Shopping Web-shopping
Web-shop
Web-selling Selling

Teaching Software Development together with Work Practice Studies

Yvonne Dittrich
ydi@ipd.hk-r.se
Department of Software Engineering and Computer Science
University of Karlskrona Ronneby, Sweden

Abstract

In this article I want to share the experiences of developing an interdisciplinary course which brings together design oriented work practice studies and software development. The development and the teaching of this course took place as part of the study program 'People, Computers, and Work' bringing together human work science and computer science. To relate work practice studies and software development in a fruitful way, we had to change not only the organization of the projects and the layout of the course. It required also relating paradigms and methods on both sides. The teachers had to learn about each other's disciplines, too. An example of one of the projects – a web-shop for stationary – illustrates the way the students worked. Us teachers learned also a lot: Both ethnographic methods to study the workplace and software development methods change if really brought together. The variety of ways of relating work practice studies to software development surprised us. We explored common themes relevant for the interdisciplinary practice. And we learned a lot about how to teach this kind of projects.

Keywords: Teaching software development, ethnography, work practice studies, participatory software development

Introduction

In this article we want to share our experiences of developing an interdisciplinary course which brings together design oriented work practice studies and software development. The development and the teaching of this course took place as part of the study program 'People, Computers, and Work' (Helgeson et al., 1996). The goal is to teach design and development of computer applications in relation to work practice studies and design. In many of our courses we relate and integrate the disciplines Human Work Science and Computer Science. The students study 50% in each discipline. The whole program is still under development and thus addressing basic issues as: What are the relevant themes for teaching? How should they be taught? Which are the relations between the two subjects? The development of the course we describe here is part of that discussion.

In 1996 a five week's design oriented work science course, including a project

where the students did a work practice study leading to design proposals for both work and technological support, was moved to the same term as a five week's project course in software development. The idea was to use the same workplace for both projects, and in the ideal case to run the two project parts as one. Run for two years in a loosely coupled way, systematic problems became visible:

- Ethnographic methods and design focusing on user participation around mock-ups didn't work together with traditional software engineering methods.
- Five weeks is tight even for a stand-alone project course in software development. Keeping the ethnographic study very open and using participatory design methods at the interface between the projects added to the shortage of time for the development project. It just didn't work.
- For ethnographic studies in workplaces a group size of three students is the upper limit. Software development projects with less than four students don't really show the need for planning, co-ordination, and documentation.
- The way to cooperate in work practice studies differs from that in software development projects. In ethnographic studies, fieldwork can be done solo, but for the analysis the correction of individual biases through the discussion is crucial. Software development is very much dependent on a common picture of the future software – which surely will have to be maintained and updated –, but it does not make sense to have the whole group sitting around the screen and programming together.
- As part of the human work science course was to reflect on the application of ethnographic methods in design contexts, reflection was associated with this subject whereas software engineering was perceived as contributing a not really fitting set of skills and methods.

Whereas the work practice studies taking place first weren't influenced much, the software development projects did not work out. This influenced the perspective of the students on the computer science part of their program later on in a problematic way. Together with other changes of the program it was decided that the software development course should be extended to 10 weeks, and the two courses should be joined, intertwined and further developed.

In the following section I will motivate the changes and give an overview of the course. Thereafter I share one of the projects, the design and implementation of a web-shop. I then summarize and reflect upon our experiences. Not only didactic questions have been addressed, but also research questions related to the interaction between ethnographic work practice studies and software design and development.

The New Course

In the spring of '98 we started to develop the new course. A group of four teachers – two rather work science, two rather computer science oriented – met several times. We all had taught in one of the two courses before. Two of us had a background in the 'Scandinavian School' of systems development (Floyd et al., 1989b). Sara Eriksén was a Ph.D.-student of Pelle Ehn, Yvonne Dittrich was a student of Christiane Floyd in Hamburg. Betty Bergqvist is an excellent computer science teacher. Gunnel Andersdotter contributed her expertise as an ethnographer, doing research on the work of software developers.

