Abstract

Requirements engineering is a process comprised of the requisite activities for creating and maintaining requirements documents. Different documents are produced at different stages of the development process. In most systems, requirements change. People develop a better understanding of what they want to do; companies and projects reorganise; when people change positions their successors may have a different understanding of problems and new ideas; modifications are made to hardware, software and organisational structures. Requirements engineering is concerned with managing changes in requirements. This paper describes requirements problems in a distributed software development project and describes them from the project members’ ‘own point of view’. The research methodology used is that of empirical and qualitative ‘ethnomethodologically informed ethnography’. It results in a discussion divided in two parts. The first part is concerned with the precise benefits and drawbacks of using the methodology. The second part illustrates how the central problem of requirements engineering is not completeness but the production of collaborative theory building and mutual intelligibility. This central problem is compared with one particular requirements engineering perspective presented in a book where ethnomethodology and requirements engineering are discussed.

1. Introduction

—Yes what does that mean? Who is it written for? Why do they need that? What roughly are its content, and what do they want it to look like? Key elements are missing there! These were the reactions of a manager, talking about his requirements. I mumbled something like, Key elements are missing? The manager continued: ... and from this point to derive who has introduced these requirements. It is the customer, but the customer is an unknown quantity. The organisational model in this project, a document-driven software development approach, succeeded in providing enough guidance for structuring the overall software development process, but it was less successful in facilitating project members' everyday co-ordination needs in handling the requirements. This paper is based on a qualitative study of a distributed project within an internationally prominent supplier of equipment for telecommunications systems and related terminals. The paper deals with two main issues. I show how the central problem of requirements engineering and software development is not completeness, but collaborative theory building and mutual intelligibility. Then I use the study to illustrate how
a qualitative approach can help realise these co-operative goals. By addressing these issues, the paper also contributes to meeting the challenge raised by Nørbjerg and Kraft (2002), the challenge of taking the context of software practice into account, especially, the importance of informal co-operation.

The structure of the remainder of the paper is as follows. First, I discuss contradictory ideas from the field of requirements engineering and software development. Second, the project is presented. Third, I describe the epistemological grounding of the methods used to study the project. Fourth, the project members' struggles with the complex organisation of work today are highlighted. Finally, conclusions derived from both the project members' struggles and the application of the qualitative methodology are presented.

2. Requirements Engineering

Requirements handling is a much-discussed topic. In analyzing the complex ways in which social and technical factors affect the development of software Jirotka and Goguen (1994) refer to the problem of requirements as a high-cost feature of the software development process in the latter stages of the development life cycle. They suggest that one possible interpretation for this is insufficient knowledge of the effects of the various available requirements methods used in the development process. They claim that we in fact only have begun to realise how requirements are actually construed and used in large software development projects (Jirotka and Goguen 1994 pp.2-3).

Within software engineering, the overall approach for handling the role of software requirements and the whole development process could be described as devoted to the idea of rigor and control. Methods have been developed to guide and control software development work. In models of this kind, requirements are the sole starting point for the development process. Developers should be able to handle them without further knowledge of the use context or the history of a project: a formulaic approach to requirements is emphasised. According to McDermid and Rook (1993 p. 29), no single interpretation of a language exists, pointing to the fact that possible misinterpretation of requirements during software development will always exist. Their contribution to this somewhat obdurate problem remains the proposal to develop an adequate formal language to minimise the misinterpretation possibilities. McDermid (1994) notes that no methods capable of adequately deal with requirements for socio-technical systems during software development seem to exist. He also points out that requirements are negotiated, not captured, warning us against the belief that well-defined requirements exists that are waiting to be discovered. He introduces three different approaches in requirements capturing and analysis, naming them orthodoxy, fundamentalism, and heresy.

The orthodox approach emphasises global holistic considerations and a uniform view of requirements. The problem is that the process leading to the creation of specifications starts in the wrong room, with requirements other than the fundamental ones. The requirements initiators are forced to provide more detail than is desirable in order to define requirements. This approach leads to low flexibility in design and implementation, because the over specification obstructs change. The system presupposed is effectively influencing the design
decisions before the determination of individual requirements takes place (McDermid, 1994 p. 25).

