Bad Practice’ or ‘Bad Methods’
Are Software Engineering and Ethnographic Discourses Incompatible?

Kari Rönkkö, Olle Lindeberg and Yvonne Dittrich

Corresponding author
Kari Rönkkö
School of Engineering
Blekinge Institute of Technology
Soft Center
SE 372 25 Ronneby, Sweden
Kari.Ronkko@bth.se

Abstract

Organisational problems in industry have evoked increased interest in empirical methodologies in the broader software engineering community. In particular, the human role in software development has been addressed. Qualitative research approaches are identified as necessary for understanding human nature. The qualitative approach addressed in this article is that of ethnography in relation to software engineering. Ethnography emphasises the members' point of view in order to understand the organisation of a social, cultural and technical setting. Today, with very few exceptions, it is sociologists who have performed the majority of ethnographic studies of software development -- but how useful are these studies to software engineers? Ethnographic studies present problems from the observer's point of view. One implication of presenting studies from an ‘inside’ perspective is that they lend themselves to being regarded as revealing ‘bad methods’, i.e. which do not work in complex work situations. Taking from a software engineering point of view it is just as easy to point to the opposite interpretation of ‘bad practice’, i.e. a bad application of existing methods. The objective of this paper is to promote ‘ethnographic knowledge’ by revealing the different implicit research attitudes of ethnographers and software engineers, and to point to possibilities which combine studies that contribute an ‘inside perspective’ on software method improvement.

1. Introduction

How can or shall software development be best understood? Software development today is by no means a coherent or unanimously agreed upon concept with clearly defined methods or process models for practical applications. Research methodologies to for understanding software development are chosen depending on the problem at hand. The orientation of software engineering research has recently changed towards real industrial problems, and an increased interest in empirical methods has followed (Finkelstein and Kramer, 2000). This implies an increased acceptance of a diversity of empirical methods (Ibid.).

Ethnography is one empirical and qualitative methodology that has attracted interest (Sim et al., 2001). It contributes an understanding of the social aspects of everyday working life from the members' own points of view. The organisation of social or cultural settings is described from an inside perspective in the continuing; this paper adopts the term ‘inside view’
throughout. The authors have identified a lack of ‘ethnographic knowledge’ within software engineering (Ibid.). Today, with few exceptions it is sociologists who have performed the majority of ethnographic studies on software development. But how useful are these studies to software engineers? Ethnography has a history within sociology which carries with it certain basic assumptions. These assumptions or underpinnings have implications for the research attitude of the ethnographer. The implicit assumptions of ethnography are important if ethnography is to be adopted and used in software engineering. This paper has a valuable contribution to make to the community of software engineering by revealing attitudes that follow with the original understanding of the methodology ethnography. It demonstrates how ethnographers' attitudes clash with the attitudes of software engineers. In this paper, the ‘original’ understanding of ethnography is used interchangeably with Anderson's (1997) historical mainstream description of ethnography. Anderson's definition is summarised in one sentence by Harper (2000), see Chapter 2. The present article further points to possible ways to address this clash and to make use of ethnographically informed studies in Software Engineering research. First, the methodology is positioned in relation to research issues in software engineering.

1.1 Software Engineering

Since the founding of Software Engineering as a discipline in 1969 it has focused on the effort to re-design practice in accordance with the model of other engineering disciplines. Software engineering as a research discipline has focused on methods for product development; it has been successful in developing programming concepts, analysis and design methods, and process models widely used today. Parallel with this development a development of programming environments, CASE- and modelling tools, document repositories and configuration management systems have been produced. Despite the success of this software engineering development, many of the hitherto accepted research results have gained little acceptance in industry. Most researchers acknowledge that practice in industry does not comply with much, by no means all, of currently available research results (Finkelstein and Kramer, 2000).

