International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

# The Role of Ethnographic Studies inEmpirical Software Engineering

Helen Sharp, Yvonne Dittrich and Cleidson de Souza

**Abstract**— Ethnography is a qualitative research method used to study people and cultures. It is largely adopted in disciplines outside software engineering, including different areas of computer science. Ethnography can provide an in-depth understanding of the socio-technological realities surrounding everyday software development practice, i.e., it can help to uncover not only what practitioners do, but also why they do it. Despite its potential, ethnography has not been widely adopted by empirical software engineering researchers, and receives little attention in the related literature. The main goal of this paper is to explain how empirical software engineering researchers would benefit from adopting ethnography. This is achieved by explicating four roles that ethnography can play in furthering the goals of empirical software engineering: to strengthen investigations into the social and human aspects of software engineering; to inform the design of software engineering tools; toimprove method and process development; and to inform research programmes. This article introduces ethnography, explains its origin, context, strengths and weaknesses, and presents a set of dimensions that position ethnography as a useful and usable approach to empirical software engineering research. Throughout the paper, relevant examples of ethnographic studiesof software practice are used to illustrate the points being made.

**Index Terms**— D.2.2 Design Tools and Techniques, D.2.14 Human Factors in Software Design, D.2.18 Software Engineer-ing Process, K.4.3.b Computer-supported collaborative work

## INTRODUCTION

Ethnography is a research method recognized as a sig- nificant qualitative empirical approach suited to un- derstanding people and cultures and their associated social and work practices [2]. Ethnographic studies are commonly performed in the Social Sciences [59]. In the context of Computer Science ethnography has been adopted within the CSCW (Computer-Supported Co- operative Work) and HCI communities to conduct studies of the workplace and inform the design of computer ap- plications e.g. [8]. Ethnographic studies are also used eve- ry day in the practical development of technology [133], specifically in the context of user experience design. In contrast, ethnography is hardly used at all in empirical software engineering. Proponents of empirical software engineering appear to have limited experience with eth- nography and hence there is little support in the literature for applying ethnography to Cleidson R. B. de Souza is with Vale Institute of Technology and the Feder- al University of Pará, Tv. Boaventura da Silva, 955, Belém, PA, Brazil, 66055-090. E-mail: cleidson.desouza@acm.org.

only tangential or partial treatment. Sjøberg et al [122], for example, do not mention ethnography in their discussion of the future of empirical methods in software engineer- ing. Kitchenham et al's [74] guidelines for empirical re- search mention observational studies but the discussion is mostly about experiments conducted in "in situ industrial settings". Similarly, Seaman [104] discusses participant observation, which is a central concept in ethnography, but does not discuss ethnography itself. Easterbrook et al [45] emphasize that ethnography aims to understand cul- ture (of a community, or an organization or a team), but they don't explore the potential that understanding such cultures may have for improving software practice. The key strength of ethnography that is overlooked by these and other empirical software engineering articles is the support it provides to explicate the rationalities of prac- tice from an insider's point of view – to capture both what practitioners do in which context, and why.

While there are some ethnographic studies of software practice, only a few of them were run by software engi- neering researchers. In addition to the lack of awareness and familiarity, applying ethnographic methods as part of software engineering research is challenging, partly be- cause ethnology and sociology – the homesteads of eth- nography – are descriptive and analytical disciplines, whereas software engineering, as an engineering disci- pline, is interested in improving the way software is de- veloped and built. Practitioners and fellow researchers expect empirical software engineers to discuss possibili- ties for improvement or new methods and technologies [30], [37], [115] and are puzzled by ethnography that ap-pears to focus only on understanding and describing a situation. Dittrich [34] reports how the software engineer- ing researchers' role was co-constructed by practitioners and management to be associated with change as well as research, i.e. they expected recommendations for change to be made.

The lack of software engineering researchers' familiarity with ethnography has been recognized before, e.g. in his contribution to the software engineering encyclopedia,Rönkkö [97] emphasises the differences in "style, mindset and expectations" between ethnography and software engineering. Ethnographic approaches can be misunder- stood or misapplied, leading to results being dismissed with a "so what?" response. In our experience, otherforms of field study such as observational study, case study and fieldwork are more

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
commonly used in empiri- cal software engineering. An ethnographic stance would add a different perspective to these approaches. Indeed, case study research is observational rather than transfor- mational, but Runeson et al [102] does not regard ethnog- raphy as a major research method, instead considering ethnographic studies as a specialized type of case study with a focus on cultural practices, or long duration stud- ies with large amounts of participant-observer data. We will address these aspects throughout the paper.

Empirical software engineering aims to improve soft- ware practice through evidence-driven research that evaluates or develops tools, methods, processes and other aspects of practice. Understanding software development practice and the rationale underlying it is key to this aim, not only so that any changes will be proposed in full knowledge of the context within which it needs to exist, but also in order for practitioners themselves to feel con- fident in the feasibility of proposals. Two of the funda- mental characteristics of ethnography support this aim. Firstly, ethnography takes an empathetic perspective, in which the researcher gains insight into social and work practices as seen through the eyes of those under study. Secondly, ethnography provides an analytical focus that allows the capture of not only what is done in practice, but also why things are done the way they are. This provides a valuable opportunity in the context of empirical soft- ware engineering, because capturing both the 'what' and the 'why' of practice provides a solid foundation for iden- tifying sustainable improvements. In addition, if software engineering researchers conduct the studies themselves, then valuable and practical insights can be achieved. Software engineers are in a unique position compared to social scientists or those "outside" the discipline [88] to be able to adopt the role of a participant observer and to feed back valuable insights and practical consequences into software practice [115].

In this article we argue that ethnographic studies con- ducted by empirical software engineering researchers would be both valuable and insightful. In addition, we explain why this is the case by illustrating different roles that ethnography can play in empirical software engi- neering research. Furthermore, and to facilitate a wider uptake of ethnography, we present a set of five dimen-

sions to be considered in the design of any ethnographic study, and specifically those aimed at software practice. These dimensions provide a practical framework to be used by researchers planning to adopt the ethnographic method. As such, they are one of the contributions of this paper.

The rest of the paper is structured as follows. Section 2 introduces ethnography: its history and applicability, its main features and the above-mentioned five dimensions of an ethnographic study. Section 3 focuses on four roles that ethnography could have within empirical software engineering, drawing on existing ethnographic studies of software practice to illustrate the roles. Section 4 discusses four questions relating to the use of ethnography in soft- ware engineering: addressing the significance of ethnog- raphy's analytical stance; how to know when ethnogra- phy is an appropriate research method; why an empirical software engineer should consider being the ethnog- rapher; and how to design an ethnographic study. Section 5 concludes the paper.

Throughout the paper we reference existing studies of software practice to illustrate the points being made. These studies are not always published within the soft- ware engineering community, were not necessarily con- ducted by software engineers, and in some cases re- searchers implementing ethnographic research do not report them as such including [31]. Despite that, all the studies we reference have had some level of impact in software engineering research and practice. In addition, eight key examples have been selected to illustrate the five dimensions of ethnographic studies and the four roles of ethnography within empirical software engineering. An overview of these eight is in Table 1; the rest of the paper will refer to these studies as Example <short name>, e.g. Example Testing.

## AN INTRODUCTION TO ETHNOGRAPHY

Ethnography has its origins in sociology but has been applied successfully outside its 'home' discipline. This section introduces ethnography: what is it? where did it come from? where has it been applied? Four main fea- ethnography are presented, and five dimensions across which ethnographic studies vary are introduced. These dimensions have been distilled from existing eth- nographic studies and related literature, and tailored for presentation to a software engineering audience in orderto facilitate the implementation of ethnographic studies by empirical software engineering researchers.

### 2.1 The origins of ethnography

The word 'ethnography' can be translated as 'writing (about) a culture'. The roots of ethnology and anthropol- ogy, where ethnography has been developed go back to the enlightenment period. With the 'discovery' of the Americas and development of trading relationships



around the world, knowledge about other peoples be-came valued. At the same time, understanding of human- ity and its variety of cultures and languages was growing [65].

