novice compilation behaviour. In ICER ’06: Proceedings of the second international workshop on Computing education research. New York, NY: ACM, 73–84.

James, R. H. 2005. External sponsored projects: Lessons learned. SIGCSE Bull. 37 (2), 94–98.

James, W. 1996. Essays in Radical Empiricism. Bison Books.

Jenkins, A. M. 1985. Research methodologies and MIS research. In E. Mumford, R. Hirschheim, G. Fitzgerald & A. T. Wood-Harper (Eds.) Research methods in information systems. Amsterdam, Holland: North-Holland Publishing, 97–103.

Johansson, C. & Molin, P. 1996. Maturity, motivation and effective learning in projects—benefits from using industrial clients. In SEHE ’95: Proceedings of the second international conference on Software engineering in higher education II. Billerica, MA: Computational Mechanics, 99–106.

Johnson, B. 1993. Teacher-as-researcher. ERIC Digests (online). Retrieved on February 2011.

Jormanainen, I., Harfield, A. & Sutinen, E. 2009. Supporting teacher intervention in unpredictable learning environments. In ICALT ’09: Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies. Washington, DC: IEEE Computer Society, 584–588.

Judith, W. C., Bair, B., Börstler, J., Timothy, L. C. & Surendran, K. 2003. Client sponsored projects in software engineering courses. In SIGCSE ’03: Proceedings of the 34th SIGCSE technical symposium on Computer science education. New York, NY: ACM, 401–402.

Järvinen, A., Koivisto, T. & Poikela, E. 2000. Oppiminen työssä ja työyhteisössä. Helsinki: Werner Söderström. (In Finnish).

Kautz, K. & Pries-Heje, J. 1999. Systems development education and methodology adoption. SIGCPR Comput. Pers. 20 (3), 6–26.

Keefe, K. & Dick, M. 2004. Using extreme programming in a capstone project. In R. Lister & A. Young (Eds.) Sixth Australasian Computing Education Conference, Vol. 30. Australian Computer Society. CRPIT, 151–160.

Kemmis, S. & McTaggart, R. 1988. The Action Research Planner. (Third edition) Victoria, Australia: Deakin University Press.

Kemmis, S. 2001. Exploring the relevance of critical theory for action research: Emancipatory action research in the footsteps of jürgen habermas. In P. Reason & H. Bradbury (Eds.) Handbook of Action Research: Participative Inquiry & Practice. London: SAGE Publications, 91–102.

Kitchenham, B. 2004. Procedures for Performing Systematic Reviews. Keele University. Technical Report TR/SE-0401.

Kivi, J., Haydon, D., Hayes, J., Schneider, R. & Succi, G. 2000. Extreme programming: A university team design experience. In Canadian Conference on Electrical and Computer Engineering, Vol. 2. , 816–820.

Klafki, W. 1988. Decentralised curriculum development in the form of action research. In S. Kemmis & R. McTaggart (Eds.) The Action Research Reader. (Third edition, substantially revised edition) Victoria, Australia: Deakin University Press, 235–244.

Knoke, P. J. 1991. Medium size project model: Variations on a theme. In Proceedings of the SEI Conference on Software Engineering Education. London: Springer-Verlag, 5–24.

Kolikant, Y. B.-D. & Mussai, M. 2008. So my program doesn't run! definition, origins, and practical expressions of students' (mis)conceptions of correctness. Computer Science Education 18 (2), 135 - 151.

Kollanus, S. & Isomöttönen, V. 2008. Test-driven development in education: Experiences with critical viewpoints. In ITiCSE'08: Proceedings of the 13th annual conference on Innovation and technology in computer science education. New York, NY: ACM, 124–127.

Kärkkäinen, T., Nurminen, M., Suominen, P., Pieniluoma, T. & Liukko, I. 2008. UCOT: Semiautomatic generation of conceptual models from use case descriptions. In C. Pahl (Ed.) IASTED International Conference on Software Engineering (SE 2008). ACTA Press, 171–177.

Lamp, J., Keen, C. & Urquhart, C. 1996. Integrating professional skills into the curriculum. In ACSE '96: Proceedings of the 1st Australasian conference on Computer science education. New York, NY: ACM, 309–316.

Lappenbusch, S. & Turns, J. 2005. Finding their place in TC: Using a community of practice model to research emerging TC professionals. In International Professional Communication Conference. Proceedings. Los Alamitos, CA: IEEE Computer Society, 524–533.

Last, M. 2003. Understanding the group development process in global software teams. In ASEE/IEEE Frontiers in Education, 33rd Annual. IEEE, S1F-20-5.

Laware, G. W. & Walters, A. J. 2004. Real world problems bringing life to course content. In CITC5 '04: Proceedings of the 5th conference on Information technology education. New York, NY: ACM, 6–12.

Lethbridge, T. C., Diaz-Herrera, J., LeBlanc, R. J. J. & Thompson, J. B. 2007. Improving software practice through education: Challenges and future trends. In FOSE '07: Future of Software Engineering. Los Alamitos, CA: IEEE Computer Society, 12–28.

Levy, D. 2001. Insights and conflicts in discussing recursion: A case study. Computer Science Education 11 (4), 305 - 322.

Lewin, K. 1946. Action research and minority problems. Journal of social Issues 2 (4), 34–46.

Lincoln, Y. S. & Guba, E. G. 1985. Naturalistic Inquiry. Newbury Park, CA: Sage Publications.

Lowe, G. S. 2000. Preparing students for the workforce. In ACSE '00: Proceedings of the Australasian conference on Computing education. New York, NY: ACM, 163–169.

Maehr, M. L. 1984. Meaning and motivation: Toward a theory of personal investment. In R. E. Ames & C. Ames (Eds.) Research on Motivation in Learning, Vol. 1. Orlando, Florida: Academic Press, 115–144.

Maslow, A. H. 1943. A theory of human motivation. Psychological Review 50 (4), 370–396.

McCutcheon, G. & Jung, B. 1990. Alternative perspectives on action research. Theory into Practice XXIX (3), 144–151.

McTaggart, R. & Garbutcheon-Singh, M. 1988. A Fourth Generation of Action Research: Notes on the Deakin Seminar.