While standard concepts, process and method guidelines were initially missing, today the problems have shifted ...we must pay attention to the complex interrelation of a number of organizational, cultural, technological, and economical factors. (Fuggetta 2000, p. 28). Both software engineering researchers and practitioners have expressed the need to address and solve organisational problems if the software engineering field is to advance (Finkelstein and Kramer, 2000; Fuggetta, 2000; Seaman, 1999).

This introduces new problems since organisations and cultures are complex and built by and made up of people. How can organisations and cultures be studied? Seaman (1999) has suggested that, due to the complexity of understanding human behaviour, qualitative methods are needed; statistical and other quantitative methods are not adequate for this task. Qualitative methods have the advantage that the researcher is forced to deal with the complexity of the social aspects. The results of qualitative methods are produced in the form of words, not numbers which by means of abstraction has obscured the complexity of mundane software practice (Ibid.). Ethnography is one such qualitative methodology that has gained some recognition in software engineering. One example of this recognition is that at
the International Conference of Software Engineering 2000 a workshop was arranged entitled *Beg, borrow and steal: Using Multi-disciplinary Approaches in Empirical Software Engineering Research* (Sim et al., 2001). The workshop accepted twenty-three papers, of which one quarter fell under the category of ‘ethnography’.

### 1.2 Ethnography in Software Engineering

This section demonstrates different understandings of ethnography, as related to different research attitudes in software engineering. Three different software engineering approaches ‘relating to ethnography’ are presented in this section. The first is related to the idea of collecting qualitative data in an initial research phase that is subsequently quantified in the analysis phase (Seaman and Basili, 1997a; 1997b). This does not qualify as an ethnographic study in its original sense. The second view takes its inspiration from the same roots from which ethnography originated (Lethbridge et al., 2001). The third view is to relate software engineering to the original understanding of ethnography. Among the studies already carried out which address ethnography in its original sense can be mentioned (Dittrich and Rönkkö, 2002; Rönkkö, 2002a), and discussions of how to relate the original understanding of ethnography to software engineering include (Dittrich, 2002; Rönkkö, 2002a and 2002b).

The two studies by Seaman and Basili (1997a; 1997b) are examples of improvement-oriented ethnographic research as well as of the tradition of quantifying research results. These two studies are the only ethnographic studies found that clearly use the term ethnography in terms of methodology too; *prior ethnography*. This way of understanding and using ethnography differs from the ‘mainstream’ use and the general understanding of ethnography within sociology (Andersson, 1997; Harper, 2000) as it is directed towards processing qualitatively collected data according to predefined schemes instead of as it is opposed to producing descriptive inside view of the studied practices.

A study by Lethbridge et al. (1997) seems to contain an initial ethnographic field study, even though the publication is not a descriptive ethnographic contribution but rather a quantitative presentation. Lethbridge et al. (1997) call the qualitative study a ‘work practice’ study, i.e. not an ethnographic study. They divide the study into two separate parts: a qualitative field study and a quantitative analysis aiming at improvement proposals. It is not clear whether the initial work practice study takes on the original ethnographic stance of an inside view or not. Obvious, according to the ethnographic definition (Harper, 2000) the second part of this paper is not ethnography; i.e. the quantitative analysis together with its numerical presentation of the results does not qualify as ethnography in its original sense. It would have been interesting from a methodological point of view if the authors would have presented whether and how it is ethnography in its original sense that have influenced the scheme for the quantitative analysis. Lethbridge, Sim and Singer (2001) have also suggested a research approach influenced by the same roots as that of ethnography.

Based on a set of smaller studies (Dittrich and Rönkkö, 2002; Rönkkö, 2002a) the authors explored how to apply ethnography in a more original sense as part of software engineering research. Based on this experience, a research approach has been suggested which has been

---

1 The authors might have missed some articles. It seems likely that the authors have found most of the articles on the subject as no further studies are referenced in the literature.
tentatively called cooperative method-development. (Dittrich, 2002) Ethnographical studies provide the ‘inside view’ as input to identify and develop possible improvements together with the practitioners involved. This way of relating to the original ethnography will be further explored in the conclusions.