Ethnography, as we know it today, goes back to Mali- nowski's research of an island outside Australia during the 1st World War [76]. The central tenet of this approachis to *describe another culture from a member's point of view.* That requires the ethnographer to become a member of the culture he is researching. In the process of becoming a member of the culture the ethnographer needs to

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

learn the language, the social norms and rules and the artefacts of the culture that is being investigated. In this process their own cultural norms are challenged. The understand- ing of and the ability to describe the new culture develop through these learning experiences. In this process, the ethnographer uses herself as an instrument. That is, she experiences directly, and captures the ways in which the foreign culture differs from her home culture. These dif-ferences are the central findings of an ethnographic study[1].

During the second half of the 20th century, with the growing importance of qualitative research methods, eth-nography became an approach to social research in gen- eral, not only of foreign cultures. Hammersley and Atkin- son [59] define ethnography as "… a particular method or set of methods. In the most characteristic form it involves the ethnographer participating, overtly or covertly, in people's daily lives for an extended period of time, watching what happens, listening to what is said, asking questions – in fact, collecting whatever data is available to throw light on the issues that are the focus of the re- search" (p. 9). Especially the Chicago School of Sociology

[10] started to use ethnography to research and under- stand urban sub-cultures and their rationalities. Further- more, the Chicago School of Sociology " … introduced a concern with work practices, with how work is carried out by social actors. This eventually led to the adoption of ethnography in the study of use, design, development, and deployment of computational tools" [42:60].

In the area of CSCW, ethnography became one of the sociological methods used to inform the design of com- puter applications e.g. [2], [24], [107]. This was led by Lu- cy Suchman's [123] seminal ethnographic work on Xer- ox's machine repair personnel. According to Anderson [2]

this was "a pivotal moment in the understanding of what social science might offer the design of interactive compu- tational systems". Note that Suchman was not the first one to adopt approaches from social sciences, but rein- forced the interest in them [42].

**Ethnographic studies outside sociology** Ethnography originated in sociology but it has been suc- cessfully applied and adopted in other disciplines too. In Information Systems and Organizational Studies, ethno- graphic studies are rather unproblematic to the communi-ties, although researchers who were concerned with the

design and implementation of information technology had to account for the role of technology in organizations. Traditional ethnography includes the description of tools of the trade, however, it does not address the influence of tools on the culture studied.

Ethnographic studies are part of HCI's methodological repertoire [93], and variants of ethnography are routinely used in research and practice to, for example, investigate existing activity and develop supporting technologies e.g. [13], [128], to co-design systems with users e.g. [67], and to evaluate interactive products and systems e.g. [17]. In recognition of the need to conduct studies under com- mercial and other time pressures, ethnography has been adapted to the time-pressured settings of product and software development, e.g. contextual inquiry [62] which relies on a two-hour apprentice-style session with the user doing his usual work in his place of work, and 'rap-id' ethnography [82] which is based on identifying key informants, using multiple observers and keeping a tight focus.

In the context of CSCW, ethnographic studies help to understand why collaboration and cooperation support is more problematic than anticipated. Observations of peo- ple collaborating have become a major source for under- standing the difficulties of designing support for them. Though the ethnographic studies themselves often do not easily translate to concrete designs of computer support

[41] they help to understand the details of the collabora- tive practices observed. For instance, the study of the London underground control room [60] showed how op- erators made their action visible to each other, thus pro- moting mutual awareness of relevant aspects of the on- going activity as a base for coordination. Later studies indicated that this mutual awareness is also relevant for

- software developers [125]. In the context of Participatory

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

[1]At this point you might be asking: "If we're trying to uncover the perspectives of our informants, why is it not enough to just ask them, e.g. through interviews?" There are two main reasons. First, people tend to say what they think their interviewer wants to hear. Second, they are inclined to create a rationalised account which may (or may not) reflect what is wanted, and to have selective memory. Both of these are natural and not malicious, butit means that using interviews on their own yields only a partial view of the picture, no matter how skilled is theinterviewer [96].

Design, ethnography has been widely accepted as a re- source in the design process [9].

Altogether the experiences from neighbouring disci- plines show that ethnographic methods can be successful- ly adapted to different contexts and used for 'unintended' purposes. Similarly to other qualitative research methods,it seems clear that applying ethnographic methods in software engineering requires a careful design so that they are implemented correctly and important aspects are highlighted in the results. By doing so, the deployment of ethnographic methods will provide a strong foundation to explicate the rationalities of software

practices and thus provide a base to develop and devise adequate processes, tools and techniques.

**Four main features of ethnographic research** There are four main features of ethnographic research: the members' point of view, a focus on the ordinary detail of life, the analytical stance, and production of 'thick de- scriptions' for academic accountability. These features

distinguish ethnographic studies from other observation- al research. In the following sub-sections, each one is de- scribed generally first, and then it is interpreted in the context of empirical software engineering research. Alongthe way, some important misunderstandings about eth- nographic studies are clarified.

The members' point of view

The *distinguishing* feature of ethnography is its focus on the informants' point of view, i.e., ethnography aims to understand what is, or is not, relevant, important, and painful for the informant. An ethnographer will take into account whatever the informant judges as important *ra- ther than what he thinks is important*. In the anthropological tradition, this can only be achieved by participant obser- vation: an ethnographer spends time working, discussing, participating, living or in more general terms, engaging with the informants. By doing that, she is able to get an understanding of what matters to the informant. This alsomeans that ethnographies traditionally take a long time tobe conducted – from months to years. The reason for this is partially explained in the previous section where we describe the context in which ethnography arose: the study of exotic cultures [42]. In this context, ethnog- raphers need to learn a new language, a new way of liv- ing, as well as learn to adapt to new food and health con- ditions which often resulted in ethnographers getting sick in their new "lifestyles". Because of all these aspects, eth- nographies took longer [84]. Nowadays, ethnographers are being employed to study informants that are not exot-ic, who are from the same culture, and live in similarconditions. Therefore, ethnographies do not need to last long anymore: in general, a couple of weeks is enough time to produce good results [84]. Assuming that studies require a long time to be conducted is a very common myth about ethnographic research.

In empirical software engineering, if the researchers al-ready have some understanding about the setting being studied, e.g. since they understand software develop- ment, then ethnographies might be conducted in shorter periods of time. For instance, in Example Agile the ethno-graphic study of agile practices was conducted over aperiod of three weeks, with one week of full-time obser- vation followed by two one-day visits over the following two weeks. This period was chosen in this case because the agile team under study had a three-week iteration cycle and hence three weeks was a significant 'chunk' of development time.

The fact that people from a culture can, and often do, perform ethnographic studies of their own culture is bothan advantage and a limitation. It is an advantage because

it allows the ethnographer to grasp more quickly what matters for that culture since she can use prior knowledgeto understand what is taking place. On the other hand, the ethnographer must be aware of his own assumptions and biases. He needs to recognise and take account of these in order to understand the important aspects of the culture being studied. Ethnographic researchers talkabout 'bracketing' their assumptions (i.e. putting assump- tions aside until later) until they are confirmed (or not) with respect to the specific context in which the study takes place. There are techniques that can be used to bracket assumptions and avoid biases, but they are out of the scope of this paper.

The ordinary detail of life as it happens

Ethnographies *often focus on the ordinary detail of life* [2]. In other words, ethnographers are interested in all the de- tails of what members of a culture do in their daily ac- tions, since a culture is enacted through these details. As with the Chicago School of Sociology, empirical software engineering is concerned with *working* life, i.e. in software developers' daily work achievements. This has two im- portant implications. Firstly, ethnographers will initially collect several types of data about different aspects of their informants' work because they do not yet know what is or is not important. Secondly, even though an ethnographer has a research focus to address, she must keep 'open' for new possibilities.

As more time is spent at the site, the ethnographer will gain a better understanding of the informants' work, in- cluding concerns, frustrations, expectations, preferences and the like, and therefore will be able to identify aspects that might be more interesting,

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
relevant, and worth ex- ploring for the informant than the original research ques-tions. So because an ethnographer is aiming to see the world from the members' point of view, data collection plans and expectations need to be flexible. Rigidly stick- ing with a plan for the day which ignores changes as they happen 'on the ground' is not productive. This flexibility is often confounded with lack of rigour, a second myth about ethnographic research. Fetterman [49:1] describes this aspect of ethnography very wisely when he argues: "Ethnographers are noted for their ability to keep an open mind about the group or culture they are studying. This quality, however, does not imply any lack of rigor. The ethnographer enters the field with an open mind not an empty head."