Melin, U. & Cronholm, S. 2004. Project oriented student work: Learning & examination. SIGCSE Bull. 36 (3), 87–91.

Mezirow, J. 1981. A critical theory of adult learning and education. Adult Education 32 (1), 3–24.

Mochol, M. & Tolksdorf, R. 2009. Praxis-oriented teaching via client-based software projects. In A. Pears & C. Schulte (Eds.) 9th Koli Calling: International Conference on Computing Education Research. Uppsala, Sweden: Uppsala University, 31–34.

Moore, M. M. & Potts, C. 1994. Learning by doing: Goals & experience of two software engineering project courses. In J. Diaz-Herrera (Ed.) Software Engineering Education, Vol. 750. Berlin Heidelberg: Springer-Verlag. LNCS, 151–164.

Müller, M. M. & Tichy, W. F. 2001. Case study: Extreme programming in a university environment. In ICSE'01: 23rd International Conference on Software Engineering. Washington, DC: IEEE Computer Society, 537–544.

Nagarajan, S. & Edwards, J. 2008. Towards understanding the non-technical work experiences of recent Australian information technology graduates. In ACE '08: Proceedings of the tenth conference on Australasian computing education. Darlinghurst, Australia: Australian Computer Society, 103–112.

Napier, N. P., Mathiassen, L. & Johnson, R. D. 2009. Combining perceptions and prescriptions in requirements engineering process assessment: An industrial case study. Software Engineering, IEEE Transactions on 35 (5), 593 -606.

Nash, R. 2005. Explanation and quantification in educational research: The arguments of critical and scientific realism. British Educational Research Journal 31 (2), 185–204.

Neagle, R., Marshall, A. & Boyle, R. 2010. Skills and knowledge for hire: Leeds source-IT. In ITiCSE '10: Proceedings of the fifteenth annual SIGCSE conference on Innovation and technology in computer science education. New York, NY: ACM, 264–268.

Newman, I., Daniels, M. & Faulkner, X. 2003. Open ended group projects a 'tool' for more effective teaching. In ACE '03: Proceedings of the fifth Australasian conference on Computing education. Darlinghurst, Australia: Australian Computer Society, 95–103.

van Niekerk, J. C. 2009. Cost and Reward as Motivating Factors in Distributed Collaborative Learning Assignments: A Grounded Theory Analysis. School of Information and Communication Technology of The Nelson Mandela Metropolitan University. Ph. D. Thesis.

Norback, J. & Hardin, J. 2005. Integrating workforce communication into senior design. Professional Communication, IEEE Transactions on 48 (4), 413-426.

Olsen, A. L. 2008. A service learning project for a software engineering course. J. Comput. Small Coll. 24 (2), 130–136.

Outhwaite, W. 1998. Realism and social science. In M. Archer, R. Bhaskar, A. Collier, T. Lawson & A. Norrie (Eds.) Critical Realism: Essential Readings. London: Routledge, 282–296.

Palmer, S. R. & Felsing, J. M. 2002. A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ: Prentice Hall PTR.

Parker, H., Holcombe, M. & Bell, A. 1999. Keeping our customers happy: Myths and management issues in "client-led" student software projects. Computer Science Education 9 (3), 230–241.

Parlett, M. 1977. The department as a learing milieu. Studies in Higher Education 2 (2).

Peavy, R. 1999. Sosiodynaaminen ohjaus: konstruktiivinen näkökulma 21. vuosisadan ohjaustyöhön; translated by Petri Auvinen. Helsinki: Psykologien kustannus. (In Finnish).

Peirce, C. 1998. The essential Peirce: Selected Philosophical Writings, Vol. 2, 1893-1913. Bloomington, IN: Indiana University Press.

Petkovic, D., Thompson, G. & Todtenhoefer, R. 2006. Teaching practical software engineering and global software engineering: Evaluation and comparison. In ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education. New York, NY: ACM, 294–298.

Pfleeger, S. L. & Atlee, J. M. 2006. Software Engineering. (Third edition) Upper Saddle River, NJ: Pearson Education.

Piantanida, M., Tananis, C. A. & Grubs, R. E. 2004. Generating grounded theory of/for educational practice: The journey of three epistemorphs. International Journal of Qualitative Studies in Education 17 (3), 325–346.

Pierce, K. R. 1997. Teaching software engineering principles using maintenance-based projects. In Software Engineering Education and Training. Proceedings. 10th Conference on. Washington, DC: IEEE Computer Society, 53–60.

Poger, S. & Bailie, F. 2006. Student perspectives on a real world project. J. Comput. Small Coll. 21 (6), 69–75.

Polack-Wahl, J. 2000. It is time to stand up and communicate [computer science courses]. In ASEE/IEEE Frontiers in Education Conference, 30th Annual. IEEE, F1G/16-F1G/21 vol.1.

Poonamallee, L. 2009. Building grounded theory in action research through the interplay of subjective ontology and objective epistemology. Action Research 7 (1), 69–83.

Popper, K. 1978. Tree Worlds. The Tanner Lecture on Human Values. Delivered at the University of Michigan. Originally published in Michigan Quarterly.

Popper, K. R. 1979. Objective Knowledge: An Evolutionary Approach. (Revised edition) Oxford: Clarendon Press.

Press, I. C. S. & Press, A. 2004. IEEE/ACM Joint Task Force on Computing Curricula. Software Engineering 2004, Curriculun Guidelines for Undergraduate Degree Programs in Software Engineering.

Pressman, R. S. 1994. Software Engineering: A Practitioner's Approach. (European edition) McGraw-Hill. Adapted by Darrel Ince.

Ras, E., Carbon, R., Decker, B. & Rech, J. 2007. Experience management wikis for reflective practice in software capstone projects. Education, IEEE Transactions on 50 (4), 312-320.

Richards, D. 2009. Designing project-based courses with a focus on group formation and assessment. Trans. Comput. Educ. 9 (1), 1–40.

Robillard, P. N. & Robillard, M. 1998. Improving academic software engineering projects: A comparative study of academic and industry projects. Ann. Softw. Eng. 6 (1-4), 343–363.