The above described literature survey of software engineers' ethnographically influenced approaches shows that software engineers' tend to have a common conviction that the analysis even of even qualitative data be nefits from quantitative analysis according to predefined schemes. This mathematical way of analysing is inconsistent with the original understanding of ethnography. The difference in the software engineers' method of analysis revealed by the authors implies a different kind of research result to that of the original sociological one.

What then is the original sociological understanding of ethnography? What would be the consequences of presenting ethnographic studies in the original understanding of the term to software engineers? The first question is answered in the Chapter 2. In Chapter 3 examples of ethnographic studies are provided. The question of consequences is discussed in Chapter 4, highlighting the different interpretations leading either to ‘bad practice’ of methods in use or ‘bad methods’ not suited for real world circumstances. Chapter 5 concludes this paper; two ways of benefiting from the original understanding of ethnography in the field of software engineering are suggested.

2. Understanding the ‘Original Ethnography’

Ethnography originated from the need to study foreign cultures (Anderson, 1997). It was Malinowski who in 1915 invented the professional stranger, the ethnographer. For Malinowski, the purpose of fieldwork was to become intimately familiar with the setting studied. This was done by learning language and culture, and by living according to studied setting's regime. This intimacy leads to the subjective understanding that is necessary to integrate and correlate the found data in the form of synoptic charts, detailed descriptions of day-to-day activities and narratives. The idea behind creating such ‘ethnographic accounts’ was that things are not always what they seem to be, i.e. appearances are not the whole story. The native or participant him/herself did not always have the ‘overview’ that the ethnographic account could present (Ibid.).

So far it can be concluded that the ethnographic account is a post hoc representation of the field studied. Obviously, it is also a representation or account that involves interpretation and analysis from the very first stage. This personal way of executing research is legitimated by the ethnographers' ways of ‘knowing’ what others do not and cannot know as they lack the personal field experience of the ethnographer. This is the ethnographic qualitative warrant or scientific trademark; the ethnographer writes with a voice and from a point of view. This approach seems unscientific if viewed from the natural sciences. But this way of thinking does not turn to the natural sciences for its guidance; it turns to the hermeneutic disciplines (Andersson, 1997). This also explains why it is the fieldwork experience, and not the fieldwork findings in themselves which is of greatest importance. Historical ethnography is often described in basic textbooks as a body of procedures and techniques that anyone can learn and apply. This has been questioned; is it the following of the procedures and techniques that actually leads to an adequate ethnographic account? As consequence of this doubt, the
techniques for identifying, listing and integrating activities from the studied peoples world are not much emphasised anymore, instead ‘the native's point of view’ has been identified as the heart of the ethnographic enterprise (Ibid.). For a more extensive review of ‘mainstream ethnography’ from its origins to the present day, see Anderson (Ibid.).

Harper (2000) has described ethnographic method based on one basic assumption: it is a method for understanding what activities mean to the people who do them (Ibid., p. 254). If this assumption is not met it is not ethnography, it is some other kind of technique or field method. From this assumption also follows that an ethnographic account is a descriptive one. This is because the account needs to present the circumstances of the phenomenon studied: their life, their meaning, their purpose and point, i.e. the things that give the observed conduct the meaning it has. If one does manage to describe all these aspects, then one has a so-called thick description. One problem in producing such descriptions is that some activities are so ordinary and mundane that it becomes difficult to know what exactly needs explanation. The suggested solution to this problem is to keep to the idea of presenting the world as perceived by those within that world. (Ibid.)