In the context of empirical software engineering, Dit- trich and Giuffrida [36] describe an ethnographic study where they were initially focusing on the usage of social software tools by software developers, but then their fo- cus evolved. They started looking *not only* at these tools, but more importantly at how they were used in the con- text of an ecology of communication channels used by software developers, including issue-tracking systems, video-conferencing and screen sharing. The importance of attending to the ordinary details of life lies in the role these details play in relation to how the everyday tasks are addressed. In other words, it is important that the

activities be seen in the real context: who, why, what and how of the moment, rather than be abstracted away into constrained empirical studies.

The analytical stance

Ethnographers are not journalists who solely report what they have observed in a particular situation. Conducting an ethnographic study, especially when writing up its results, involves *the interpretation and analysis* of what was seen, heard, felt, and found in the field [2], [42]. Anderson

[2] is very clear in this point when he illustrates how Ma- linowski described the "Kula Ring":

"The centrepiece of this work [Malinowski's work] is a famous description and explanation of "The Kula Ring"; that is the practice among Trobrianders of sailing hun- dreds of miles across dangerous seas simply to ex- change in very strictly ritualised ways apparentlyworthless amulets and necklaces. What on earth are the Trobrianders up to in doing this? By dint of his detailed descriptions of what is exchanged, with whom and how often; how the amulets and necklaces circulate around the islands; what the required formulae are; what asso- ciated activities may be carried out in association with Kula; what magic governs the journeys; and so forth, Malinowski demonstrates how Kula promotes social in-tegration. In carrying out the Kula, Trobrianders, willy-nilly, are solving one of the central problems of all socie- ties. The function of the Kula is social cohesion." [2: 7]

The analytical feature of ethnographies is fundamental and powerful, and has been overlooked by research methods articles in empirical software engineering. Infact, this feature has also been neglected by researchers inhuman-computer interaction [14]. In other words, there is a misconception that ethnography is purely descriptive; that it is only concerned with the description of what has been seen in the field. This is not correct. A true ethnog- raphy will not only present the evidence acquired in the field, but will provide an analysis of the results explain- ing how this evidence is, or is not, relevant for a particu- lar purpose.

Example Awareness demonstrates this analytical aspect of ethnography. de Souza et al [30] discuss the roles of application programming interfaces (APIs) in the coordi- nation of software development work in a large-scale or- ganization. This account reports what was seen during the study, but most importantly, it is reported in an ana- lytical way that *demonstrates* how APIs are artefacts that can facilitate or hinder the coordination of large software development teams. So ethnography not only provides a detailed account of how software development takes place, but also highlights the local rationalities of the de- velopers' actions and behaviour, i.e. it explains why things are the way they are, from the community's point of view. The rationalities are vital because they provide the deep insight upon which future improvements can be

certain notations are used for certain tasks; why are dif- ferent communication channels in a distributed project team used for different situations; why is an office laid out in a particular way even though it impacts collabora- tion; and so on. An ethnographic analysis in empirical software engineering needs to extend findings into in- sights that can inform the improvement of software de- velopment practice through tools or processes, although not all ethnographic work looking at software practice has achieved this.

Thick descriptions for academic accountability

The result of an ethnography is often more comprehen- sive and detailed when compared to results using other research methods. This is because it aims to communicatethe broad picture. This comprehensive and detailed set ofdata is often referred to as a "rich picture", or a "thick description" [53] of the community and culture studied. This assures that the results obtained with an ethnogra- phy are realistic[2], have high internal validity, but, on the other hand, have weak external validity, i.e. do not

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

neces- sarily generalize to other contexts [80]. This causes some empirical software engineering researchers concern as they are seeking generalizability. However, the richness obtained through an ethnographic study allows a detailed investigation of the situation that may yield significant insights that can be transferred to other 'similar' contexts. For example, Ferreira et al's [48] findings emerged from ethnographic studies in just three organisations, but the resulting list of guidelines has been recognised as useful in several other contexts, e.g. [43]. In other words, because of its analytical nature, the results of an ethnographic study are suitable for analytical generalization, but not suitable for statistical generalization.

As Anzai and Simon have commented [3]: "[Even if]
one swallow does not make a summer, ... one swallow does prove the existence of swallows. And careful dissec- tion of even one swallow may provide a great deal of reli-able information about swallow anatomy"

**Five dimensions of ethnographic studies** Especially with the adoption of ethnography outside of its original sociological context, a diversity of ethnographic research designs have emerged. There is not a single right way to do ethnography; the following is a key set of fivedimensions along which ethnographic studies differ.

These dimensions have been distilled from existing eth- nographic studies and related literature, and tailored spe- cifically for a software engineering audience. The balance of dimensions chosen for any one study will be influ- enced by the community or practice to be studied, and by the research question to be answered. This means that these dimensions will provide a starting point for anyempirical software engineering researcher interested in
built. An ethnographic study in empirical software engi-
neering will aim not simply to observe activity but to de- termine rationalities and explanations: what causes frus- tration, stops progress, or makes the team work well; why
Realistic in as far as the situation or context within which the evidence is gathered is comparable to the con- texts in which you want your evidence to apply [80].

conducting an ethnography study. Table 2 maps thesedimensions to the eight example studies used throughoutthe paper.

TABLE 2
DIMENSIONS OF ETHNOGRAPHY MAPPED TO EXAMPLE STUD-
IES

Participant or non-participant observation

Observation is key to ethnographic studies and both par- ticipant and non-participant observation are legitimate forms of ethnography. Observation is almost always complemented with other forms of data collection such as interviews or document analysis. The role the researcher takes with respect to the observed community will partly define which areas of the community can be accessed.

Participant observation involves the researcher effec- tively performing the same actions as the informants [69]. In contrast, non-participant observation involves the eth- nographer observing the actions, but not necessarily do- ing them as well [69]. Participant observation can be im- practical in a work-based situation such as software engi- neering as there are practical concerns about being able to perform all activities a professional software engineer would do in the field site. Participant observation in this context is more often interpreted as taking on a meaning- ful role within the community and engaging with the community's everyday business. For instance, when ob- serving software engineering practices, companies are very wary of letting non-employees access their source-code repositories, although some kind of participation is usually necessary in order to keep up with a fast moving project. In this case the researcher might take over docu- mentation tasks [129] or support the project through ad- ministration [75].

The duration of the field study

Traditionally, an ethnographic study in ethnology would require a long-term participation in the field and extend- ed visits with the community studied, often abroad, and often for many months. The adoption of ethnography by HCI and CSCW researchers led to the acceptance that both long-term and short-term ethnographic studies could provide important insights and meaningful input for design [84]. Examples of long-term ethnography in software engineering include Low et al's [75] 15 month- long study (4 days a week) of a water utility company and Prior et al's [94] 20 month-long study (45 days in to- tal) of a group of professional software developers. An example of short-term ethnography in software engineer- ing is Example Agile, which consisted of 7 days of study over a three-week period. These examples illustrate that the researcher does not need to be located

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
full-time at thestudy site, but that fewer days may be spent at the site, allowing time to start analyzing the data during the field study time. This supports development of the analytical stance and allows for flexibility (see previous section).

Space and location

This dimension focuses on where you can find and study
your community. Ethnography is traditionally perceived as a method practiced in a delimited geographical space by engaging in face-to-face interactions. However the internet and globalisation have changed that, challenging ethnographic anthropologists and sociologists research- ing topics such as globalizations, migrations, nationalism and other issues not typically present just in one single site [77]. In global and open source software projects, online environments play a crucial role, and the concept of a field site changes: alongside the physical workplaces, the virtual environment needs to be regarded as a field- site e.g. [61], and online observation will be required [91]. In these settings, the ethnographer cannot be physically present in different sites at the same time [92]. To support this situation, Marcus [77] developed the practice of mul-ti-site ethnography for studies that involve more than onefieldsite and where the ethnographer moves between fieldsites following people, artefacts, metaphors, the plot, life and conflicts. As an example in empirical software engineering, O'Riain [89] investigated software develop- ers in Ireland and Silicon Valley deploying an approach that conducted ethnographies wherever key people were to be found. An alternative strategy is to choose one site from which to study participants, and another is to follow the traces of communication, physical or digital artifacts, and emails, instant messaging logs and intranet blog posts [54]. Most researchers will apply a hybrid approach
– observing real settings, participating in the virtual envi-ronment and collecting and analyzing documents of dif- ferent kinds [68], [109], [119].