Robinson, H. A. 1994. The Ethnography of Empowerment: The Transformative Power of Classroom Interaction. Bristol: The Falmer Press.

Rogers, C. R. 1983. Freedom To Learn for the 80's. Columbus, Ohio: Charles E. Merrill Publishing Company.

Rolland, C. 1998. A comprehensive view of process engineering. In CAiSE '98: Proceedings of the 10th International Conference on Advanced Information Systems Engineering. London, UK: Springer-Verlag, 1–24.

Roos, B. & Hamilton, D. 2004. Towards constructivist assessment? In Annual Conference of the Nordisk Forening for Pedagogisk Forskning, Reykjavik, 10–14 March.

Ross, D. 1989. The Nato conferences from the perspective of an active software engineer. In Software Engineering, 1989. 11th International Conference on. ACM, 101–102.

Rover, D. T. 2000. Perspectives on learning in a capstone design course. In ASEE/IEEE Frontiers in Education, 30th Annual. IEEE, F4C/14–F4C/19.

Royce, W. 1970. Managing the development of large software systems. In Proceesings. IEEE Wescon.

Rusu, A., Rusu, A., Docimo, R., Santiago, C. & Paglione, M. 2009. Academia-academia-industry collaborations on software engineering projects using local-remote teams. In SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education. New York, NY: ACM, 301–305.

Santanen, J.-P. 2008. Tietotekniikan opiskelijaprojektien kehitys, Projektiopetuspäivä. `http://www.mit.jyu.fi/palvelut/sovellusprojektit/tilaisuudet/projektikehitys.pdf` Retrieved on June 2009 (In Finnish).

Sayer, A. 2000. Realism and Social Science. London: Sage Publications.

Schulte, C. & Knobelsdorf, M. 2007. Attitudes towards computer science-computing experiences as a starting point and barrier to computer science. In ICER '07: Proceedings of the third international workshop on Computing education research. New York, NY: ACM, 27–38.

Schwaber, K. 1995. SCRUM development process. Presented at OOPSLA'95 Workshop on Business Object Design and Implementation.

Shafer, D. 2002. Practical graduate software engineering projects: Can something useful be built in one semester? In ASEE/IEE Frontiers in Education, 32nd Annual. IEEE, S3G-23 - S3G-29 vol.3.

Shaw, M. & Tomayko, J. E. 1991. Models for undergraduate project courses in software engineering. In Proceedings of the SEI Conference on Software Engineering Education. London, UK: Springer-Verlag, 33–71.

Shukla, A. & Williams, L. A. 2002. Adapting extreme programming for a core software engineering course. In Software Engineering Education and Training. Proceedings. 15th Conference on. Los Alamitos, CA: IEEE Computer Society, 184–191.

Shull, F., Lanubile, F. & Basili, V. R. 2000. Investigating reading techniques for object-oriented framework learning. Software Engineering, IEEE Transactions on 26 (11), 1101–1118.

Simon, B., Davis, K., Griswold, W. G., Kelly, M. & Malani, R. 2008. Noteblogging: taking note taking public. In SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education. New York, NY: ACM, 417–421.

Smarkusky, D. L. & Smith, H. H. 2004. Team projects throughout the curriculum: Course management, teaching initiatives and outreach. J. Comput. Small Coll. 19 (5), 119–129.

Smith, M. K. 1996; 2001, 2007. Action Research. The encyclopedia of informal education. `http://www.infed.org/research/b-actres.htm`

Sommerville, I. 2007. Software Engineering. (Eight edition) Harlow, England: Addison-Wesley.

Stake, R. E. 1978. The case-study method in social inquiry. Educational Researcher 7, 5–8.

Strauss, A. & Corbin, J. 1990. Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Newbury Park, California: Sage Publications.

Strike, K. A. 1982. Liberty and Learning. Oxford: Martin Robertson.

Suddaby, R. 2006. From the editors: What grounded theory is not. Academy of Management Journal 49 (4), 633–642.

Sudol, L. A. 2008. Forging connections between life and class using reading assignments: A case study. In SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education. New York, NY: ACM, 357–361.

Sun, N. & Decker, J. 2004. Finding an "ideal" model for our capstone experience. J. Comput. Small Coll. 20 (1), 211–219.

Suri, D. 2007. Providing "real-world" software engineering experience in an academic setting. In ASEE/IEEE Frontiers in Education Conference, 37th annual. IEEE, S4E-15-S4E-20.

Tan, J. & Jones, M. 2008. A case study of classroom experience with client-based team projects. J. Comput. Small Coll. 23 (5), 150–159.

Taran, G., Root, D. & Rosso-Llopart, M. 2008. Continuing challenges in selecting industry projects for academic credit: Points to consider and pitfalls to avoid. In Software Engineering Education and Training. Proceedings. 21st Conference on. Los Alamitos, CA: IEEE Computer Society, 163–170.

Teles, V. & de Oliveira, C. 2003. Reviewing the curriculum of software engineering undergraduate courses to incorporate communication and interpersonal skills teaching. In Software Engineering Education and Training. Proceedings. 16th Conference on. Los Alamitos, CA: IEEE Computer Society, 158–165.

Todd, R. H., Magleby, S. P., Sorensen, C. D., Swan, B. R. & Anthony, D. K. 1995. A survey of capstone engineering courses in North America. Journal of Engineering Education 84 (2), 165–174.

Tomayko, J. E. 1987. Teaching a Project-Intensive Introduction to Software Engineering. Software Engineering Institute, Carnegie Mellon University. Technical report CMU/SEI-87-TR-020.

Tomayko, J. E. 1996. Carnegie Mellon's software development studio: A five year retrospective. In Software Engineering Education. Proceedings. 9th Conference on. Los Alamitos, CA: IEEE Computer Society, 119–129.

Tomayko, J. E. 1998. Forging a discipline: An outline history of software engineering education. Annals of Software Engineering 6 (1), 3–18.

Turnock, C. & Gibson, V. 2001. Validity in action research: A discussion on theoretical and practice issues encountered whilst using observation to collect data. Journal of Advanced Nursing 36 (3), 471–477.