3. Ethnographic Studies of Software Developments

What can the original descriptive ethnographic research approach contribute to software engineering? One way to answer this question is to look at existing ethnographic studies of software development practice. The following demonstrates small sections of larger studies that have software development as a common theme. The sampling of papers could be labelled ‘convenience sampling’ (Robson, 1997, p. 141); the set of studies we discuss is not intend to cover the entire field of software development. Neither would presenting those studies contribute to the argument. From these small sections we demonstrate the different nature of the research findings presented in comparison with what is expected in software engineering. The studies captured in these smaller sections were originally carried out in accordance with aims which do not necessarily correspond with the way in which the studies have been applied in this paper. In order to shed some light on the different attitudes possible to apply to software development some of the studies has been reinterpreted in some parts and used in a way for which they were not intended. This reinterpretation is perfectly in line with one of the main points of this paper, namely that studies often lend themselves to different kinds of interpretations.

3.1 Accepted Vulgar Competencies

Button and Sharrock (1995) present a study of code writing practices in The Mundane Work of Writing and Reading Computer Programs. One of the conclusions in this paper is that in software engineering ‘the vulgar competencies of common-sense knowledge and practical reasoning’ are accepted as essential for software development. In fact, programs are written to format the code automatically in accordance with the computational process to support the application of these ‘vulgar’ competencies. A programmer who tries to understand a program uses any available information, jumping from high-level documentation to low level code reading, following functions or calls to other related parts. Software engineering methods that demand documentation and code comments have the unintended side effect of making the work that went into the writing of the code visible to others that later try to make sense of the program in order to change it. Hence, the ‘vulgar and common sense practices of writing and
interpreting code’ have documentation support related to the code. The authors contrast this with the natural and social sciences' practice where the practical work that sets constraints on scientific results goes unrecognised, and with mathematics, where in the final product, -the proof-, there is no longer any visible trace of how the mathematician succeeded in finding the proof. In software engineering there is a tendency to formalise the division of labour. It is not always clear whether the ‘vulgar practice aspect’ is taken into consideration, or not.

3.2 Wait with Some Decisions Until a Later Stage

In the same paper by Button and Sharrock (1995) the focus is on the detailed practices applied when writing code and not the overall development process. Nevertheless, these practices have implications for the validity of basic assumptions in software engineering. One example is the programmer's practice of sometimes using temporary names for variables (e.g. \texttt{temp-1}) instead of immediately providing an intelligible name. This is because programmers are uncertain about how to name the variable until they have completed the major part of the program and the role of the variable has been clarified. They then change the temporary name to an intelligible one. The authors describe the activity of selecting informative names in terms of the programmer engaging in constructing a taxonomy of a semantic domain, for their practical and situated purposes. They exemplify by asking what to call the function of turning on a warning light on a photocopier: should it be called a fault indicator or a problem indicator? The correct name cannot be chosen without taking all other warning lights into consideration.

This observation presents a problem with one particular view of SE, the top-down view. The top-down strategy for program development is based on the idea that the program is broken down into parts that can be implemented independently of each other. The observations in the study make it painfully clear that naming is a global problem. The names of different conventions must be the same throughout the program to avoid confusion.

3.3 Should Requirements or Architecture be Developed First?

In her article Here, there, and nowhere at all Newman (1998) describes a study of the development of middleware in a fortune 500 company. The actors involved in the design of this software are a diverse and shifting group, coming from both inside the company and from other companies (she lists 12 types of actors). These actors form an interconnected web which changes over time. The main problem taken up in the article is how to define the place where the ethnographic study should be carried out. This is not our concern here. However, it is interesting to note that the design is made by shifting groups of different actors, each with her/his own set of concerns. This suggests that the design process described is in no way a linear one starting with goals and requirements and ending with a complete design. Instead, it is done concurrently with the specification.

From a software engineering point of view, the situation described does not at all fit into one ‘desirable framework’ where the development starts with requirements analysis. What Newman is describing is a situation where the internal technical design of the middleware is constructed at the same time as the description of what the middleware will actually do. For a software developer this is not very surprising, the architecture of the middleware will make some things easier and other things nearly impossible, it is not possible to make the tradeoffs
necessary before the technical design is conceptualised. Newman also describes a situation where there is no clear customer but a whole set of actors with different agendas.