The use of theoretical underpinnings

Ethnographic research does not come with a specific the-oretical underpinning that should be used. However, there are several that are commonly used. Table 3 pre- sents the theoretical underpinnings most used in softwareengineering and related scientific discourses. These theo-ries act as lenses that focus on specific aspects of the field while de-emphasizing others. This means that a theoreti- cal framework strongly influences not only how data analysis is conducted, but also which data is collected. Forexample, an Activity Theory-informed ethnography would use the notion of *activity* as the scope for observa- tions and focus on the central elements of an activity sys- tem identified by the theory: actors, objects and outcomes, tools involved, written or unwritten rules governing the collaboration, the community the actors belong to, and division of labor involved in achieving the intended out-come. In constrast, ethnography using a distributed cog- nition underpinning would focus on how the environ-ment, including other people, artefacts and the physical world, supports an activity through the exchange and transformation of information. More details, including examples are in Table 3.

While some ethnographic studies of software engineers do adopt these frameworks, e.g. Martin et al [78] and Rönkkö et al [99] apply an ethnomethodological under- pinning, others, e.g. de Souza and Redmiles [29], present their research without adopting a theoretical framework
during data collection or analysis.

Whether or not to use a theoretical underpinning de- pends on the research question and the ethnographer's intent (see below). As ethnography applied in software engineering focuses on understanding software engineer- ing as social practice and the rationalities of practice, a group of theories categorized under the term 'practice theories' [87] offer themselves as most compatible with the main features of ethnographic research (section 2.3 above). This family of theories regards social practice as constitutive of social organisation and structures. So far, only a few contributions towards a software engineering specific theoretical underpinning have been published.For example, Dittrich [35] suggests initial steps to adopt and adapt practice theory to the software engineeringcontext.

TABLE 3

THE THEORETICAL UNDERPINNINGS MOST USED IN SOFTWAREENGINEERING AND RELATED SCIENTIFIC DISCOURSES

*The ethnographers' intent in undertaking the study* Traditionally, ethnography investigates an unfamiliarculture and aims to understand it. In this case, how thatunderstanding is acted upon, if at all, was not part of theethnographers' intent. However in the early days of eth-nography, there were some cases of commercial exploita-
tion of the understanding achieved through ethnographic studies, e.g. with the results being used for targeted mar- keting. To guard against this and other misuses, many proponents of ethnography today reject the notion of eth-nographic results being used to exploit or change the cul-ture it observes [56]. However, in the context of CSCW forinstance, ethnographic studies are

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
intended to provide not only an in depth understanding of a work practice and organizational culture, but also support for the de- sign of specific functionality which may then change ways of working.
In the context of software engineering, it may be valua-ble simply to understand and capture a description of practice. For example, if a community-wide practitioner- driven change has occurred, such as the adoption of agile methods [110], then explicating and disseminating that practice can be informative for empirical researchers;where programmes of reform are driven by governmentor industry standards, such as software quality manage- ment systems [63], then understanding why they are or are not accepted by practitioners can be informative for researchers and practitioners. In these cases, those other than the ethnographers themselves, including the practi- tioners under study, may use that information to im- prove, evaluate or modify practice.
On the other hand, simply understanding practice may not satisfy all empirical software engineering researchers, and may also puzzle those under study. For empiricalresearchers, understanding observed practices is only thestarting point to improve them, maybe through evaluat- ing methods, or developing tools and so on. Practitioners being studied may expect help with improving their prac-tices, or addressing their problems and be puzzled if the
researchers do not help them to improve, as they are per- ceived as peers with access to relevant knowledge about methods and tools.
The intent of the ethnographers will change the interac- tion between the researcher and the field [34]. It is there- fore important that the researcher both is open about her research interest and her intent and also takes into ac- count the way her research might impact the observed software development practice.
Designing the ethnographic study
When designing an ethnographic study the dimensions discussed above need to be taken into account. These di- mensions are not independent, and it is common for one to impact on the other. Location and space, for example, might define intensity and duration of a study. Observing an online community such as in Example PyPy might not require full time observation even over a short time, but ifthe observed community meets for a sprint, participatory observation might require travelling and participation foras long as the meeting takes place. We will address this aspect in more detail in section 4.4.
Some researchers and practitioners have crystallized certain balances of dimensions and condensed the result-ant adaptation of ethnography into an identifiable ap-proach. Examples include rapid ethnography [82] and contextual inquiry [62], mentioned in 2.2 above, and cog- nitive ethnography [5] This last one is tailored for goal- directed development of technology through data collec- tion in timeslices of observation, tight focus and verifia- bility through multiple data sets. Making such compro- mises is not easy or straightforward, and there remain many challenges to designing an ethnographic study for software practice [90]. It is important that any adaptation or design does not compromise the central idea of eth- nography, which is to uncover the participants' view- point.

ETHNOGRAPHY IN SOFTWAREENGINEERING
Some of the most well-known research using ethnogra- phy in software engineering tried to integrate ethno- graphic studies into software engineering as a techniquefor requirements engineering [131]. This paper takes a different stance, it focuses on *ethnographic studies as a tech-nique to study the community of software engineers and im- prove the way in which they work*.
The first studies using ethnographic methods to investi- gate software practice, conducted from a software engi- neering perspective, were implemented and published in the mid 1990s, often by interdisciplinary groups of re- searchers including software engineers and sociologists
e.g. [63], [75], [121]. For instance, Example Bug Report focused on the concept of a coordination mechanism and has been influential in global software engineering recent-ly (see Table 2). Another example is the work ofHovenden and colleagues [63], who describe a study of quality management that lasted just one week, and was

largely non-participative.
This section presents four different roles for ethno- graphic studies in empirical software engineering. The discussion is illustrated using the eight example studies introduced above; the way in which they relate to each ofthese four roles is shown in Table 4. These four roles are:
To strengthen investigations into the social and hu-man aspects of software engineering
To inform the design of software engineering tools

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

To improve process development
To inform research programmes by articulating morespecific research questions and complementing other research methods, such as code analysis and quanti- tative studies, thereby providing context grounded inpractice
One study or series of studies may address more thanone of these roles.
TABLE 4
THE ROLE OF ETHNOGRAPHIC STUDIES MAPPED TO EXAMPLESTUDIES

To strengthen research on social and humanaspects of software practice
People and their interactions are central to software de- velopment, and the significance of the social and human aspects of software practice is well-established. In the early 1970s Weinberg [134] highlighted the importance of social interaction in code development, Nygaard [88] proposed a set of dimensions of 'Program Development as a Social Activity', and later, Floyd [50] emphasized the importance of focussing on the software development process as it unfolds in reality as a basis for supporting developers with adequate tools and techniques. Other notable writers were: Scacchi who based software engi- neering tool development on empirical studies and find- ings from sociology [7], [81]; and Naur [85] who ques- tioned the value of documentation as a sole communica- tion tool, and emphasized the need to directly communi- cate the rationale behind the design.
Research on the human and social aspects of software development has quite a broad scope, but its key charac- teristic is that it covers any aspect that is related to people involved in software development, or their interactions [32]. For example, psychological aspects of individual developers e.g. [33], [103], developer behaviour e.g. [135], teams e.g. [6], users e.g.[1], and so on, would all be con- sidered under this theme. Given its focus on culture and the informants' perspectives, it is not surprising that eth- nography can strengthen investigations of the social and human aspects of software practice.
Research in this area often relies on interview data. In- terviews are a good approach to use because they can elicit data from people relating to their thoughts and ra- tionales, which are difficult to obtain through other means. When used on their own or with other self- reporting instruments such as questionnaires, they rely on the informant's account of events, and any such ac- count is necessarily partial and rationalized [96] (see also
footnote 1 above). Ethnography's focus on daily activity as it unfolds in the real context helps to capture a holistic view of key social and human events relevant to the study of software practice, and hence complements inter- views. Together with ethnography's emphasis on the in- formants' perspective, this approach overcomes any limi-tations of using self-reporting instruments alone. Ques- tions in this area which might be answered using ethnog- raphy include *How do developers manage to produce quality code in a complex environment? What is it important not to change? Why do problems, conflicts and successes occur? What is behind compliance and non-compliance with espoused meth- ods?*
Example Agile indicates one set of studies where an ethnographic approach has been used to explore the so- cial and human aspects of software engineering, in par- ticular focusing on team interactions. A key finding from these studies was that the social behaviour of teams helped to enforce the disciplined use of supporting arte- facts [111], [112]. This finding was based on a number of individual studies in different settings, all with XP (ex- treme programming) practitioners and a range of under-pinning theories including distributed cognition [66] and cognitive dimensions [57]. One of the consequences of this finding for practice is the need to compensate for so- cial discipline when the social context of the team chang- es, e.g. through distribution [109]. Another consequence is the need to consider the impact of virtual rather than physical artefacts [108].
Several of the Example studies' results show that soft- ware development is a social process as much as a tech- nical process, e.g. Example Coordination and Example Bug Report, and there are several other ethnographic studies that exemplify this role. For example, Dittrich and Lindeberg [38] identified a rich set of practices that pro- mote user-developer cooperation. Their findings particu- larly highlight how the dynamics of a participatory and evolutionary process can be accommodated while at the same time making the process accountable and managea- ble from an organizational perspective. Avram et al's study emphasises the role of ethnography in uncovering the rationale for certain practices [4]. They used partici- pant observation, among other methods, to investigate workflow at the project level and at local level, within adistributed team. Their results confirmed that imposing processes or workflows does not necessarily support the organization in meeting its goals, and that local arrange-ments may be supportive rather than obstructive.
In short, ethnographic studies of software practice fo- cused on social and human aspects can expose the mech- anisms used to