The **fundamentalist** approach addresses the issue of over specification through focusing on a small number of key requirements for a system so-called **cardinal-point requirements**. This approach seems to solve some of the problems of the orthodox approach, but it introduces others. The cardinal-point requirements are typically very **room** resulting in disparate and under specified requirements, and often necessitating large amounts of non obvious problem-domain knowledge in order to be interpreted correctly. There is no guarantee that the requirements are consistent, and problems can arise from under specification. There is also the risk of not bringing enough problem-domain knowledge into the requirements; system developers often lack this kind of knowledge (McDermid, 1994 p. 27).

The essence of the **heretical** approach is captured in the idea that it is impractical to produce complete, consistent, and implementable requirement specifications. Potential conflicts always exist that can only be unambiguously identified once there is an overall idea of the design. This view conflicts with ideas such as those represented by the Waterfall model, and thus it requires another process model enabling iterative and evolutionary development work. An advantage of the heretical approach is that conflicts between non-functional requirements and fundamental objectives can be recognised, which is not easy in the orthodox or fundamental paradigm. With the heretical strategy there is more freedom to change requirements and to achieve an implementable compromise. It is easier to validate the design with customers than it is to reach agreements on requirements, since customers agree to what they are actually going to get, not to what an analyst assumes that have required. In the most fundamental heresy, there is no fully elaborated requirements specification and requirements are finalised with the produced design. Ehn (1988) represents the Scandinavian participatory design approach (Ehn 1988 p. 35). Since they are critical to existing requirements analysis, especially the orthodox approach, McDermid still acknowledges the requirement approaches as being effective if used under appropriate circumstances. He sees requirements as a problem of appropriate choice and implementation of a normative method (McDermid, 1994 p. 37). Work-related problems, in this view, still solved through putting trust in rigor and control applied in the use of adequate methods.

Other authors have offered perspectives on requirements handling more closely connected to human and social issues. Concerned with what it means to program, Peter Naur claimed in his article *Programming as Theory Building* as early as 1985 that programmers build theories relating the design documentation and the software itself to its anticipated use. In other words the texts resulting from software development are outside the reach of what can be determined by rules (Naur, 1985 p.57). In her article *Software Development as Reality Construction* (1992) -influenced by Naur- Christiane Floyd claimed that we do not analyse requirements; we construct them from our own perspective. She bases her argument on constructivist thinking. She suggests that methods in use are orientation aids affected by our personal priorities and values and by our interaction with others constructing requirements (Floyd, 1992 p.95).
During the case study that provides the empirical basis for this paper, we tried to understand how software developers actually deal with requirements during design and implementation. The following section introduces the project. I then discuss the methodological approach we used in greater detail. After the analysis of the empirical material, the issues discussed in the preceding paragraphs will be revisited.

3. The Project Studied

How to approach widely distributed software development projects through qualitative methods is a well-known difficulty (Newman, 1998; Harper et al., 2000 pp. 76-77). One reason is that the project members' roles often change with every project phase, that is, there is no recurrent stability. Another problem pertaining to the distribution of work is how and where to identify what is relevant for the study. An additional problem is how to recognise and make sense of distributed work taking place at different locations at the same time. The project studied was divided into four subprojects distributed over five different locations in Sweden. The sub projects in turn were divided and distributed. Contracted companies were involved in the project. Consequently, it was difficult to define the setting and approach of the study.

The company involved produces advanced products and systems for wired and mobile communications in public and private networks for customers in more than 100 countries and has a long history in the telecommunications field. The project studied was performed in and by a Swedish component of that organisation.

The project aimed at developing a graphical programming environment, including training and methods. The environment was supposed to handle the company's existing telecom code used in their telephone exchanges. The high-level programming language supported by the graphical environment is Specification and Description Language (SDL). The main project was named SDL-Project. SDL makes tool-supported code compilation into lower-level languages possible. This means that an SDL description can be translated into an executable application without manual coding, leading to shorter development time and increased quality.