Additionally, we made use of experiences from teaching in a small team project course within the software engineering program at our university; to allow the students to

learn from their own experience, the core of that program is project courses with real customers. (Ohlsson and Johansson 1995) Putting the emphasis on estimation, planning and reporting and on the relation to the customer, only part of the ideas could be taken over. The ‘people, computers and work’-program and our course put work practice and use-oriented development in focus. Nonetheless, both project courses take place in co-operation with a customer or users outside the university.

The major change was the already mentioned extension of the software development part. The new course is now a 15 week course, 10 weeks half time and 10 weeks full time studies. Beside that, the following subsections describe the areas of change. The last subsection gives an overview of the new course.

Changing the Group Size for the Software Development Project

From prior experience we knew that three, or at most four, students are the upper limit for a workplace study, but it is at the lower limit when it comes to software development projects. Furthermore, the type of co-operation is different: The goal in ethnography is to gain a wide and detailed picture about the work practice. Working together helps to explicate that picture and to keep each other from ‘going native’; i.e. becoming too familiar or involved with the field to be able to analyze it. Intense interaction, common analysis of field materials, and even writing together are important means. Of course, developing and maintaining a common picture of the future software, even together with the user is necessary in software development. Communication and co-operation are important here too (Pasch 1994). However, work also has to be distributed. It doesn’t make sense to sit with the whole group in front of a screen, and using the role concept to distribute responsibilities helps to keep track of what has to be done in a distributed way.

To meet both requirements and encourage a conscious change in the way of co-operation, we decided to join two ethnographic studies focusing on two workplaces related to the same future software. Two projects focussed on similar workplaces, on different workplaces that might be affected by the same application, or on different user groups. Each workplace study was carried out by 3 or 4 students. The joined team consisted of between 6 and 8 students. That matches the size of the ‘basegroups’ we established in our study program to enable mutual teaching and learning among the students, support them in finding a social home in the university, and help us teachers in providing group division for teaching in smaller units.

Joining two smaller groups after one third of the course time and half the term forced a conscious establishment of the software development project and an agreement on roles and responsibilities, ways of co-operation and coordination, and distribution of tasks

A New Way of Supervision

Having taught in the project courses in the software engineering program at our university, we realized that the split of supervision between the work science part of the project and the computer science part had major disadvantages. The supervisors did not meet the project groups regularly enough. Possibly conflicting requirements were not recognized. The students had to relate to two supervisors, and the supervision at each instance required a lot of time and effort; for a class of 40 students hand-ins for ten different projects had to be read, and constructively criticized. The groups had to be met and, besides the feedback, the project status had to be discussed. In both projects only

two teachers were responsible for the whole class.

We adapted the supervision model of the software engineering projects. There the supervisor stays with the team throughout the whole project, taking on the role of a head of department. We decided to change the role for the supervisor into a methodological consultant or a coach (Schön, 1983). The supervisor was responsible for two connected workplace studies and the joined software development project. Besides consulting the project in the choice of methods both regarding workplace studies and regarding software development, s/he had to keep contact with the project and give feedback on documents and deliveries.

Additionally, the students had the opportunity to book a limited amount of hours of supervision with design mentors and database design mentors who contributed their expertise and coached them regarding the architectural design of the software and the design of the database.

Teaching Each Other

Each project part put special requirements on the supervisors. The workplace studies included video-based interaction analyses where the supervisor has to actually do some analysis together with the students. It results in two reports, one focusing on the existing situation, and one on the analysis of participatory design sessions together with the users. In the software development project emphasis was put on the development and maintenance of a project plan – consisting of an overview of the project, the form of participation, the project organization, a time plan, and the methods to be used as well as the documents to be produced – the mock-up, and the existence of some kind of requirements, design and implementation document. We tried to encourage careful time planning and reporting. However, as the emphasis was not on project management, but on use-orientated development, we did not enforce it. In the end a common reflective report connected the two parts of the course.

A main problem now was our own qualifications. Each of us had an emphasis either on human work science or on computer science. We had experience in supervising either software development projects or work practice studies. We decided to meet every second week within the supervisor group, discuss the projects, the requirements for the partial deliveries of each project part, and help each other with supervision. Additionally we built peer groups, teaming up one computer scientist and one work scientist. These pairs helped each other in concrete supervision and backed each other up in cases of absence.