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

make things work, explicating practices that may be taught and shared, explanations that allow key activities or artefacts to be protected, and potential problems to be detected early.

### 3.1 To inform the design of software engineeringtools

The relationship between ethnographic research and de- sign is by no means a simple one. In fact, it has been dis-

cussed in the CSCW community since its very beginning. For instance, in 1994 Hughes et al [64] suggested four practical strategies to using ethnography in design. As an example, one of these approaches is called concurrentethnography, which means that the ethnographic investi- gation of the work and the systems design proceed in parallel (see Figure 1). Debriefing meetings between eth-nographers and designers are held to "identify, discuss, and elaborate issues of relevance to design" [22: 91].

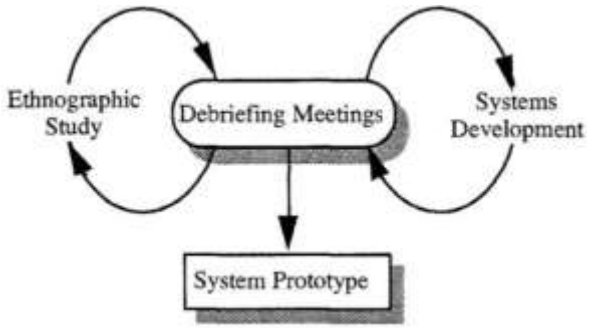


Figure 1. Concurrent Ethnography and Design

Other authors have also discussed this relationship. Anderson [2] concludes that "Ethnography has been drawn into the circle of design. [...] The task we now face is to try to understand what the incorporation of this andother modes of social science might mean and just what their investigative methods can contribute. I do not be- lieve this is a straightforward, quick, or painless process."Dourish and Button [42] highlighted an issue which is crucial to software engineering research too: can a de- tailed account of how some work takes place be used to provide 'design recommendations' for the software that will inevitably also change the work on which the study was based, i.e. the software that will affect the culture andthe practice once it is adopted (as discussed in section 2.2)? The debate has continued for several years and Crabtree et al [23] caution designers "to carefully consider the contributions different approaches <to ethnography> make to design. They are not all the same and widespread adoption of new approaches may undermine the practical relationship ethnography has developed with design as the computer moves into new contexts."

However, it is important to have in mind that this dis- cussion assumes that ethnograhers do not possess design skills [22:95], and therefore, that ethnographers and de- signers are different stakeholders. This translates the rela- tionship between ethnography and design into an issue of information exchange and communication, i.e., how can the ethnographers report, in a meaningful way, to the designers? And, how can designers address the important aspects of the work highlighted by the ethnographers?

When conducting an ethnographic study of software practice from a *software engineering perspective*, these roles are not different, i.e., the ethnographer and the designerare one and the same person. Example Awareness illus-

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

trates this aspect. The initial study showed that APIs play multiple roles in collaborative software development: they are at the same time contracts between developers, reification of organizations' boundaries and communica- tion mechanisms, therefore both facilitating and hinder- ing the coordination of software development teams. This resulted from an analyisis of the data collected, i.e., this example also illustrates the analytical aspect of the ethno- graphic work (see section 2.3). This means that building a tool to support these results should not focus on the data *per se,* but instead on the *analytical results*.

In Example Awareness, the same authors built and evaluated the tool Ariadne [26], [127]. This illustrates the use of a typical software analysis technique (in this case dependency analysis) to facilitate the coordination of complex software engineering projects. Results of the de- pendency analysis are used to support the identification of software developers who are working on similar is- sues, developing redundant parts of the code, and so on. The study was one of the first to establish a research focus on socio-technical dependencies in software engineering and how to manage them. Later on, Cataldo and col- leagues [20] extended this approach by focusing on (un)matched coordination needs.

The explicit link between an ethnographic study and tool development is not always made within one piece of work, but it is more common for an ethnographic study to identify areas requiring new tools, or to explain how the existing tool is used and hence how it might be improved: Example Awareness, for instance, focuses on the ethno- graphic analysis while the results of the tool construction and evaluation are presented in other papers [26], [127]. Similarly, Boden et al [11] studied two teams in small software development German companies and later on implemented the concept of Articulation Spaces [12] to address some of the coordination problems they had ob- served in these companies.

Being a software engineering researcher and an ethnog- rapher in the context of tool design is not without prob- lems, however. The most important problem to be ad- dressed is the need to be cautious about when to start designing, or better yet, not to start designing too soon. In other qualitative research, hypotheses to be tested in one data collection step may be created in the previous one. In the beginning of the process, software engineers then should avoid making design decisions or even imple- menting something, when it is unclear which of these hypotheses is true. Instead, the researcher should hypoth- esize about tools, or features, that could be useful for software developers and seek ways to validate them with the informants. By this, we mean collecting additional data to support, or reject, the hypothesis that a particular set of designs would be useful for the informants. Fur- thermore, the appropriation of the design will provide additional information about which aspects of the sup- ported practice interact with the task the tool is designed to support. Another risk associated with early design de- cisions is giving too much emphasis on the data *per se*

instead of the focus on the analytical results. For instance, de Souza and colleagues [31] faced this situation during one of their studies. They observed a situation in which the versioning systems and email could be "integrated" to facilitate the studied team's work. However, as the re- search progressed they realized it was more important to focus in the overall communication among the different teams. Combining these two tools would be only one lim- ited part of the solution. Again, they reached this conclu- sion by hypothesizing about solutions and tools, and later on validating them in following data collection periods.

### 3.2 To inform software process and methodimprovement

Designing tools for software development is only one way in which software engineering aims to improve software development practice. Methods, processes, and techniques also package research results for practitioners. Ethnographic research provides a unique opportunity to better understand the interaction between methods and the situated context of their deployment and to use that understanding to improve the practice at hand, as well as the methods and processes adopted. If the researcher be- comes involved in the deliberation of change and the ex- ploration of methods, though, the role of the researcher changes from a pure observer to an actor and sometimes even a change agent.