The SDL project included the SDL ToolCore subproject handling development of the code generator together with features of the tool. The following were also included: Training subproject handling and developing SDL training; Methods subproject handling the coordination of all SDL methods as well as standard methods; and SDL Tool subproject caring for signal handling, configuration management, release handling, function change, test port, and test methods. Two associated projects existed. The main technical orderer was located in Germany, and subprojects had technical orderers located in Sweden, Germany, the United States, and Spain.

The study began in the feasibility phase, and was completed during the 3rd (execution) phase. The field material is built up of maintaining and taping project meetings such as system group meetings, project leader meetings, steering group meetings and the execution phase's kick–off day. By hanging around in the local field where the main projects management and the SDL Tool subproject were located. Non-taped interviews were held with code developers, and taped interviews with the main project manager, four sub-project managers, the product
owner, the configuration manager, the project's quality responsible, and the main customer (main technical orderer). Because of distance, the interview with the main customer located in Germany took place by telephone. All other project members were interviewed at their local workplaces, with the concrete artefacts they used at work within reach to refer to. As can be seen, the study is somewhat top heavy, with the majority of the members studied being in some kind of management position. To place the study in the organisation and get responses on the findings, a steering group was created consisting of two people from the university, the SDL Tool subproject manager, the product owner, and the maintenance project member. In that group, the decision was made to give the study a subproject perspective. The group had regular follow-up meetings where fieldwork and early writing results during the field study were discussed. This paper is the product of later retrospective reflections on the material. The next section discusses the methods used, their epistemological grounding, and the problems perceived when relating them to software engineering.

4. Research Methods

The empirical study was conducted during a five-months period through applying quick and dirty (Hughes et al., 1994) ethnomethodologically informed ethnography, supplemented by interviews as a way of uniting perceived field experiences. I use the term informed to refer to the approach that Hughes, Randall and Shapiro (1992) pioneered, as the first serious attempt I am aware of, to connect ethnomethodology (EM) with design issues in work practice. Recent reflections concerning the usefulness of EM-informed ethnography can be found in Harper, Randall, and Rouncefield (2000 pp. 66-71). EM perceives the division of labour between the project members as routinely manifested in their own meaningful orientation to their work. Technology and work are, in this view, treated as technology-in-use, which is perceived as indivisible. EM is a highly descriptive framework with the goal of providing an alternative procedural description of achieved and achievable phenomena without sacrificing the describable, recognisable recurrences of ordinary activities. In this way, the perceived or recognised ‘structure’ in itself becomes the achieved phenomenon. This perspective may not seem obvious with respect to software engineering, where developed structures are valued in and of themselves, and where the goal of the research is to develop these new structures in the form of processes, methods, rules, and insights generally applicable to software development work. As a consequence, from an EM point of view, software engineering approaches also tend to obscure, ‘misrepresent’ or ignore the existence of a ‘real world of work’. Instead the perception of work is derived from the model or structure of work rather than from the activities in the work setting itself. EM-informed ethnography focuses on the activities themselves embedded in the socially organised domains -the locus of decision making- not in the structures developed.

It seems important to specify the epistemological foundation that the ethnography carried out in this project rests on, as a way of integrating other research perspectives (Anderson, 1996 p. 16). With respect to methodological assumptions, I take it for granted that phenomena in the social world are not constituted in such a way that they can be retrieved again by social scientists to enable scientific experiments, in contrast to the situation in the natural or physical sciences. Thus social scientists' conclusions remain speculative. I also assume that no one scientific method exists, in the sense ‘the method’, that is better suited to reveal the social organisation of the social setting's activities.
From an EM informed ethnographic viewpoint, a crucial question is whether researcher(s) have spent enough time in the field. Tracing patterns and identifying themes in a rigorously scientific manner is resolved through the use of the documentary method (Garfinkel, 1996b chap. 3). The documentary method cannot be evaded; there is no ‘time out’. This method is employed without exceptions to establish the correspondence between the phenomena actually witnessed and the underlying phenomena studied. Investigators interpret what appears to happen based on documentary evidence from the corpus of their experienced knowledge from within that setting.