Tvedt, J. D., Tesoriero, R. & Gary, K. A. 2002. The software factory: An undergraduate computer science curriculum. Computer Science Education 12, 91 - 117.

Tyler, R. W. 1949. Basic Principles of Curriculum and Instruction. Chicago: University of Chicago Press.

Törngren, M., Grimheden, M. & Adamsson, N. 2007. Experiences from large embedded systems development projects in education, involving industry and research. SIGBED Rev. 4 (1), 55–63.

Umphress, D. A., Hendrix, T. D. & Cross, J. H. 2002. Software process in the classroom: The capstone project experience. Software, IEEE 19 (5), 78–81.

Urquhart, C. 2002. Regrounding grounded theory—or reinforcing old prejudices? A brief reply to bryant. The Journal of Information Technology Theory and Application 4 (3), 43–54.

Vaughn, R. B. 2001. Teaching industrial practices in an undergraduate software engineering course. Computer Science Education 11 (1), 21–32.

Vygotsky, L. S. 1978. Mind and society: The Development of Higher Mental Processes. Cambridge, MA: Harvard University Press.

Waterman, H. 1998. Embracing ambiquities and valuing ourselves: Issues of validity in action research. Journal of Advanced Nursing 28 (1), 101–105.

Williams, L., Layman, L., Slaten, K. M., Berenson, S. B. & Seaman, C. 2007. On the impact of a collaborative pedagogy on african american millennial students in software engineering. In ICSE'07: 29th International Conference on Software Engineering. Los Alamitos, CA: IEEE Computer Society, 677–687.

Wood, D. J., Bruner, J. S. & Ross, G. 1976. The role of tutoring in problem solving. Journal of Child Psychiatry and Psychology 17 (2), 89–100.

Yuen, T. T. 2007. Novices' knowledge construction of difficult concepts in CS1. SIGCSE Bull. 39 (4), 49–53.

Zilora, S. J. 2004. Industry-based web services project as a classroom teaching tool. In CITC5 '04: Proceedings of the 5th conference on Information technology education. New York, NY: ACM, 13–18.

YHTEENVETO (FINNISH SUMMARY)

Ohjelmisto- ja tietotekniikan projektiopetusta on tutkittu usean vuosikymmenen ajan, käytännössä yhtä kauan kuin termi 'ohjelmistotekniikka' on ollut olemassa. Projektikurssikirjallisuudessa on esitetty useita projektimalleja mukaanlukien tässä väitöskirjatyössä tutkittu projektimalli, jossa opiskelijat toteuttavat yhden lukukauden aikana sovellusprojektin oikeille asiakkaille.

Tutkimusalueen pitkä historia asettaa haasteita uudelle tutkimukselle. Tämän työn tutkimusotteeksi muodostui teoretisointi klassisen grounded teorian mielessä.Teoretisoivan tutkimusotteen motivaatio oli se, että sen avulla tutkimusalueelle tyypillistä kuvailevaa ja kokemuksia raportoivaa tutkimusta voidaan jäsentää helposti kommunikoitavaan muotoon sekä löytää uusia hyödyllisiä näkökulmia.

Väitöskirja koostuu kahdesta osasta, joista ensimmäinen pintauttaa mekanismin liittyen opiskelijoiden kurssikokemuksiin. Tämän syntyneen teorian mukaan opiskelijoiden positiiviset kokemukset vahvistuvat sen vuoksi, että he pääsevät kosketuksiin realimaailmaa vastaavan käytännöllisen kurssin kanssa. Kun realismia tarjotaan oppijoille, jotka ammatillisen kehittymisen mielessä tarvitsevat sitä, tapahtuu positiivisten kokemusten vahvistuminen ja toleranssi negatiivisia kokemuksia kohtaan kasvaa. Näin ollen kokonaiskokemus kurssista on usein positiivinen. Tämä mekanismi tarjoaa näkökulman, jonka avulla voidaan tarkastella objektiivisesti kurssipalautetta realistisilta ja käytännöllisiltä opintojaksoilta: positiiviseen kokonaiskokemukseen voi liittyä vahvistuminen johtuen oppijan kokemattomuudesta ja kurssin asemasta opinto-ohjelmassa. Tästä näkökulmasta opetuksen ja tutkimuksen tulee muistaa kiinnittää erityistä huomiota myös negatiiviseen palautteeseen.

Jälkimmäinen väitöskirjan osa teoretisoi toimintatutkimuksen viitekehyksessä tapahtunutta kurssikehitystyötä, jonka tarkoituksena oli etsiä ja teoretisoida kurssille sopivia toiminnallisia malleja. Tavoitteena oli se, että kurssi pystytään läpiviemään ilman projektien viivästymisiä ja opiskelijoiden resurssien ylityksiä, mutta silti riittävin tuloksin ja oppiminen huomioiden. Syntynyt teoria painottaa toiminnan ja kurssikontekstin välistä yhteyttä siten, että toiminnalliset mallit on havainnoitava kurssikontekstiin sopiviksi. Opettajan reflektointi on tässä keskeisellä sijalla, koska opettajalle, joka pystyy seuraamaan peräkkäisiä projekteja, on mahdollista havannoida ja käsitteellistää vertailemalla toimintaa, mikä puolestaan ohjaa projekteja oikeaan suuntaan. Opettajan on jaettava reflektoitua tietoa opiskelijoille sekä tuettava opiskelijoiden reflektointia, jotta oppiminen voidaan taata osana tuloksellisuuteen tähtäävää intensiivistä kurssityötä.

APPENDIX 1 GT REVIEW NOTES

Tables 4 – 11 provide review notes from systematic literature review regarding GT usage in CSE/SEE research (see Section 3.4).