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

The way methods are used informs practice in a differ- ent, less visible and less accountable form than tool de- sign and usage. This has been discussed in research fo- cussing on cooperative aspects of software engineering,
e.g. [51] and [79]. As other ethnographic research has shown, the application of methods is not as straightfor- ward as many software engineering textbooks imply [15]: methods, processes and techniques, despite their pre- scriptive nature, are still subject to interpretation in the specific context. Applying ethnographic research to focus on the situated implementation and adaptation of meth- ods, processes and tools in software practice leads to an understanding of how methods, processes and techniques actually influence that practice, what specifically influ- ences the interpretation and appropriation of methods, what makes them work, and what prohibits their applica- tion. Two examples that illustrate this role of ethnogra- phy are a study by Dittrich and Lindeberg looking at use- oriented software development [38] and Rönkkö et al.'s study on the applicability of personas in specific circum- stances [100]. The former is an example of how different processes and techniques together facilitate a participa- tory design and development process. The latter indicates how external factors impact the applicability of a method, in this case the use of personas.
The rationality of engineering disciplines, though, aims not only at understanding but also improving human affairs by developing technology. Software engineering researchers are interested in developing and evaluating methods, processes and tools as practitioners are interest- ed in improving their practice by applying these innova- tions. This sometimes leads to the situation that software
practitioners as well as their managers expect recommen- dations and improvement when engaging with software engineering researchers implementing an ethnographic study. For example, in Example Scientist, the software engineering researcher was invited to observe the project specifically in order to provide a reflective account and help the team to solve communication and cooperation problems.
This expectation influences the participants' interaction with the researcher, and this may be negative as well as positive. For example, the researcher might be perceived as a managerial agent who will provide recommenda- tions to improve control. In turn, the practitioners in- volved might hide aspects of their practice they do not want to have documented for management. One way to address this situation is to be explicit about the intention of the research and the interaction with the involved stakeholders.
In response to this, Dittrich and colleagues adapted ac- tion research as a framework for making the research and intervention accountable both towards the research com- munity and the research setting [34], [40]. Their Coopera- tive Method Development is based on five guidelines to keep the member's point of view and the focus on the day-to-day activities when deliberating, implementing and evaluating improvements:
an action research cycle consisting of three phases: empirical studies, deliberation of changes, and intro- duction and research of the changes decided;
the empirical research is ethnographically inspired;
focusing on the everyday shop floor software devel- opment practice ;
improvements are deliberated and evaluated from the shop floor software development perspective; and
the changes are deliberated and implemented togeth- er with the practitioners.
Example Architecture illustrates this role for ethno- graphic studies in empirical software engineering. The study focused on the processes and practices around agile architecture, and applied the above-described Coopera- tive Method Development approach [40]. The study pro- vided an understanding of the architectural practices that were used in this context, and showed how heavyweight architecture practices that focus on comprehensive docu- mentation might not be suitable in a context where the architect needs to be aware of how the code architecture is evolving. It proposes and evaluates lightweight archi- tecture practices and methods that are in line with the software development practices observed. More widely, Example Architecture shows that when proposing and introducing methods and tools to support architectural practices, the social practices that are to be supported by the tools need to be adapted explicitly.
To inform research programmes
Ethnographic accounts have the potential to highlight under-researched areas of software engineering, to identi- fy more specific research questions, and to complement

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
other research methods by providing context for them. For example, Example Agile highlighted the importance of physical artefacts plus social practices to maintain dis- cipline in agile software teams. In this case, the role of physicality in certain tasks had been researched within CSCW, but the findings there had not been translated or applied in this "new" context, software engineering. Questions remained regarding what is lost and what is gained when teams need to become distributed or dis- persed, and hence physical artefacts are not feasible [109]. An ethnographic study may form the core of a research project or programme of research (as described in Sec- tions 3.1-3.3) or it may be used to generate new research questions or complement other methods by providingcontext.

To articulate more specific research questions

Because data collection plans need to be flexible within an ethnographic study, and more importantly, because of the focus on the informants's point of view, it is very com- mon for such a study to identify more specific research questions once it is underway. These questions may be investigated within the same ethnographic study or through other methods of data collection and analysis as a separate study.

Example Scientist was originally aiming to investigate cooperation problems, believed to be caused by the dis- tributed nature of the project. It was hoped that formal and informal discussions prompted by the researchers would support team members to reflect on their ownpractices. However this initial focus on distribution was overridden by the research analysis which pointed to con-flicting cultural values and practices between the profes- sional software developers and the scientists as scientific end-user developers. The analysis of Example PyPy be- gan with an industrial case study looking at how Open Source teams handle the challenges of distributed devel- opment. This initial study led to a more specific focus on how changes to common practice are negotiated, and im-plemented in order to address these challenges.

To complement other research methods

Ethnography may be the main research approach taken, or it may be used to complement other methods. When used as a complementary approach, ethnographic studies can bring context to more quantitative findings, or other qualitative investigations. For example, Capiluppi et al

[16] report a quantitative analysis of code evolution that was developed by one of the Example Agile teams. Through the deep understanding gained of the team un-der study, useful contextual information was available tohelp interpret the quantitative findings. Other contextual information such as team facts and figures, practitioners' descriptions, organizational characterisations etc usually collected through case study research also provides a use- ful context, but it is of a different nature.

Instead of collecting just measurements or reported ac- tivities, the ethnographic investigation reveals how the measurements come about and what the reported activi-

ties look like in practice. Observations may also support making connections between various data sets. For in- stance, when investigating the use of instant messaging indistributed software engineering [55], observation helped understand the relationship between what happened on the instant messaging channel and what happened on other channels like the issue tracker, audio calls, and e- mail.

Several of our Examples illustrate the use of ethnogra- phy to complement other research methods. Example Bugreport form used interviews and elements of case study research; Example Coordination complemented ethnog- raphy with interviews and document analysis; and Ex- ample PyPy used open-ended questionnaires.

With the member's perspective, ethnographic studies contribute an understanding of the rationalities of the observed practice. This facilitates the capture of not only deviation or adherence to methods, but also an under- standing of why experienced practitioners might deviate.Based on ethnographic studies and complementary inter-views, Example Architecture proposes that there might begood reasons why software architects do not use architec-tural documentation: they are afraid that they will not be kept up-to-date about the state of the product if team members can access the documented architecture without consulting them.

## DISCUSSION

The examples throughout this paper illustrate what can be achieved when ethnography is used to investigatesoftware practice, but are these achievable uniquely through an ethnographic approach? It is not possible to state *categorically* that the findings gleaned through an ethnographic study *cannot* be uncovered using another empirical method. However it is extremely unlikely that such a deep, rich and detailed understanding of practices and their justifications will be forthcoming unless an eth- nographic approach is taken.

Why the analytical stance matters

An ethnographic study does not produce just an account of activity; it also supports the generation of insights and consequences (through the analytical stance). The core of ethnography in software engineering is to investigate the everyday activities of software engineering practice and to articulate the rationalities of those activities from a members' point of view. An ethnographic study provides insight into the fine-grained activity through which soft- ware practitioners achieve useful and usable software in an imperfect environment: what difficulties they are fac- ing and what, for them, has proven beneficial to