The mutual support turned out to be an important feature of the supervision. It additionally helped us to explicate our tacit knowledge within the two subject areas for each other. We wouldn’t have been able to supervise the whole projects without these regular meetings.

Changing the Paradigm of Software Development

The former software development course put the traditional way of developing software in the center. The main course literature was (Oscarsson 1994), a cookbook-like presentation according to the waterfall model, which even gave patterns for the different documents. The use and necessity of producing a complete and final requirements’ specification early in the project was never questioned, and the example fitted this paradigm. Additionally the students had Gilb’s ‘Principles of Software Engineering

Management ‘(1988) as course literature. This book emphasizes evolutionary delivery, software development management and quality control. Neither of the books referred to workplace studies, user involvement, or considered the evaluation of early versions by the actual users. The situatedness of real world software development and the necessary adaptation of methods were not an issue.

Already the year before we had added lectures on participatory and evolutionary development (Floyd et al., 1989a), communication and co-operation, and tried to talk about situated use of methods. Due to the lack of time and the existence of the Swedish software cookbook, they just followed that recipe.

We decided to change the whole paradigm of the course, putting participatory and evolutionary software development in focus. As the waterfall model and its deviates are part of the software development reality the students will meet outside the university, we had to teach the mainstream, the criticism of it, alternative approaches, and talk about their situated adaptation – no easy lectures, either for us or for the students. Teaching concrete methods, for example for planning and estimation, and at the same time discussing their necessary adaptation provided similar difficulties. Understanding a new tool and in parallel reflecting upon its constraints proved to be tough. We did not expect the students to learn everything during the lectures themselves. We wanted to give some hints, some conceptual anchors they could come back to during their projects. We gave all seven lectures in the beginning of the course. To support the students in making use of the contents we offered two repetitive seminars in smaller groups twice during the project time.

We decided to use no book about software development at all. ‘Professional Systems Development’ by Andersen et al. (1990) is out of print. We collected a set of six articles ranging from a mainstream overview (McDermid, Roock 1991) about evolutionary and participatory development (Floyd et al. 1989a) to object oriented design and development (Bürkle et al. 1995), and asked the students to use the library to find and select a set of methods fitting their project. They should select their own methods and design their own process and argue for it in a conclusive report.

The lectures began with a historical overview of software engineering, the software crisis, different threads of argumentation, and the ‘Scandinavian School’ of software development (Floyd et al., 1989b). We introduced the students to process models, planning and communication. A set of lectures focussing on the more product-related tasks followed: requirements, design, implementation, and testing. Throughout the lectures we emphasized the necessity of situated adaptation of methods as well as their reflected and agreed-on selection.

Of course, you improve the lectures every time you give them. Otherwise we decided to keep to the overall contents and order. Additionally, in the future we will encourage the reading of the compendium by asking the students to do short summaries.

The change in paradigm regarding software development made it much easier for the students to move from the workplace studies to the software development part. The methods in the two parts were no longer perceived as incommensurable. The reflective orientation of the lectures put more responsibility on the students themselves. However, in the end the practical experience from the project part was the main teacher.

The Layout of the Course

The course began with the Swedish fall term in September ’98. During the first 9 weeks the students focused on their field work, videotaped aspects of the work practice, and

analyzed part of it. During this period all of the software development lectures were held as well as most of the work science lectures and seminars.

After handing in the first part of their work science reports about the existing work practice at the beginning of week eight, the two subgroups working with related workplaces met and began to discuss what kind of computer application should be developed.

A formal project establishment should be set up around the 9th project week, resulting in a project plan that was subject to supervision the week after. The same week the development of an interface mock-up, of perhaps system visions, and some overall design should begin. This first time we ran the course, only the following mock-up based design sessions together with the users were considered common between the human work science part and the computer science part of the project. However, a mock-up without the preparation work for it is not possible. And subject of design on that level is also the anticipation of the future work practice. Looking back this whole part can be regarded as belonging to the work science part as well as the computer science part of the project.