It makes a difference what the documentary method is applied to; whether to a description of the phenomena or to already-developed structures. In the latter case, the findings are also influenced by some scientists view, and by the circumstances occurring when the coded results are worked out, that are, circumstances surrounding the need to use scientific language to furnish a scientific way of to create consensus and action (Garfinkel, 1996b p. 24).

In both cases there exists the danger of ground an analysis upon the perceived coded structures instead of on the actual phenomenon. The problem is that the coded results, the secondary scientific constructions, can be taken to be a part of the actual social organisation they purport to describe. That leaves us at the point where we started; the crucial point is whether the researchers have spent enough time in the field to grasp what is going on. To be able to interpret the ‘story’, it is necessary to have worked up enough feeling for how a ‘competent member’ in that field uses the documentary method.

To show the validity of my use of EM-influenced epistemology, I have to show how I handled this methodological problem. I have to explain my field material and its credibility from my own perspective. I have to show how and where I have spent time in the field (necessary in order to be able to make sense of the members' activities). And I have to keep to the EM commitment to show the ‘raw material’ in the resulting text (the justification I have for arguing that any particular thing is ‘going on’ should be evident, as ethnomethodology is highly concerned with the warrantability of data). I address these points in the following section, which present the result of EM-informed ethnography. Later I draw conclusions from the material shown.

Most of the ‘raw’ evidential material included in this paper is drawn from interviews and is not in itself the direct representation of the members' methods in action (which the transcribed meeting would be an example of). The interview material shown is actually retrospective ‘talk about’ the interviewees' methods in use. This conflicts to some extent with the EM idea, though ethnography is anything but a unified method and is perhaps not even a method at all as Shapiro (1994) has pointed out. Perhaps it is best regarded as an umbrella term for various analytic frameworks; in other words, it is a qualitative methodology. Yet, it is important to show the epistemology influencing the ethnography performed despite the tensions in the usage of the approach.

The next section presents part of a discussion from a steering group meeting as well as comments by subproject leaders, ending with suggestions for improving the project.
5. Requirements in a Distributed Project

The project members had problems getting organisational support in coming to terms with the scope of the requirements they were supposed to implement, despite the company's traditional way of managing projects. Through a document-driven waterfall-like model (or perhaps because of it), some requirements could only be traced with major difficulty; others could not be traced at all. The SDL-project preceded a reorganisation caused by a strategic overall resuffling of the international firm's line-organisation. This reorganisation resulted in a reduction in scope of some of the subprojects. As a result, and also because these cut downs and partly caused of the complexity of the work distributed it became difficult to adhere to the project's original plans.

5.1 Problematic Regarding Requirements

5.1.1 Steering group meeting

At the time of the change from feasibility phase to execution phase, the different subprojects were not in phase with each other's and the project plan. Some parts of the necessary requirements specifications not yet finished were discussed during a steering group meeting:

Standard methods-subproject leader: At this point I had a question, how shall we describe it? Because the description in the implementation proposal (IP), because in the IP we describe, we do not have these requirements so to say, we will remove very much of the requirements, the issue will be totally different.

Chairman: To reach formalism in the...

Standard methods-subproject leader: Yes it has to be described in some way...

Chairman: Yes.

Standard methods-project leader: Shall it be done in the IP or in some other way?

Customer-organization representative: Draw it in an updated IP.

Main technical orderer: I do not believe we have the competence to exactly describe it...

Standard methods-subproject leader: No.

Main technical orderer: In order to support the project, we must have a new suggestion on what has to be done, we need new suggestions on that...

Chairman: In the project we really would want to be able to take the toll gate decision; can we handle the issue in some way that will not delay that decision any more?

Main project leader: The fact that this delays both the delivery plan, it is surely affected whatever case, this is what worries me.

Chairman: Not in all of the...

Main project leader: Yes in all, we will have to remove standard methods from delivery five and six, we have an open IP, we also have an open project specification, and an open main project specification.

Chairman: Is there no way we can close them, and still have them with us? Not to get stuck in a lot of...