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86
address these difficulties. This fine-grained activity often only becomes visible when it becomes problematic, e.g. when coordinating software engineering practices fail over dis-tances [25].
Understanding the fine-grained activity of software practice, and its rationalities increases the chance that improvements in software practices built on these find-
ings will be sustainable, because they take account of lo- cal conditions and embed existing local expertise. Specifi- cally, this understanding helps to: identify what not to disturb when introducing new methods, tools and tech- niques; explain what has happened when things go wrong; uncover the local adaptations necessary in order for methods to be adopted and accepted; design tools to address some of the issues identified; and appreciate the ongoing changes applied to keep projects working suc-cessfully.
*What not to disturb*: In Example Architecture, the reason for not adopting recommended software architecture documentation practices was that they interfere with the software architects' practices that keep themselves up-to- date with emerging issues and concerns that might re- quire changes to the considered software architecture.
*Why things go wrong*: Example Agile provided insights into the role of physical artefacts in a co-located agile software development team, and how the move to a dis- tributed setting disturbs this role. Damian et al. [25] also show that a very little difference in the way a distributed team used the set of tools rendered locally well-established, awareness-creating, coordination mecha-nisms useless when moving to a distributed setting.
*How methods are interpreted and adopted locally*: Button and Sharrock [15] report the appropriation of methods and techniques beyond what the method authors would un- derstand as a proper implementation of this method. However, the appropriation takes place for good profes- sional reasons. Rather than being read as a sign of incom-petence, these appropriations can be seen as a reaction tothe mismatch of the rationalities of practice and the method to be applied [98].
*How to design software tools:* Example Awareness illus- trates the different roles played by APIs in the coordina-tion of software development work. Furthermore, its analysis suggested the usage of dependency analysis techniques, a traditional *technical* approach, to uncover *social* aspects that would be relevant to collaborativesoftware development practice.
*How projects are kept on track*: Example PyPy investigated how distributed development projects, private as well asopen source, change their practices in order to react to contextual changes or to address problems identified by the project members. Similarly, Avram et al. [4] show how a commercial distributed team adjusts its local prac- tice to the common infrastructure in order to "keep the local work flowing". Cohn et al [21] show how software engineers negotiate the boundaries of the project defining what is and what is not part of it.
When is ethnography appropriate
Ethnography is not the only, nor the best qualitative ap- proach to be used in all circumstances, but the more thor- ough understanding of software practice that ethnogra- phy brings has great advantages. For example, software engineering research often develops methods, tools and techniques that are designed to improve practice, but are rarely adopted. Why is this? There has been much debate
concerning the evidence required by practitioners to adopt research outcomes, and to adapt their practice e.g. [44]. Although questions remain, it is clear that apprecia- tion of practice and practitioners' points of view will im- prove the chances of adoption, and hence these research outcomes can only benefit from the insights ethnography can bring.
Qualitative research has long been discussed as a way to generate hypothesese and problem formulations that then can be investigated using quantitative research methods. Seaman [104] was one of the first within soft- ware engineering to promote qualitative research with this idea. However it is also appropriate even within are-as that have been well-researched because ethnographic studies (within and outside software practice) provides a different perspective that can produce new insights [121].
In Example PyPy, earlier observations of how members of a distributed industrial project coped with distributed software engineering led the ethnographic researcher to understand that distributed teams consciously adjust their practices when contingent changes in cross-site col- laboration arise. Similar practices of articulation and me- ta-work were thereafter observed in the PyPy project. Such re-negotiation of coordination and cooperation prac- tices have not been reported previously from studies in empirical software engineering. Similarly, Example Awareness looked at a technical construct (APIs) and un- covered their role in the coordination of software devel- opment work, which was a new insight. In both examples we see that the ethnographic research unearthed unex- pected, innovative research questions, which in turn led to a better understanding of the cooperative and social aspects of software engineering.
Ethnography puts cooperative, social and human as-pects of software engineering practice in the centre, and is therefore very well-suited to any research question focus- ing on these aspects. Whether or not, more broadly, an ethnographic study is appropriate as the main source of data depends largely on the research question to be an- swered [95]. For example, asking 'How do software prac- titioners develop systems using XP?' rather than 'Is single programmer coding more productive than pair pro-gramming?' or 'Why don't scientists adhere to a company manual of software development practice?' rather than 'Does structuring the manual this way help scientists produce more lines of code an hour?'

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

However all empirical phenomena take place within a context, and that context will influence the phenomenon, and hence its study. Using ethnography as a contextual research method is therefore relevant for a much wider set of research questions, especially if sustainable im- provements are sought, because they can then take ac- count of local conditions and embed local exper- tise.

There are many (empirical) software engineering ques- tions that an ethnographic study may address: What is so interesting about discussions of APIs (Example Aware- ness)? How do software engineers use graphical repre-

sentations in design [124]? How do they use artefacts in their environments in Agile Development? And can they replace these tangible artefacts when working in a dis- tributed manner without losing the benefits of physicality (Example Agile)? How do software engineers involve their users throughout the software development that is, on first sight, governed by a waterfall model [38]? Why do steering committees carefully re-plan a delayed devel- opment project using the terminology of the company- wide project model while at the same time violating the rules of this very model [99]? What is the meaning of 'applying object oriented design' [15] in a way that is not recognizable as such, when looking from the outside?

We would like to add a reservation here, though. Radi- cal changes in tools, techniques, methods and processes, cannot be *supported* by ethnographic research. Ethno- graphic methods rely on existing practices. If a tool, tech- nique or method is supposed to radically change a prac- tice, ethnographic research is of little help in effecting thatchange, although it can be used to *evaluate* the proposed practices in pilot studies.

Why a software engineer should consider beingthe ethnographer

The relationship between the ethnographer and her in- formants is key. If the ethnographer is a software engi- neer, then to some extent he is already a member of the community being studied, which brings both advantages and disadvantages. A key advantage is that the insights produced from the study are more likely to be relevant and of interest to other software engineers. Another is that access to the informants' discussions and rationalities will be more meaningful, which in turn leads to more meaningful analysis. Like other professionals, peer re-spect is a significant force in the software practitioner community, and we have found that software practition- ers react differently to a researcher who has technical credibility rather than someone from a different discipline[34], [114].

Two key disadvantages of a software engineer being the ethnographer are corollaries to the advantages above. From the ethnographer's point of view, a software engi- neer may be tempted to make assumptions or to be judgmental [115]. From the practitioners' point of view, they will expect someone knowledgeable in the area to offer guidance, especially comparative comments if the ethnographer is visiting several organisations [114]. It can be difficult for the ethnographer to keep a research stanceand a suitable scientifically accountable manner. One framework to handle this (the Co-operative Development Method) was discussed in Section 3.3.

How the dimensions and roles relate to eachother

The previous sections highlighted five dimensions of an ethnographic study (section 2.4) as well as four different roles for ethnographic studies of empirical software engi-neering (section 3). So how do these dimensions and rolesrelate to each other? First of all, it needs to be said that

there are no strict guidelines about this, i.e., this is not about building a 5x4 matrix of dimensions and roles and filling the cells with recommendations. As ethnographic studies recognize, the context where the research takes place is very important, therefore, a matrix like that would be a huge simplification. In any case, the im- portant point is that these aspects relate to and impact upon each other. Their relationships are quite complex.

For instance, if the goal of the study is to investigate the relevance of social and humans aspects of software de- velopment practice, a researcher may spend less time in the field, because even short periods might be enough time to reach a good understanding about the site and gain useful insights. Examples Agile and Testing both illustrate this. In Example Agile, time spent in the field was short but intense and related to the length of agile iterations, whereas in Example Testing, the study was dispersed over months with shorter periods of data col- lection on site.

In contrast, using ethnographic studies to inform soft- ware process and method improvement will typically require intense fieldwork in the beginning, followed by workshops and deliberations of the changes, followed again by intense fieldwork and evaluative interviews andfocus groups. In other words, this will likely be a longer ethnographic study – compared to others with different goals – requiring more engagement from the researchers and informants. Example Architecture illustrates such a study design.

Finally, it is important to have in mind that ethnograph-ic research allows the adaptation of the research design based on initial findings, therefore the kind of engage- ment and participation often will need to be adapted dur-ing the course of the study.

## 5. Conclusion

Ethnographic studies can and have already contributed tothe goals of empirical software engineering. The four main features of ethnography underpin the value of thesestudies to an engineering discipline such as empirical software engineering as follows:

International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

a focus on the members' point of view allows the ra- tionalities of practice to be made explicit, and hence exposes *why* software engineers do what they do;
a focus on the ordinary detail of life emphasises *local*context and *local* expertise which can be overlooked when using other research approaches;
the analytical stance makes the results of an ethno- graphic study more than a simple account of activity;and
the production of 'thick descriptions' supports aca-demic accountability.
Section 3 highlighted four significant roles that ethno- graphic studies of software practice can and have ful-filled. Each of these roles is substantial and contributes to the goals of empirical software engineering, but this list is not exhaustive and might be extended through future studies. Many examples of ethnographic studies that have
focused on software engineering have been referred to in this paper. Eight of these are focused on in more detail to illustrate the implications that such studies have had (see Table 1), the five dimensions of an ethnographic study (see Table 2), and the four roles that ethnographic studiesmay play in empirical software engineering (see Table 4).
By investigating and articulating both the work behav- iour and the rationalities behind that behaviour, meaning- ful and sustainable improvements can be researched, de- vised and introduced to practice. Moreover, if software engineers undertake the ethnographic study themselves, then this can increase the likelihood of the findings being of use within an empirical software engineering context.
Adopting paradigms that have a different disciplinary origin can be daunting, and adopting ethnography is no different. Each study design requires a number of deci- sions about the usage of the ethnographic method within the specific context and circumstances. In section 2.4, we discussed five dimension along which the studies re- viewed for this article differ: the degree and kind of par-ticipation, the duration of the study, the space and loca-tion of the study, the theoretical underpinning applied, and the researchers intent. Approaching the design of an ethnographic study with these five dimensions provides a practical framework for the newcomer, supporting the design according to the research question being investi- gated, the context of the fieldwork and the characteristics of the main focus of the study. As mentioned above, eth- nographic research allows adaptation as new data and findings unfolds. This means that the study does not need to be over-engineered; the researcher needs to start and be aware of and sensible to the circumstances of the study and adapt her research accordingly.
The central contribution of ethnographic studies is theirability to uncover the rationalities of the observed prac- tices. Therefore, ethnographic studies provide an im- portant complement to other empirical research methods,like experiments, that rely on prior formulation of hy- potheses. Ethnographic studies help understand *how*, andmore importantly, *why*, software teams do the things thatthey do, such as organize themselves in a specific way, coordinate their activities, apply or not apply methods, and use, or not, specific tools and techniques. These find- ings not only improve our insight into the subject of our discipline but can and should also be used to inform tool design and method development.