The students videotaped the design sessions with the users. The analysis of these sessions respectively their taping was then subject of the second work science report, just before Christmas. Meanwhile the software development project was running in its hot phase. If possible a computer-based prototype should be delivered before Christmas. The goal was a running first version in the beginning of January. The students then had one week to write a reflective report on the whole project, the relation between the two parts of the project and their choice of methods. During two days at the very end of the term they presented their projects to each other.

To give you more of a flavor of what kind of projects took place, the following section tells the story of one of our projects; a web-shop design and nearly finished implementation.

Designing a Web-Shop

Base group five did their project together with our small university bookstore. The university bookstore mainly sells course literature, diskettes, pens and similar things at a fair price for the students. The two people who own the bookstore also work there. On an hourly basis they have additional help for unpacking books and delivery. So the two shop owners were both customers and users at once.

The intention was to build a web-based complement to their shop in a way that they could still handle it themselves. During the first weeks a complex arrangement became clearer: The web-shop was to collect orders that were to be delivered by another company. However, orders, billing, payment, and customer contact should be handled by the university bookstore.

The basegroup, three male and four female students, split up into two subgroups. As a web-shop clearly connects two different kinds of users – buyers and sellers – it was an easy decision on the work science projects. One group focused on Eva, one of the owners of the university bookstore, and her work, the other looked more into the customer side of it. Both ‘workplace’ studies had their specific difficulties.

“To control seventeen times, not to forget anything…”

The study of Eva’s work seemed at first to be a straightforward job. There was a set of places where it took place: the university bookstore has a shop at each campus of our university with set open hours. Most of the administrative work is done in the home office in the owners’ apartment. Just be at the right place at the right time, observe, and interview. Nothing sophisticated.

However, Eva was not just one of the future users, she was also a customer. And she had some experience with that role. She had been a customer to student projects of the software engineering program. During the interaction analysis of a taped session it became clear that she was not treating our students as ethnographers trying to understand her profession and her skills. She was treating them as software developers whom she had to tell what to do and what was important for her. This became especially obvious, when she picked up the phone and changed her attitude from someone requiring a service to someone providing a service.

Altogether the students got quite a good idea of what it means to work in such a profession. As they order the books from the publisher or the wholesaler on demand the way of working with the orders from the web-shop would be quite similar. The students studied the different subtasks and the paper-based system. It was important for Eva to keep track of the orders from the university students and of every step she was taking in order to provide their books. She had invented a very sophisticated system of representing different data necessary in different subtasks with partly redundant information, so that she had the possibility to proof-check herself and to correct mistakes. The computer based administrative system they used in parallel was rather an annoyance than a support. Eva perceived it as unstable. She did not dare to rely on it. Additionally, it was only accessible at home, so she had to keep track of the orders at the shops on paper nonetheless. The computer had not eased her task as could be expected. Besides giving insight into the task of selling books, the issue of security and control were recognized in their importance regarding the design of the web-shop.

Different Kinds of Customers

The other group had a somehow more problematic situation. As there was no existing web-shop, they could not do any observations of people buying stationary via the web. And can you really use the study of people buying in a real world shop as input for the design of a ‘virtual’ shop? They started out with a kind of experiment. They filmed people with different levels of expertise using two different web-based bookstores. Partly they sat beside the observée helping when issues became too sophisticated, partly they left the room leaving the camera behind.

However, that did not really feel like ethnographic fieldwork, more like laboratory experiments. During that time the first results from the discussion between the university bookstore and the wholesaler were settled on. The web-shop should address mainly small companies or private customers buying stuff in bulk. The students decided to visit a local wholesale store for stationary. During their visit, they were able to observe different customers. Even customers who were obviously familiar with the shop and its assistants showed different kinds of behavior; using the help of the shop assistant to find a fixed set of items, doing the familiar round through the well known shelves in order to collect the regular supply, or strolling around, testing and deciding for a special kind of pen, a special pack of post-its, etc.

The analysis of both field materials focused on the differences between the two 'mediating artifacts' – the real world shop and the web-shop –, the different possibilities to (re-) present the goods and the different ways to proceed towards the completion of the purchase. The question was, how the design of the web-shop could map the real world so the use of it would resemble as much as possible, or at least make use, of the customers' prior experience.