Customer Organization representative: That is my opinion too, on how to solve this, because this is actually not the only 'one' we have, there is the training, we could also include the code generator. In my opinion it would not be wrong to close them, to give TG2 on the prerequisite that actually is the existing situation.

Main project leader: hmm.
Customer organization representative: Then we will have a TG2B later, or whatever you call it. (Steering group meeting March 5, 1999)

Despite the fact that the expected foundation for the TG2 decision did not exist, the suggestion from the customer-organisation representative was exactly that: to make the TG2 decision. The situation that occurred together with the perception agreed upon in the meeting actually became the foundation on which the TG2 decision was made. In this way, the decision reached in this meeting was not about the completeness of any requirements specification, but about collaborative theory building and mutual intelligibility. The result of the discussion was the development of a sustainable arrangement that would allow the project to continue without a rigorous implementation plan and clearly delineated requirements specifying the work to take place.

In the following sections I will elaborate on the difficulty the project members articulated in coming to terms with their requirements, and will also discuss a related issue: their problems in tracing requirement originators. That brings to the surface the way project members described their requirements-handling work as a collaborative activity. In the first subsection I give two examples of how members perceived their trouble in handling requirements. In the next subsection I sketch solutions proposed by the members to the requirement problems. I then conclude the section by discussing what I suggested to help the situation.

5.1.2 ToolCore-subproject leader problem

How to handle requirements was planned as a straightforward issue. The idea of their model was to deepen the conception of requirements through references in the documentation. On the first level, the requirements were put together to form an overview; on this level, there were a few lines of explanation connected to each requirement and references to other documents. If a reference was pursued more detailed explanations, its origin, the customer organisation, reference names, and other related documents should be found. But when discussing requirements with the project members, it often seemed that there was something about the model that did not make sense:

Yes there is, as I said before there is a reference pointing to another document where it is possible to read more exhaustively, at least sometimes, not every time, sometimes there only exist a few lines of text. This has caused a lot of major problems (ToolCore-subproject leader)

5.1.3 ToolCore-subproject developer problem

When asking a project developer in the same subproject how he managed to grasp the requirements he was supposed to implement, he answered:

I don't, I mean I can't understand them only from the text in the requirements specification. What I do is, I contact the technical person that stands behind that specific requirement. The problem with this is that the person's name is never mentioned in the requirements specification, I mean the actual persons that once figured out the requirement (developer in the ToolCore-subproject).
Despite the fact that he was involved as an expert on compiler requirements in the formal project, he could not put his trust in the project's model for handling requirements.

5.1.4 Training-subproject leader problem

The subproject leader in the Training project expressed the same confusion about coming to terms with requirements:

Yes what does that mean? Who is it written for? Why do they need that? What roughly are its contents, and what do they want it to look like? Key elements are missing there! And from this point to derive it to who has introduced these requirements. It is the customer, and the customer is an unknown quantity (Training-subproject leader).

5.2 Developing Solutions to the Requirements Problems

The project members deemed the reference-document plan unsatisfactory. When travelling around talking to project members, I stumbled on a number of different ways to solve this conflict, solutions developed independently by the different project members. They actually seemed to be unaware that this was a common problem. These meetings with the project members also showed that, on some occasions when the project members succeeded in locating requirements initiators, they actually did not always have the time to help them:

They perhaps thought that yes, yes, I will look at that problem soon and get back to you later, but they did not always get back to us. This has made things really difficult for us (Toolcore-subproject leader).

5.2.1 ToolCore-subproject leader solution

The ToolCore subproject leader developed a sophisticated strategy of turning the problem around, actually making it the requirements-originator organisation's problem:

In my subproject, during the autumn in the feasibility, I solved this. I chose to solve this problem as follows; if we did not manage to get a requirement clear, we simply decided to produce a solution. Out of that solution the customer then had to understand how we perceived the requirements (ToolCore-subproject leader).