TABLE 4 Theorizing vs. data analysis

Authors & Source	Q1: GT for theorizing or data analysis?
Isomöttönen & Kärkkäinen (2008a), IEEE CSEE&T, search: SEE & GT	Data analysis, theorizing by accident (see Section 3.4).
Williams et al. (2007) IEEE ICSE, search: SEE & GT	Data analysis, theorizing not mentioned but aims to (re)develop a conceptual framework.
Ho et al. (2004) IEEE Frontiers in Education, search: SEE & GT	Data analysis ("we used GT to analyze interviews and students' retrospective essays"), three initial themes given not a theory.
Last (2003) IEEE Frontiers in Education, search: SEE & GT	Cannot tell as paper does not explicate that it tries to theorize, resembles theorizing that leads to a typology of four themes and team cohesion as its core attribute.
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	Data analysis or "tentative" theorizing, hypotheses raised on the basis of qualitative and quantitative data.
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	Data analysis.
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Data analysis "We used grounded theory to analyze data".
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	cannot tell based on the text, most probably because of the one-page poster paper.
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	Cannot tell based on the text. Authors speak of theorizing only when they refer to DGT as a means of theorizing. Seems like data analysis.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	Data analysis.
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	More like data analysis to allow generation of new categories. Note: authors extend an existing external coding scheme.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	Data analysis, but refers to theorizing when speaks of grounded theory. . The end goal of the study in progress is to develop a "descriptive model".
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	Data analysis, but refers to theorizing when speaks of grounded theory.
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Data analysis (categories developed with NVivo software using grounded theory approach).
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	Both, the study present a goal of developing theories and cites DGT. Theorizing is most clear in the section where temporal ordering of student code tracing strategies is given. Otherwise is of data analysis.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	Data analysis, the identified categories were grouped into themes, and frequencies of the themes were given.
Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	Data analysis, there is no integration, the study provides themes with data examples.
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	Data analysis, there is no integration, themes with data examples given.
Jadud (2006) ACM ICER, search: CSE & grounded theory	GT is given only as a reference for theorizing from quantification, no theory is reported.
Kautz et al. (1999), ACM SIGCPR Computer Personnel, search: CSE & grounded theory	Cannot tell as the authors does not explicate their own objective, but seems like data analysis, GT (Strauss) is cited as a means to develop substantive theories.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	Data analysis.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	Data analysis, qualitative analysis is referred to, not theorizing. However, some integration between categories are given (condition-consequence causalities).
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Theorizing is given as a goal, the work presents initial themes (cf. data analysis) and aims to theorize later on. Authors say they do not start with preconceptions but the findings are guided by a typical taxonomy (technical vs. non-technical) which may prevent focusing on fresh analytic viewpoints in the first place.

Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	Not explicated, but refers to theorizing when speaking of research design, seems to be of verification of a carefully built and described preconception, more of data analysis than theorizing.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	According to author it's theorizing: is stated to provide a general theory on how students know and use their knowledge regarding difficult CS1 concepts. No core category, does not provide any explicit theory.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Not theorizing, and not given. More of content analysis to foreground categories from the data. Starts by identifying something that has an impact, but this "something" seems to be the pre-determined research assumption.
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	Not theorizing. GT used for categorizing thus as a means for data analysis.
Levy (2001) JCSE, search: grounded theory	Not theorizing, but constructing of initial categorical system from students' discourse.
Kolikant & Mussai (2008) JCSE, search: grounded theory	Theorizing given as an aim by referring to DGT. The study applied both a questionnaire and interviews and GT was used with the interview data. The interviews were informed by the responses to the questionnaire (cf. theoretical sampling).

TABLE 5 Description of data collection and analysis

Authors & Source	Q2: How was the data collection and analysis like, and how it was described?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Analysis technically, no, interpretations & outcomes with examples, yes
Williams et al. (2007) IEEE ICSE	Data collection technique mentioned; use of semi-structured interviews which has a written protocol, but the protocol itself is not given. With regard to the analysis, the relation between the data extracts given and conceptualization is unclear; data is interpreted with a previously developed framework.
Ho et al. (2004) IEEE Frontiers in Education	A general description of open, axial, and selective coding given but not examples of how the authors did it, i.e., how they interpret the data and use comparison. This may be due to the short poster paper.
Last (2003) IEEE Frontiers in Education	Research method given as is set out in GT books, illustrations on codes and related data examples, not on actual method usage.
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	Data collection and analysis descibed, an illustration (yet not very accurate) given on analysis (comparison).
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	General references given, but actual method usage remains unclear. Software was used for data analysis.
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Seems like conceptual labeling of subjects' actions, this information is given with examples.
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	Cannot tell, most probably because of the one-page poster paper.
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	Data collection given, no details on GT analysis (we developed a list of foci).
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	Data sources described; analysis not described with regard to use of GT.
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	Data collection described and analysis described. The latter is not detailed and it remains unclear whether the constant comparison (first given in DGT) was applied.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	Data collection and analysis described. The latter not detailed; a research method textbook is cited, i.e., uses general method description.
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	See previous item (this is a continuation study of it).
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Data collected using open-ended questions; this is given but not detailed. Analysis conducted using software; analysis is not described otherwise than by a reference to the software and GT in general.
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	Data collection (think aloud transcript data) and analysis described, the latter not explicated in terms of grounded theory (cf. constant comparison), general references made to DGT and data sorting techniques.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	Data collection described (collected with one question); analysis not detailed, a reference made to Strauss and Corbin without detailing the analysis.

Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	Data collected described (based on students' verbalization in problem solving), and also analysis steps described and codes given. However, the use GT procedures not explicated.
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	Data collection given. With regard to analysis, a general reference made to Strauss' procedures; micro-level analysis used (every sentence coded).
Jadud (2006) ACM ICER, search: CSE & grounded theory	Data collection is based on an algorithm taking student code as its input. The analysis is about visualization based on the quantification provided by the algorithm.
Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	Questionnaire partly described in detail; analysis described by referring to open, axial, and selective coding.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	Data collection described (all questions), analysis not detailed.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	Not detailed. Analysis was conducted in two steps, using part of the data to generate categories and then focusing on two dominating categories (theoretical sampling thus seems to be grounded on the dominance of the two categories), causalities were also searched.
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Uses interviews with pre-determined questions. The analysis not detailed , "major issues and themes were identified", but a criterion for them is given. Describes GT in a general manner. Ends up listing findings under pre-determined areas of an interview.
Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	Uses biographies as data. This is explained, but the actual coding is explained only by conceptually confusing citing of a secondary grounded theory source.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	Data collection (different sources and manners) given, coding and analysis not explicated. Only a general reference to GT when describing the analysis.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Data collection given (the students' assignment). Coding was about extracting qualifying student statements according to a set of pre-determined criteria. Common themes were then searched from these statements. Coding process not detailed, but resultant categories supported with data extracts.
Simon et al. (2008) SIGCSE, search: CSE & grounded theory	Data sources and techniques given, but GT usage not explicated: "Blog entries were categorized using a grounded theory based analysis...". Categories supported with data.
Levy (2001) JCSE, search: grounded theory	Techniques and data sources described. Inductive analysis used (GT not mentioned at this point) to find such patterns that repeat and stand out. First finds four dimensions and then focuses on one of those.
Kolikant & Mussai (2001) JCSE, search: grounded theory	Data collected with both a questionnaire and interviews that were tape-recorded and transcribed. A predefined set of questions/areas of interests was used in the interviews. Think aloud method and then probing questions were used to get the answers to certain areas of interest. Constant comparative method was used in the analysis of interview data.