A Moving Target

Meanwhile – perhaps even because of the discussions with the students – the university bookstore decided on another way to organize the web-shop. Considering the effort to maintain an attractive stand-alone web site, they decided on a combination between a paper-based catalog and a very simple order interface. New customers should send an e-mail to the university bookstore and get a user-id and a password together with a catalog. The web-shop would then replace normal mail order.

The organizational structure on the seller's side became much clearer, too. The orders are collected by the university bookstore and passed on to the wholesaler's. They pack the orders and ship them according to the requirements of the customer, and they even write a bill on behalf of the university bookstore. Having done their job, they also send an invoice to the university bookstore. The university bookstore then pays the wholesaler and gets the money from the customer.

The information about the articles is provided in some simple data-format by the wholesaler. It is to be fed into the web-shop by the university bookstore.

This looked pretty thought through already. However, during the implementation partial deliveries became an issue. How are they handled between the involved parties? Who keeps track of them? The interdependency between organizational and technological change became a cause for delay. In the end the web-shop was not finished, partly because of the difficulties of defining the procedures to handle this issue.

Two Different Mock-Ups

Besides deciding on the software to be developed and the initial design, the next step in the project was the development of a mock-up and its usage in a participatory design session together with the users. The web-shop project decided to develop two different mock-ups, one about the buyer's side – the so-called outside – and one about the university bookstore's interface – they called it the inside.
The students built paper- and card-board-based representations about their design ideas. The outside mock-up was to be discussed and tried out together with potential customers and with the university bookstore. The inside mock-up was to be subject to a session together with Eva. Both mock-ups provoked feedback not only about the interface design. As expected, the functionality of the future software was the central issue of the discussion around the mock-ups. The prototypical customer proposed an extra page with special offers and asked for additional information about customer specific discounts. During the mock-up session with Eva functionality was even more of an issue. Despite the ethnographic study, the students did not feel sure about the requirements from the administrative point of view. The whole mock-up session was dedicated to developing the functionality and its representation on the screen. The mock-up sessions with Eva took altogether 6 hours, and can be regarded as a joint design process. During the interaction analysis it became visible that the relationship between her and the students

had changed. They became partners in design.

Both mock-ups were used as central requirement and design documents during the development process. They were important means for the communication between the students and the users as well as among the students themselves.

A Common Project Room

The web-shop happened to be the only project group with the privilege of an own project room: the university bookstore hired a server from the computer department. This server was the computer the students used for their implementation. The room where the server was located became the project room and main meeting place. This resulted in a very close and intense cooperation and in a very implicit planning and coordination process.

The project plan they developed in the beginning contained role descriptions, coordination procedures, co-operation policies, estimations and a project schedule. However, sitting in one room and working in close contact made it seem unnecessary to maintain the project plan, the estimations and the schedule. Of course they also changed the roles and the distribution of tasks, without any documentation. They themselves described their way of organization as a bit anarchistic.

Later, during the hot implementation phase, they had a detailed project schedule with daily deadlines on the whiteboard. In their reflection report they put forward that it would have been an advantage if they would have maintained the project plan and used it as a planning and coordination tool within the project group.

Documents

Because they themselves and the customers perceived the paper-based mock-ups as the central media for the design of the layout and the functionality of the web-shop, they did not spend much time and consideration on other documents.

For a long time they did not have a satisfying requirements specification for the inside for example. As late as when they began testing they recognized the usefulness of such a document, and then produced one. As the inside of the web-shop from a technical point of view consisted mainly of an update interface for the database in Visual Basic, no architectural design was considered necessary. The result of the mock-up session was captured in paper copies. They used these as the only design documents.

For the outside they developed different kinds of representation. They actually re-interpreted notations from object orientation in order to understand the interplay of web pages and active server pages. Only for the relational database – supplying the interface between the two parts – a thorough design was done providing necessary information for the other parts.

In the end the students agreed that an earlier requirements specification and additional documents would have helped them to structure their application in a better way and to stumble over inconsistencies earlier. The experience had probably taught them more than any amount of lecturing.