3.4 Creating and Constructing on the Same Occasion

In their article *Artificial intelligence as craftwork* Suchman and Trigg (1993) concentrate on the use of whiteboard sketches. One conclusion is *that the work of AI involves a series of transformations or re-representations*, starting with observations of the social world and ending in executable code. In the article, a design talk between two AI researchers is analysed. The AI researchers use a whiteboard to sketch on and use gestures to show the dynamics. The figures on the whiteboard are seen as both representing a logical formalism and a Lisp construction. They do not clearly differentiate between these levels but make reference to the level most relevant for the moment.

One issue in this paper that is interesting in relation to software engineering is stepwise formalisation. In a sense, it is possible to conclude that the AI researchers follow a traditional stepwise formalisation model since they do not start writing the actual code until the formalism is elaborated. In another sense, they completely break with the stepwise formalisation model since they allow reasoning on code level to settle how the formalism should look.

4. ‘Bad Practice’ or ‘Bad Methods’

By comparing the improvement oriented software engineering view with the presented ethnographic cuts, contradictory views and discourses can be identified. With an improvement orientation in mind these studies could be read as consciously suggesting weaknesses in methods, ‘bad methods’ overlooked by software engineering research. On the other hand, the studies can be interpreted as reporting ‘bad practice’ of incompetent software developers. In the following, the study results are reconsidered, together with a suggesting for one possible software engineering response:

- From the ethnographic inside view the first study suggests more awareness of programmers' practical needs to support vulgar competencies of common-sense knowledge and practical reasoning. Another interpretation would suggest that these aspects are not important enough to be the subject of software engineering research.
- In the second study the naming problem was taken up, it can be read as supporting the idea that you must sometimes wait to make some decisions until a later stage. Methods should support this need for to wait with some decisions. Another interpretation would be that the design should already have clarified which distinctions to make. Delays in naming should not be necessary.
- The observations of the middleware project might suggest that it is not always possible to specify requirements before the architecture is defined. Neither is it possible to decide on the architecture without having specific requirements. The study suggests that methods must allow for requirement analysis and design to be intertwined.

An alternative interpretation would be that the project management did not establish a
stable enough project environment. An evolutionary software development model would solve part of the problem.

- In the fourth study, it is possible to conclude that the developers’ follow a stepwise formalisation approach since they do not start writing the actual code before the formalism is elaborated. However, they also break completely with the stepwise formalisation model since they allow reasoning at code level to settle how the formalism should look like.

The other interpretation is ‘so what’ this phenomenon is part of mundane problem solving following from the complexity at hand.

In the descriptive studies we cite above, respect for the observed developers is emphasised. The ethnographic work is characterised by a humble attitude. The researcher acts as a novice in the field in order to learn about it from the participants own points of view. The material gathered is intended to represent the field as seen and understood by those who live and work within it. This research focus leads to descriptions of software development practice, as the practitioners themselves perceive it. From this point of view the people observed are regarded as the most competent experts in their field. The consequence of the ethnographers' humble attitude and focus on revealing software development problems from the studied peoples own point of view is by definition doomed to end up in a lot of revealed and described ‘bad methods’ findings.

Let us compare how such ethnographic findings, and the ‘humble’ research attitude that lead to such findings, relate to a software engineering research attitude. In improvement oriented software engineering the goal of research is to find new methods, rules and perspectives to apply to software development based on the assumption that it is always possible to improve current practice. The latter ‘always possible to improve’ attitude implies that ‘an outside’ view is applied on the study in question, a view that does not easily fit in with the ethnographic aim of providing a description with a special emphasis on those observed. From the improvement oriented software engineering view practitioners are likely to talk in terms of ‘bad practice’ where suggested methodologies are not followed. New and better methodologies and structures have to be developed to handle complex work situations. With this attitude, software engineers are, by definition doomed to interpret ethnographers' ‘description of found methods problems from the studied people's point of view’ as a criticism of developed methodologies. That is, ethnographers' presenting ethnographic studies that over and over again reveal weaknesses in software engineering methods, i.e. weaknesses missed or not understood by software engineers' developing methods.