TABLE 6 Joint data collection, coding and analysis

Authors & Source	Q3: Joint data collection, coding and analysis?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Data collected first with the questionnaire; seems to continue research based on teachers' previous observations, thus has some character of the joint approach.
Williams et al. (2007) IEEE ICSE	Cannot tell; the interview technique allowed divergence from the protocol; redevelops a conceptual framework.
Ho et al. (2004) IEEE Frontiers in Education	Cannot tell.
Last (2003) IEEE Frontiers in Education	Cannot tell.
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	Data collected first, but the interviews allowed to lead to other than intended directions.
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	Cannot tell, because software used in the analysis, probably not.
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Cannot tell. Because of observation technique in which the subjects were not interrupted, probably not.
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	cannot tell, most probably because of the one-page poster paper

Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	No.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	Cannot tell, but the results of first action research cycle informed the latter.
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	No, given that the data collection is outlined as a separate step.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	Parallel coding and analysis mentioned by referring to a research method textbook, but it remains unclear whether this took place in the study.
Deibel (2008) ACM ITiCSE, search: CSE & grounded theory	Yes (obviously also in the previous item in the table as this a continuation of it).
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Cannot tell, because software used in the analysis.
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	No, it is acknowledged, but was not possible due to the study setting.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	No.
Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	No.
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	No.
Jadud (2006) ACM ICER, search: CSE & grounded theory	No, but use of visualization aids to analyze the data while working with the data
Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	No; data collected first with the questionnaire. Cannot tell regarding the coding and analysis.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	No; data collected first with a questionnaire and repetition mentioned regarding the analysis.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	Data collected first. Theoretical sampling is based on the dominance of the categories, thus some kind of joint coding and analysis.
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Cannot tell; pre-determined questions used in interviews but the responses were also further probed for nuances.
Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	No; data collection purposively not affected by the researchers.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	Cannot tell; use of probing questions mentioned.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Cannot tell; two phases in data collection but it seems that reading the first set of data did not direct the later collection.
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	Cannot tell, but does not seem so.
Levy (2001) JCSE, search: grounded theory	Cannot tell, but does not seem so (collects data first and then analyzes it)
Kolikant & Mussai (2001) JCSE, search: grounded theory	Cannot tell; is possible as the interview group was selected according to preceding responses to a questionnaire.

TABLE 7 Approach to GT

Authors & Source	Q4: Appoach on GT?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Strauss' techniques, Glaser's approach by accident (latter not recognized in the paper)
Williams et al. (2007) IEEE ICSE	GT coding techniques used in collective case study. A mixture of collective case study and Strauss' approach and procedures
Ho et al. (2004) IEEE Frontiers in Education	Strauss' techniques, "tailored coding process for textual data"
Last (2003) IEEE Frontiers in Education	Strauss' approach
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	The main methodological reference is theorizing from a case study (not GT). Only a general reference made to DGT, seems to follow Strauss' style.

Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	Cannot tell, the primary method was an instrumental case study: "research design strongly influenced by grounded theory" is not explicated, NVivo software with GT mode was used in the analysis.
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Observational case study with "a grounded theory descriptive analysis" (seems more like Strauss' approach)
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	Cannot tell, most probably because of the one-page poster paper. Authors demonstrate a comparison of males and females responses coded to the same category (cf. method of comparison given in DGT)
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	Not explicated, only general reference made to DGT.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	Not explicated, GT only mentioned as a method for data analysis
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	"Hybrid of typological analysis and grounded theory". GT referenced as authors not only use pre-determined categories but also ones discovered in the data, otherwise not explicated
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	Not explicated, only a general reference made to grounded theory: "the goal is to inductively to develop hypothesis, concepts, and theories..."
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	See previous item (this is a continuation study for it)
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Cannot tell, DGT referred to as the mode used in NVivo analysis software.
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	Cannot tell, not differentiated. Besides citing DGT, uses also sorting techniques (unconstrained card sorts) to group the findings.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	Cannot tell, cites Strauss and Corbin, but does not explicate the method usage
Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	Strauss
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	Strauss
Jadud (2006) ACM ICER, search: CSE & grounded theory	GT referenced as a means of theorizing, no particular approach given, quantification and visualization to theorize from the data
Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	Strauss
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	Mentions that uses GT procedures only for data analysis, objective in not theorizing.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	Strauss
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Cites both DGT and Strauss & Corbin, not thus clear which stand is and will be taken.
Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	A modified coding, but remains unclear. No differentiation regarding Glaser vs. Strauss.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	Cannot tell; Only a few general references were made. "GT informed the analysis...".
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Discovery of GT is cited ("read in preparation for evaluating the student data") but the approach (Glaser vs. Strauss) is not otherwise explicated. Pre-determined criteria used in the analysis, but this does not seem to be theoretical sampling based on a discovery from the data
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	Cannot tell, GT just mentioned in the text as an analysis tool
Levy (2001) JCSE, search: grounded theory	Cannot tell, not explicated, mentions a possibility to further the findings to a grounded theory. Thus uses inductive analysis to develop an initial categorical system, perhaps without a need to account for a particular GT approach at the current point of the research.
Kolikant & Mussai (2001) JCSE, search: grounded theory	DGT and constant comparison are referred to, later differences between Strauss and Glaser not explicated.