Learning through Experience

During supervision meetings of the project I recognized some of the shortcomings and their possible implications of an earlier stage than the students themselves. I tried to argue for maintaining and updating the project plan. I talked about the usage of a

requirement document to identify lacking aspects and contradictory requirements. I did not convince them to do one with any more success than I managed to convey to them the importance of a design document. As the students were told to decide for themselves about methods, documents and process model, we could not make them work the way we thought best.

Part of the problem might also have been the difference in culture and the mutual prejudice between the students studying the software engineering program and our students. Planning, estimations, and requirement specifications as the main base of agreement with a customer are ingredients belonging to the software engineers' culture. Perhaps partly because of the ongoing development, the way of working within the 'people, computers, and work'-program is described by older students as 'creative chaos'. This course was the first time we tried to introduce planning, methods, and documents as tools that – adapted to the project at hand – can support the participatory and evolutionary development of a complex product. With our experience in supervising and with the documented experiences of last year's students, we might be more successful next time.

However, as a colleague from the software engineering program expressed it regarding the teaching of their project courses: You have to find a balance between teacher influence and adequate opportunity for the students to learn by the experience of making mistakes.

Linking Ethnography and Software Development

Looking back on the course we gained a lot of experience regarding teaching ethnography and software development together. Some of the issues that popped up during the course can also be formulated as research questions. Of course the projects were not run under commercial conditions. However, the real world setting in which the projects took place and the real customers and users allow at least the formulation of issues and difficulties that might be relevant in research contexts as well.

Ethnography and Software Development Change if Brought Together

In the beginning of the course development, mainly the software development part was perceived as problematic. A traditional set of methods, a cookbook recipe-like process model, and a production-oriented paradigm did not work together with ethnographic field studies, where situated action of the observed as well as the observant are a central issue. The changes made as a result of the related considerations are described in the first part of the article.

That the changes done on the computer science side of the course, and the better integration of both parts of the projects that became possible with them, had influences on the work science side as well became visible during the course. Both, for the students and for the customers and users it was required from the very beginning that the project should lead to a first version of a software product. The students tried not to focus on a software solution during their ethnographic fieldwork. Nonetheless the setting changed the character of their studies. An example was Eva from the university bookstore addressing our students as software developers, not as participatory observers. The interaction between the ethnographer and the field, and the influence of that interaction on the result is an issue of discussion in ethnography itself. (Tedlock 1991) It becomes even more visible if ethnography is done in a context where the results will be applied to

design of artifacts, which can be expected to affect the field itself.

From the observers' side similar biases occurred. Most of the students knew in advance about possible needs of users, problems with the current situation, or – like the web-shop project – had a specific goal set from the very beginning. The ethnographic studies then had to be designed/laid out in a way yielding as much information as possible relevant for the software development project. Whom to observe when, and by what means, was not only decided according to the logic of the ethnographic study but was also influenced by the anticipation of the software development project.

Last, but not least, the observed practice was subject to change already through the process and the results of the study. In the web-shop example, the students pointing out the necessity of pictures and the presentation of the goods in categories helped the university bookstore to anticipate the difficulties of maintaining a stand-alone web-shop. In other cases, the future users began to think about improvements of present practice already during the field study.

On the other hand, the mock-up session could be seen as an additional ethnographic method. During the session with the users, aspects of the work practice that did not show up during the study become visible by talking about a possible future practice and its constraints. Playing 'What would happen if?' provoked a more detailed discussion of the existing and possible practice and its constraints. To use mock-up sessions or discussions around a prototype as an additional source for ethnographic data can be regarded as a creative re-interpretation of the provotyping concept by Mogensen (1991).

How Ethnography May Contribute to Software Development

At the latest since Suchman's book 'Plans and Situated Action' (1987) the relation between ethnography and design of technology has been explored. Using ethnographic field methods '… to provide designers with new ways of gaining a deeper understanding of user work practice and to provide a context for designers to collaborate with users over the design of new technologies' (Blomberg et al., 1991) is widely accepted. Ethnographic field methods are becoming even part of participatory design methods (Kensing et al., 1996). The relation between ethnography and participatory design is subject to controversial discussion. (See (Crabtree, 1998) for a presentation and contribution.) Is ethnography an alternative approach to requirements engineering, as implied by Anderson (1994)? Should the results be used to represent the user? (Brun-Cottan and Wall, 1995) If yes, in what respect? Or does ethnography reframe the understanding of the present situation and the formulation of what is the problem? (Anderson, 1994) Beginning experiences in joint projects (Christensen et al., 1998) have shown the mutual influence of ethnographic analysis, participatory design, and implementation.