The studies provide evidence to support both the ‘bad practice’ perspective as well as the ‘bad methods’ perspective. It seems that either the practitioners are not following the methods or that the methods developed are not capable of handling real world problems. Of course, experienced researchers and practitioners are already aware of the problems in the field presented by ethnographers' descriptions. Probably most software engineers would categorise those described problems as normal exceptions in ordinary software development. Software engineers may also point to the fact that methods are thought of in terms of giving support, not as rules to be rigorously followed. In a methodological discussion relating to ethnography and design it has been shown that several studies that are primarily concerned with revealing
social phenomena sometimes are dismissed with precisely a *so what* attitude by designers (Plowman, 1995).

5. Conclusion

Ethnographers study what is actually going on in a community from the point of view of people being studied. Software engineering research aims at improving and thereby presenting possible improvement directions for the way in which software development is carried out. The different research attitudes lead to different interpretations of the original ethnographic descriptions of software development practice. One can either assume that ethnographic descriptions present ‘bad practice’ of methods usage which should be changed, or ‘bad methods’ not well suited for that must be improved. This is an issue that will probably never receive a clear ‘yes’ or ‘no’ answer as it depends too much on the applied views on the research material presented. Different existing implicit research attitudes among ethnographers and software engineers make a difference for the interpretation of ethnographic studies. One conclusion could be that ethnography and software engineering are in a sense doomed by definition to end up in the attitude and interpretation clash described here.

This prediction is to some extent counteracted by the author's own work (Chapter 5). However, quantification within software engineering implies the loss of what actually could actually be the most important contribution of ethnography: the inside perspective on software development practice. The quantification of ethnographic field material leads to results that are in agreement with the accepted discourse in software engineering, at the same time as it contradicts the original understanding of ethnography. The question that needs to be asked is, what will be missed by not taking in the discourse of original ethnography? The original descriptive ethnographic material provides not only an additional set of data but makes other aspects of software development visible; a different set of research questions leading to different answers might result. Conceptual concerns and theoretical reasoning, and lessons learned from ethnographic work are in danger of never entering the software engineering field. Ending the conclusion it is below suggested two possible ways of benefiting from to the original understanding of ethnography in the field of software engineering.

One suggestion is the *cooperative method-development* (Dittrich, 2002). The co-operative approach makes it possible to take the ‘inside perspective’ into account throughout the research process. In this approach, the ethnographic studies provide with an unbiased ‘inside view’ as the common ground for the development of improvements together with the practitioners involved. The implementation of these changes can then be studied in the same way. A set of consecutive cycles shows what parts of the method innovations can be implemented successfully and why others fail to be applicable. Experiences have shown that ethnographic methods change character when applied in the context of software engineering research. A software engineering researcher might be a novice in a specific software project, but she can never be a novice in the same way as an ethnographer in a different culture. These kinds of methodological concerns can only be explored if the researcher takes the attitude and underpinnings of both sides in this cross-disciplinary adaptation of methods seriously.

Another possible suggestion concerning how to use original ethnography is as a means of promoting methods development, to shed light on how methods are actually used in different
industrial contexts compared to what researchers' original aims were with regard to the methods developed. This means, to relate the original understanding of ethnography to a discussion of what is actually meant by methods in relation to methods following. From this point of view it is important to discuss what is meant by descriptions of how to do things in a method; is it a rule to be followed always? something that we most normally follow but in every single case? is it rather a goal or ideal to strive for, recognising that the strict following of methods is unpractical? It is only after discussing what is really meant by methods that it can be decided if the discrepancies between methods in use and researchers intentions with the methods exist or are a problem at all, and what does what mean. If ethnographic studies are regarded from this point of view they will not only give us a deeper understanding of practice but also of the relationship between practice and methods.

Acknowledgements
We would like to thank Prof. Claes Wohlin for providing valuable critique to an early version of this paper.

References