TABLE 8 Citing of GT literature

Authors & Source	Q5: Correct citing?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Yes (Strauss with open, axial, and selective coding)
Williams et al. (2007) IEEE ICSE	Yes (Strauss with open, axial, and selective coding)
Ho et al. (2004) IEEE Frontiers in Education	No, refers to DGT but discusses open, axial, and selective coding
Last (2003) IEEE Frontiers in Education	Yes (DGT as original source and Strauss with open, axial, and selective coding)
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	Yes, references GT as a method of theorizing originating from sociology. Based on the analysis a reference could have been made to Strauss as well.
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	Yes, but refers only generally to grounded theory using many sources (DGT, Glaser, and Strauss) without differentiating them
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Grounded theoy mentioned but not cited at all
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	No references, most probably because of the one-page poster paper
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	Yes, only general reference made to DGT.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	Not cited.
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	Yes, DGT referenced as one first popularizing GT.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	Cites a general qualitative methods textbook; no differentiation between the approaches.
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	See previous item (this is a continuation study for it)
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Cannot tell, DGT mentioned when referred to NVivo analysis software.
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	Yes, DGT as the original book on GT. No differentiation between the GT approaches.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	Cannot tell, as does not explicate the use of the method.
Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	Seems so as the analysis steps resemble Strauss approach which is cited. Also two qualitative research textbooks are cited regarding the use of grounded theory.
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	Yes, Strauss with open and axial coding.
Jadud (2006) ACM ICER, search: CSE & grounded theory	Cannot tell, many references used without any commentary on them.
Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	Yes, Strauss with open, axial, selective coding.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	Cannot tell, DGT rerenced but analysis procedures are not explicated in detail.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	Yes, as cites Strauss and searches for causalities in a way which resembles Strauss' paradigm model
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Yes, but due to the absence of explicating the approach taken, this remains unclear
Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	Unclear in methodological concepts, might be because of using some secondary literature (methodological) source. For example, the work associates theoretical coding with open, axial, selective coding, and speaks of axial coding as a selection, and yet then speaks of selective coding.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	With these general references probably yes, but the citations are not informative regarding the approach taken.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Yes, refers to DGT as the original source for GT.
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	GT referred to without any citations
Levy (2001) JCSE, search: grounded theory	Yes, a general reference to GT by citing DGT.

Kolikant & Mussai (2001) JCSE, search: grounded theory	Yes, DGT with constant comparative method. Yet, some unclarity in concepts as authors speak of discussional theory building strategy not only a discussional way of laying out the theory — which to my knowledge is what DGT brings out.

TABLE 9 Core category

Authors & Source	Q6: Core category?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Yes, one results from the use of Strauss' procedures on the student data, but is also based on the challenges identified by the authors in the course context; an emergent hypothesis is also given which is actually more analytic, thus there is some unclarity in method usage.
Williams et al. (2007) IEEE ICSE	No; previously generated framework seems to be the core ("social interaction model of pair programming"), which, however, does not seem like an analytic core category but a label for a model. Based on the findings authors conclude with three "conjectures" whose emergence is unclear.
Ho et al. (2004) IEEE Frontiers in Education	No.
Last (2003) IEEE Frontiers in Education	Yes ("team cohesion"), but an external source is cited regarding the core.
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	No; Provides several tentative hypothesis.
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	Not from the data; an external concept (community of practice) is used as a frame for the analysis.
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Not from the data; An external conept, newcomer socialization, is applied to explain the results.
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	No; four categories given, one picked out as an example.
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	No.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	No.
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	No.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	No; Three tentative themes given.
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	No; Several themes emerged from which four is reported due to space limitation.
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	No.
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	No.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	No.
Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	No.
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	No.
Jadud (2006) ACM ICER, search: CSE & grounded theory	No.
Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	No.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	No.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	No.
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	No.

Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	No; In terms of GT, a preconception guides the analysis
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	No; Guided by pre-determined questions.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	No; List of categories.
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	No; Blooms's taxonomy is taken to interpret the findings.
Levy (2001) JCSE, search: grounded theory	No; list of categories (initial categorical system) under the selected areas of interest.
Kolikant & Mussai (2001) JCSE, search: grounded theory	Student perception, "partial correctness", stands out as an emergent main discovery while the paper also presents the students' methods of verification.

TABLE 10 Research process characteristics

Authors & Source	Q7: What was the research process like?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Analysis on one questionnaire data set, accompanied with identified practical challenges in the course context
Williams et al. (2007) IEEE ICSE	Conceptual framework built in several steps. This study continues the authors' previous work $\rightarrow$ theoretical sampling is referred to here
Ho et al. (2004) IEEE Frontiers in Education	A start up for a longer research process, relates to three-year research project (reports on initial findings)
Last (2003) IEEE Frontiers in Education	Relates to an on-going process. Results from a three-year period.
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	A case study (one course instance, not iterative) in which the data were collected at several time points during the students' work.
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	A case study (not an iterative) using GT methods for data collection and analysis. "Strongly influenced by grounded theory", but the actual GT process remains unclear
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	A case study with analysis on observations logs and subjects' video diaries, not an iterative process.
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	The poster paper reports on a result of a study that had started earlier
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	Two related projects which focused on reporting frequencies of found issues, not an iterative process.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	One project with two phases (two action research cycles), an ethnographic study which mentions GT only as a means to analyze data
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	Analyzing verbalizations of students problem solving in order to evolve a coding scheme and understand the capabilities of the students. Data from two different courses, but not an iterative process.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	An illustrative case study that interviewed two subjects twice during an academic term, GT taken for data analysis.
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	Four illustrative case studies (four subjects) in which all subjects were interviewed twice. GT adopted for data analysis (continuation study of previous item)
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Evolving a conceptual framework based on related work. Then collecting data and developing categories which were discussed in the light of the given framework
Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	Collecting the data and then analyzing it into initial findings (themes). This was followed by a return to data to verify the presence of the themes. It is noted that the result can be used to inform further data collection in order to advance the research.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	Collecting the data set and analyzing it into themes. The presence of the themes in the data was then judged (cf. verification).
Herman et al. (2008) ACM ICER, search: CSE & grounded theory	First identifying initial categories which were elaborated to thematic patterns. These were then verified with the data ("a decision about the presence of the theme was needed for it to be included in the final list of themes")
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	Codes generated first. Then the codes were developed to axial codes, which again were verified across the data.
Jadud (2006) ACM ICER, search: CSE & grounded theory	Analyzing (judging) students' code with an algorithm, and then visualizing the results to understand students' behavior; about creating means/tools for the research purpose

Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	Data collected with questionnaire and analyzed. Both quantitative data (background information) and qualitative data (actual research interest addressed with open-ended questions) collected.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	Students experiments collected with questionnaire and then analyzed.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	Collecting data by encouraging students to write esseys, and then analyzing the data.
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Devising in-depth interviews and then analyzing the resultant data.
Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	Presenting a theoretical framework, and a related preconception which is verified with student-written biographies.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	Devising data collection and analysis with eight study subjects. A pre-designed study.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Four phases of in the process, setting a question with a potential impact, collecting data, and evaluating data in two phases. A typical one-shot study on one course offering.
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	One course offering: collect data, analyze it, and interpret it with an external framework (Bloom).
Levy (2001) JCSE, search: grounded theory	Concerns one course offering. Data collected (observations and audio recorded student discourse) and then analyzed.
Kolikant & Mussai (2001) JCSE, search: grounded theory	Selecting students to respond to the questionnaire, and then selecting students for the interview part of the study so that the data from the former informed the latter selection. Then analysis on the interview data and reporting of the prominent theme.

TABLE 11 Preconception

Authors & Source	External preconception?
Isomöttönen & Kärkkäinen (2008) IEEE CSEE&T	Sets specific questions to the questionnaire according to previous teacher observations. However, the preconception is based on teacher conclusion from the same study context, it is not an external theory. There is thus a theoretical sampling character which is not acknowledged.
Williams et al. (2007) IEEE ICSE	partly yes (valuing pair programming).
Ho et al. (2004) IEEE Frontiers in Education	Can't tell because coding and interpretations not given (obviously due to page restrictions).
Last (2003) IEEE Frontiers in Education	Not likely, but the emergence of the core attribute is unclear.
Shull et al. (2000) IEEE Transaction on SE, search: CSE & GT	No. But the choice of techniques taught limits the study to only those two approaches.
Lappenbusch & Turns (2005) IEEE International Professional Communication Conference, search: capstone project & grounded theory	Preconception, framework of "community of practice" was applied in the analysis.
Begel et al. (2008) ACM ICER, search: SEE & grounded theory	Not apart from the section where results are explained with an external concept.
Almstrum & Last (2006) ACM SIGCSE, search: CSE & grounded theory	No.
Chen et al. (2007) ACM SIGCSE, search: CSE & grounded theory	No.
Cheng et al. (2004) ACM ITiCSE, search: CSE & grounded theory	No. A chosen toolset sets the boundaries for the results.
Chinn et al. (2007) ACM ITiCSE, search: CSE & grounded theory	Both a pre-determined coding scheme and emergent categories were utilized to evolve the pre-determined scheme.
Deibel (2007) ACM ITiCSE, search: CSE & grounded theory	No. Study subjects were allowed to talk about whatever they felt important while an interviewer had a set of questions for guidance.
Deibel (2008) ACM SIGCSE, search: CSE & grounded theory	No; see previous item (this is a continuation study for it).
Dick et al. (2003) ACM SIGSCE Bull. ITiCSE working group report, search: CSE & grounded theory	Categories emerged, but a pre-determined framework was applied to discuss the findings.

Fitzgerald et al. (2005) ACM ICER, search: CSE & grounded theory	Partially; problem solving strategies found seem to come from the data, but were also verified with the data, and then given under Bloom's Taxonomy and concepts of Strategic Learning. Also generally known concepts from the research domain (programming languages) are used to group the found strategies. Also includes a separated sections for emergent findings.
Hanks et al. (2009) ACM SIGCSE, search: CSE & grounded theory	No external concepts, but the findings were verified with the data.
Herman et al. (2008) ACM ICER, seach: CSE & grounded theory	No external concepts, but the findings were verified with the data (cf. deduction and verification).
Hewner et al. (2008) ACM ICER, search: CSE & grounded theory	No external concepts, guiding sample excerpts used in autobiography data collection. The results of axial coding were verified with the data.
Jadud (2006) ACM ICER, search: CSE & grounded theory	No, as does not focus on the results but the means and tools to gain understanding of the research topic.
Kautz et al. (1999) ACM SIGCPR Computer Personnel, search: CSE & grounded theory	No, but study context is based on particular systems development methodology, and study discusses results within context of existing theories.
Kollanus et al. (2008) ACM ITiCSE, search: CSE & grounded theory	No, concepts and conclusions derived from data. Constrained by pre-determined questions.
Melin & Cronholm (2004) ACM ITiCSE, search: CSE & grounded theory	No, the paper, however focuses on the results (two main concepts derived from data) that are introduced before the results of the analysis.
Nagarajan & Edwards (2008) ACM ACE, search: CSE & grounded theory	Constrained by pre-determined questions.
Schulte & Knobelsdorf (2007) ACM ICER, search: CSE & grounded theory	Yes, is of the tradition where the data are examined with a lens of existing conception — yet refers to GT.
Yuen (2007) ACM SIGCSE Bulletin, search: CSE & grounded theory	No, but limited and guided by two pre-determined questions.
Sudol (2008) ACM SIGCSE, search: CSE & grounded theory	Categories emerge, but the analysis affected/limited by the pre-determined criteria which arise from the research question.
Simon et al. (2008) ACM SIGCSE, search: CSE & grounded theory	Cannot tell based on the text; the categories may emerge from the data, but the analysis is guided by two research questions, and based on Bloom's taxonomy. Thus affected/limited by external framework.
Levy (2001) JCSE, search: grounded theory	No/cannot tell; categories seem to emerge but the study sets two specific, guiding research questions.
Kolikant & Mussai (2001) JCSE, search: grounded theory	Not biased. Guided by the questions, the main theme seems to emerge.