In each of our projects, the relation between field studies and design and development of software was different. Already the web-shop example shows a variety of interactions: the students got a deeper understanding of the work practice to support and could derive constraints and requirements important for the future system. The fieldwork provided important requirements for the future work practice – control and security, in our example. The study about buying stationary in a wholesaler store and the resulting requirement to present the offer in an attractive way and to support different patterns of buying resulted in a different organizational design of the whole web-shop, and therefore in different requirements for the software supporting it.

Other groups used different but similar workplaces to objectify and juxtapose the

requirements from one potential use context: What are specific solutions in one workplace? How can they be generalized beyond one use situation?

The variety of possible contributions of ethnographic studies to systems development broadened our perspective: The evolving practice – not only in teaching contexts – might yield a richer picture than yet anticipated.

Common Themes

During the teaching of this first instance of the course we frequently discussed the relation between ethnography and software development. Despite the systematic distinction between a descriptive and analytic practice and a design practice (Button and Dourish 1996) we recognized also similarities, especially when applying an evolutionary and participatory paradigm for software development.

The situated use of methods is one very important theme connecting both areas. Methods provide both support and constraints regarding what can be achieved through their usage. They have to be selected and adapted according to the situation at hand. A reflection about what method is fitting in what respect regarding the project at hand is important in both ethnography and software development.

Another common theme is the use of representations. If representations in software development are not regarded as intermediary products in the intellectual assembly line of the waterfall model, but as design artifacts supporting the learning about the present situation and the anticipation of the future technology in use (Keil-Slawik 1993), their usage can be compared to representations and writing in ethnography. In both cases, representations mediate the development of an understanding; in ethnography an understanding of the object of observation is achieved, in software development the future technology in use is anticipated. In both cases, the representations serve to explicate and test hypotheses respectively design ideas.

Both themes might be used to connect and relate the practice of and reflection on ethnography and software development within the course.

Conclusions

Looking back on the experience from last fall, there are, of course, some things we would like to change. One issue is the name of the course. We just put together the both old course names: ‘Computers in Use/Software Development Methodology’. After teaching the course the first time we are now considering ‘Work Practice, Design, and Development of Software’, leaving open the possibility to relate design to the development of software as well as to the organization and practice within the use context. Actually the course is about both.

Already now we recognize further connections between the parts. Reports like (Christensen et al. 1998) support the effort to relate work practice not only to design but also to development of software. The experience of the co-operation between teachers from different disciplines was a challenge but also provided a lot of learning and experience for us. One of the most exciting aspects is that interdisciplinary cooperation can work out in such a promising way. Exploring the connection between computer science and work science in teaching and research for me is the main challenge in teaching in an interdisciplinary program, and working together in research with my colleagues at our university.

Even though the projects done by the students of this course cannot really be compared with 'real world' projects, and the students are no professional ethnographers, the projects provide a lot of input regarding the relation between ethnography and software development. The different ways in which the students related their workplace studies to the development of technology opened up for considering a larger variety of relations. Perhaps research questions cannot be addressed using the student projects as examples. However, reflecting on the students' experience seems to raise relevant research questions.

Acknowledgements

The development I describe would not have been possible without my colleagues Sara Eriksén and Gunnel Andersdotter now working in the Department of Human Work Science, and Betty Bergqvist from the Department of Software Engineering and Computer Science. Kari Rönkkö and Edith Sanchez Nunes joined the common adventure of teaching the course the first time and contributed important ideas to the in-process development of the course.

References

Niels Eric Andersen, Finn Kensing, Jette Lundin, Lars Matiassen, Andreas Munk-Madsen, Monika Rasbech, Pål Sørgaard. Professional Systems Development – Experience, Ideas and Action. Prentice Hall (UK) 1990.

R. J. Anderson. Representations and Requirements: The Value of Ethnography in System Design. In Human Computer Interaction, 9, 2:151-182, 1994.