5.2.2 ToolCore-subproject developer solution

The code-developer from the same subproject presented another solution. On his own initiative he had established an informal network of contacts including technical persons all over the world, persons involved in the use of the earlier version of the code compiler now being tried out in real work settings:

I have to do my own research, starting to contact people, asking around to find out where this requirement originated. That means finding the technical individual responsible for every requirement, finding the people who originally wrote down the requirements, and other technical people as well, people who might have an interest in the specific requirements. I start asking around with the help of e-mail and the telephone, asking questions like: What
does x mean, and why? Then negotiations arise concerning the requirements. It is very seldom that I don't use my contacts; sometimes there are up to six people involved in a discussion concerning a requirement, people from all over the world. It often takes days to get an answer (developer in the ToolCore-subproject).

The same technicians who were part of this informal network actively contacted the developer as well. A network of technical engineers had established outside both the project's formal structure and its model for documentation.

5.2.3 Training-subproject leader solution

The Training-subproject leader approached the problem by travelling to his technical orderer, and ‘refused to leave’ before they had tracked down and talked to all the requirements-initiators together:

Requirements could come from separate individuals who later on had just disappeared. There is especially one requirement that has endured despite major protests from both the contractors and me; it is a requirement about a higher degree of simulation. But there is already a great deal of simulation. There is simulation all the time, you do a little design piece, then you simulate that, almost like doing a compiling test when coding, it is the same way with the high-level specification language. We started to sort out where this requirement came from, and who it could be attributed too. It turned out to be someone from Finland; it was a woman who did this off the top of her head at a meeting (Training-subproject leader).

This requirement was eliminated after the conversation with the Finnish requirements initiator. In fact fifteen of the original twenty-five requirements were removed as a result of spending weeks with tracing and negotiation work.

5.2.4 Proposed method changes as a result of the study

As a consequence of the study methodological refinements were proposed. These refinements included additions to the company’s requirements reference model: names, references, and contact information regarding the requirements initiators or other individuals who could take responsibility for the requirements should be given together with other details pertaining to the requirements specifications. If requirements sources are unknown, difficult to track down or not available, that should be noted in the field reserved for contact information. The company, of course, already provided such information, but not on this level. As could be seen from the examples, the contact information previously available was insufficient. The new suggestion would save many hours time, in future distributed software development projects.

Implementing the solution suggested above is not as easy as it might appear. Crucial investments, responsibilities and organisational sacrifices have to be handled. In a large international company even a small change in the world-wide project management-model becomes a large effort. Changes of course have to be explained and changed in existing manuals and training programs, and distributed in the organisation. How to change experienced project people who do not use manuals and are not the subject for training programs? Other things are, that employed now and then change their division and
organisation, and employed in key positions perhaps do not wish to be, or even have the time to be disturbed. The ToolCore-subproject leader touched the problematic when expressing: I believe that the individuals who once wrote the requirement had a good grasp at that time, but when time passes. And of course they do not always detail the requirement in a large amount of text... As in this case, four lines. If you go back half a year later and ask this person again, then he or she perhaps does not even remember, even worse, they might have changed their opinion. A proposed change in methods of the type just outlined would benefit from following-up research aimed at showing how it adapts to and incorporates ‘ad hoc’ features not easily described or captured in a method recommendation.

6. Conclusions

6.1 Is Completeness the Core Problem?

By focusing on the actual ‘work’, this paper has highlighted mundane achievements during software requirements handling in a distributed software development project. These are achievements that otherwise risk passing unnoticed during the members' ordinary daily interactions.

In McDermid's (1994) view, the difficulty with requirements handling seems to have to do with finding the ‘right’ model to control the requirements process. He suggests a hybrid approach: an orthodox model incorporating features from the fundamental approach, as they address different problematic aspects of requirements handling. The present study led to a somewhat different insight into the developers' needs. It seemed as if the most prioritised need was to interact with someone who could take responsibility for being requirements initiator, having adequate authority to negotiate a requirements interpretation. From this standpoint, the complete specification of requirements would not actually make the difference hoped for. This seemed to be the aim of McDermid's (1994) work. In other words, developing structured processes that lead to a complete requirements specification is perhaps not the core problem to be solved.