Jeanette Blomberg, Jean Giacomi, Andrea Mosher, Pat Swenton-Wall. Ethnographic Feield Methods and Their Relation to Design. In Douglas Schuler and Aki Namioka (eds.). Participatory Design: Perspectives on System Design. Lawrence Erlbaum, NJ, 123-155.

Francoise Brun-Cottan and PatriciaWall. Using Video to Re-Present the User. In Communications of the ACM, May 1995, Vol. 38, 5: 61-71.

Ute Bürkle, Guido Gryczan and Heinz Züllighoven. Object-Oriented System Development in a Banking Project: Methodology, Experiences, and Conclusions. In Human Computer Interaction 10 (1995), 2&3, pp. 293-336.

Graham Button and Paul Dourish. Technomethodology: Paradoxes and Possibilities. Proceedings of the ACM Conference on Human Factors in Computing, CHI '96, Vancouver 1996.

Michael Christensen, Andy Crabtree, Christian Heide Damm, Klaus Marius Hansen, Ole Lehman Madsen, Pernille Marqvardsen, Preben Mogensen, Elmer Sandvad, Lennert Sloth, Michael Thomsen. The M.A.D. Experience: Multiperspective Application Development in evolutionary prototyping. In Proceedings of the ECOOP '98, Belgium, Springer 1998.

Andy Crabtree. Ethnography in Participatory Design. In R. Chatfield, S. Kuhn, M. Muller (eds.). PDC 98 Proceedings of the Participatory Design Conference. Seattle, WA USA, 12-14 November 1998.

Christiane Floyd, Fanny-Michaela Reisin, Gerald Schmidt. STEPS to Software Development with Users. In G. Ghezzi, J.A. McDermid (eds.):ESEC ’89. Berlin 1989.
Christiane Floyd, Wolf-Michael Mehl, Fanny-Michaela Reisin, Gerald Schmidt, Gregor Wolf. Out of Scandinavia: Alternative Software Design and Development in Scandinavia. In Journal for Human-Computer-Interaction 4 (1989), 253-380.
Thomas Gilb. Principles of Software Engineering Management, Addison Wesley 1988.
Bo Helgeson, Sara Eriksén, Berthel Sutter. How can Participatory Design practice be taught? In Jeannette Blomberg, Fin Kensing, and Elisabeth: Dykstra-Erickson (eds.). Proceedings of the Participatory Design Conference ‘96, MIT Cambridge, MA USA, 13-15 November 1996.
Reinhard Keil-Slawik. Artifacts in Software Design. In Christiane Floyd, Heinz Züllighoven, Reinhard Budde, Reinhard Keil-Slawik (eds.). Software Development and Reality Construction. Springer Verlag: Berlin 1992.
Finn Kensing, Jesper Simonsen and Keld Bødker. MUST – a method for participatory design. In Jeannette Blomberg, Finn Kensing, and Elisabeth: Dykstra-Erickson (eds.). Proceedings of the Participatory Design Conference ‘96, MIT Cambridge, MA USA, 13-15 November 1996.
John McDermid, Paul Roock: ‘Software Development process models.’ In John McDermid: Software Engineers Reference Book, 1991, 15/3-15/36.
Preben Mogensen. Towards a Provotyping Approach in Systems Development. In Scandinavian Journal of Information Systems, Vol. 3 (1991), 31-53.
Lennart Ohlsson and Conny Johansson. A Practice Driven Approach to Software Engineering Education. In IEEE Transactions on Education, Vol. 38 (1995), No 5: 291-295.
Östen Oskarsson. Programutveckling i liten Skala – en praktisk handbok. Studentliteratur 1994.
Jürgen Pasch. Software-Entwicklung im Team. Springer Verlag: Berlin Heidelberg, 1994.
Donald A. Schön. The Reflective Practitioner. How Professionals Think in Action. Basic Books 1983.
Lucy Suchman. Plans and situated actions. The problem of human-machine communication. Cambridge: Cambridge University Press 1987.
Barbara Tedlock. From Participant Observation to the Observation of Participation: The Emergence of Narrative Ethnography. In Journal of Anthropological Research 47, 1: 69-94.