6.2 Human and Social Issues through Descriptions

A methodological problem occurred when relating the EM-informed ethnography to software engineering. EM is a truly descriptive epistemological framework, and as such it has nothing to say about design. Software engineering is about improving the software development practice. To make a difference in software engineering it is necessary to provide design proposals on the perceived problems. Replacing activities in work practice with ‘structures’ or tables conflicts with the EM epistemology. EM-informed ethnography describes the social world as it unfolds, avoiding any transformation of the perceived activities taking place. EM politely refuses to lend itself to any imaginary work, claiming that the ‘instances’ collected from the field speak for themselves and do not need to be organised around a core theory or cognitive model. EM is suspicious of the opposite tendencies, that is, to transform the ‘raw material’ by making it confirm to preconceived formats (Lynch and Bogen, 1996 pp. 266-267). Without disputing such achievements, EM asks what more is there to be discovered that the artificial tend to obscure (Garfinkel, 1996b p. 6).
The conclusions I draw from applying this approach are the following.

First, it is possible to relate ‘human and social issues’ to software engineering without doing a transformation of the ‘raw’ evidential material.

Second, the raw material gleaned from the field is actually closer to the social phenomena than the interpreted versions are. I do not deny that the raw material has to be interpreted at times, as the discussion of the documentary method discussion earlier revealed. I also do not deny that we need certain formats or frameworks that make it possible to discuss important themes in creative work. But I argue that social practices as they actually occur cannot be discerned or judged through rationalisations built on imaginary work. Garfinkel has captured the issue metaphorically by saying that it is: …very much like complaining that if the walls of a building were gotten out of the way, one could see better what was keeping the roof up (1996b p. 22).

Third, EM-informed ethnography in itself seems to suit the task of studying an ongoing work practice, but it does not support the envisioning of possible future ways of developing software. It is merely a starting point by making visible the actual activities that have occurred; in other words, it brings to light the work of today that is to be changed.

Fourth, I am not totally convinced that EM-informed ethnography is the best methodological choice when studying distributed software development. The use of the interview material seems to bear this point out. Much of the field material included in this paper is project members talk about their own work-methods, i.e. not by the researcher perceived ‘methods’ in use. Within EM this is referred to as talkaboutable or storyable. This situation occurred because, as a researcher, I found their talk about their requirements extremely interesting and did not have an opportunity to observe the occasions where they actually did what they talked about in the interviews.

To the extent that the EM descriptive view has been borne out in this paper in some sense, it has been done in the spirit of regarding the result as one possible version of many others (Garfinkel, 1996b). I also point to the humble attitude expressed by the founder of EM; I hope that there is room in this discussion for those studies which take the importance of witness able recurrent phenomenal fields of detail seriously and as a primary issue, in whatever other respects they may differ (Garfinkel, 1996a p. 6).

Acknowledgements

I would like to express my appreciation to the other contributors of the grounding section of the book Social Thinking - Software Practice: Yvonne Dittrich, Urs Andelfinger, Jacob Norbjerg and Philip Kraft, who have questioned, clarified and made useful suggestions on various points in the paper. Thanks also Bo Helgeson, Jeanette Blomberg, and Dave Randall who in their roles as course leaders in the doctoral course Work Practice and Technology II provided helpful comments on an early version of the paper. Thanks as well to the course participants, especially Hans Tap who critiqued the EM material.
Notes

1. This description is borrowed from the company's own Web site.
2. Garfinkel (1996b p.78) has expressed that The method consists of treating an actual appearance as 'the document of,' as 'pointing to,' as 'standing on behalf of' a presupposed underlying pattern. Not only is the underlying pattern derived from its individual documentary evidences, but the individual documentary evidences, in their turn, are interpreted on the basis of 'what is known' about the underlying pattern. Each is used to elaborate the other. "To decide a correct correspondence is a question of producing through interpretive work a correspondence that members of a community of cobelievers would agree upon, that is, on "common sense knowledge of social structures (Garfinkel 1996 pp.96, 76).
3. Lynch and Bogen (1996 p.281) note that Strange as it may sound, one must establish a right to have seen something and to have seen it that way (as storyable). What the members talked about in the taped conversations was the perceived real-world problems that bothered them. This was talk about the actual unfolding-work-practice side of 'structures' such as plans and methods.

References