## ACKOWLEDGMENT
The authors wish to thank attendees at their tutorials dur- ing ICSE over the last 5 years, and all the collaborators who have supported our ethnographic studies.

## REFERENCES
Abelein, U., Sharp, H., and Paech, B. (2013) 'Does Involving Users in Software Development Really Influence System Success?', *IEEE Software*, Nov/Dec 2013 pp13-19.
Anderson, B. (1997) Work, Ethnography and System Design. The Encyclopedia of Microcomputers, Vol. 20 (A. Kent and J. G. Williams, eds.), Marcel Dekker, New York, pp. 159-183.
Anzai, Y., and Simon, H. A. (1979). The theory of learning by doing. Psychological Review, 86, 124-140
Avram, G., Bannon, L., Bowers, J., Sheehan, A. and Sullivan,
D.K. (2009) 'Bridging, Patching and Keeping the Work Flowing: Defect Resolution in Distributed Software Development', Com-puter Supported Cooperative Work (CSCW), 18(5-6), 477-507.
Ball, L.J. and Ormerod, T.C. (2000) 'Putting ethnography to work: the case for a cognitive ethnography of design', *Interna- tional Journal of Human-Computer Studies*, 53(1), 147-168.
Belbin, R.M. (2010) *Team Roles at Work*, 2nd ed., ButterworthHeinemann.
Bendifallah, S. and Scacchi, W. (1987) "Work structures and shifts: An empirical analysis of software specification team- work." In *Proceedings of the 11th international conference on Soft-ware engineering*, pp. 260-270. ACM, 1989
Blomberg, J. and Karasti, H. (2013) "Reflections on 25 Years of Ethnography in CSCW, Computer Supported Cooperative Work, 22:373–423
Blomberg, J., and Karasti H. (2012) *Positioning ethnography with- in Participatory Design*. In: Simonsen, J. and Robinson T. Routledge International Handbook of Participatory Design. Routledge, London.

Blumer, M. (1984). *The Chicago School of Sociology: Institutionali- zation, Diversity, and the Rise of Sociological Research*. Chicago: University of Chicago Press.
Boden, A., Avram, G., Bannon, L. and Wulf, V. (2012): "Knowledge sharing practices and the impact of cultural fac- tors: Lessons from two case studies of offshoring in SME". In Software Maintenance and Evolution: Research, and Practice, vol. 24, no. 2, pp. 139-152.
Boden, A., F. Rosswog, G. Stevens, and V. Wulf (2014): 'Articu- lation Spaces: Bridging the Gap Between Formal and Informal Coordination'. In: *Proc. of the 17th Conf. on Computer Supported Cooperative Work & Social Computing*. pp. 1120–1130.
Bondarenko and Janssen, (2005) 'Documents at Hand: Learning from Paper to Improve Digital Technologies', In Proceedings of CHI'05, pp 121-130
Button, G. (2000) The Ethnographic Tradition and Design.
*Design Studies*, 21, 319-332.
Button, G. and Sharrock, W. (1994) 'Occasioned practices in the work of software engineers', in Requirements Engineering: social and technical issues edited by M. Jirotka and J. Goguen, pp 217-240, Academic Press
Capiluppi, A., Ramil, J.F., Higman, J., Sharp, H. and Smith, N. (2007) 'An empirical study of evolution patterns for an agile system, that combines qualitative and quantitative approaches' in *Proceedings of ICSE 2007*, ACM, pp 511-518
Carroll, J. and Rosson, M.B. (2001) 'Better Home Shopping or New Democracy? Evaluating Community Network Outcomes', In Proceedings of CHI '01, pp 372-379.
Carstensen, P.H. and Sørensen, C. (1996) From the social to the systematic. *Comput. Supported Coop. Work* 5, 4 (December 1996), 387-413.
Carstensen, P.H., Sorensen, C. and Tuikka, T. (1995) "Let's talk about bugs." *Scandinavian Journal of Information Systems* 7: 33-33.
Cataldo, M., Wagstrom, P.A., Herbsleb, J.B. and Carley, K.M. (2006) Identification of coordination requirements: implications for the Design of collaboration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (CSCW '06). ACM, New York, NY, USA, 353-362.
Cohn, M. L., Sim, S. E. and Lee, C. P. (2009). What counts as software process? Negotiating the boundary of software work through artifacts and conversation. *Computer Supported Cooperative Work (CSCW)*, *18*(5-6), 401-443.
Crabtree, A. (2003) *Designing Collaborative Systems*, Springer
Crabtree, A., Rodden, T., Tolmie, P., & Button, G. (2009). Eth- nography considered harmful. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 879-888). ACM.
Crabtree, A., Rouncefield, M., Tolmie, P. 2012. Doing Design Ethnography. Springer.

Damian, D., Izquierdo, L., Singer, J. and Kwan, I. (2007) Awareness in the wild: Why communication breakdowns occur. In *Second IEEE International Conference on Global Software Engineering*, 2007. ICGSE 2007. pp. 81–90. IEEE.
de Souza, C. R. B. (2005) On the Relationship between Software Dependencies and Coordination: Field Studies and Tool Support . Ph.D. dissertation, Donald Bren School of Information and Computer Sciences, University of California, Irvine, Irvine, CA, USA.
De Souza, C., Froehlich, J., & Dourish, P. (2005) 'Seeking the source: software source code as a social and technical artifact' In Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, ACM, pp. 197-206.
de Souza, C. R. B.; Redmiles, D. F. (2008) 'An Empirical Study of Software Developers Management of Dependencies and Changes', in *Proceedings of International Conference on Software Engineering*, Leipzig, pp 241-250.
de Souza, C. R. B. and Redmiles, D. F. (2009) 'On The Roles of APIs in the Coordination of Collaborative Software Development', *Computer Supported Cooperative Work*, **18**, 445- 475.
de Souza, C. R. B., Redmiles, D., Cheng, L.-T., Millen, D. and Patterson, J. (2004) 'How a Good Software Practice thwarts Collaboration - The Multiple roles of APIs in Software Development', in *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering,* Newport Beach, CA. pp 221-230.
de Souza, C. R. B., Redmiles, D. F., Mark, G., Penix, J., Sierhuis,
M. (2003) 'Management of Interdependencies in Collaborative Software Development', in Proceedings of *ACM-IEEE International Symposium on Empirical Software Engineering*, Rome, IEEE Computer Society, pp 196-203.
de Souza, C.R.B., Sharp, H., Singer, J., Cheng, L-T., Venolia, G. (2009) 'Guest Editors' Introduction: Cooperative and Human Aspects of Software Engineering', *IEEE Software*, November 2009
Detienne, F. and Bott, F. (2002) *Software design-cognitive aspects*,
Springer Verlag
Dittrich, Y. (2002) 'Doing empirical research on software development: finding a path between understanding, intervention, and method development', in: Dittrich, Y., Floyd,
C. and Klischewski, R. (eds.) *Social Thinking – Software Practice*, MIT Press, pp 243-262.
Dittrich, Y. (2016) 'What does it mean to use a method? Towards a practice theory for software engineering', *Information and*

*Software Technology* **70**, 220–231..
Dittrich, Y. and Giuffrida, R. (2011) 'Exploring the role of in- stant messaging in a global software development project', in *Proceedings of Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on* (pp. 103-112). IEEE Computer Socie- ty, Washington, DC, USA, pp 103-112.

Dittrich, Y., John, M., Singer, J. and Tessem, B. (2007) 'Editorial for the special issue on qualitative software engineering research', *Information and Software Technology* **49**(6), 531–539.
Dittrich, Y. and Lindeberg, O. (2004) 'How use-orientated development can take place', *Information and Software Technology* **46**, 603-617.
Dittrich, Y., Randall, D. W. and Singer, J. (2009) 'Software engineering as cooperative work.' *Computer Supported Cooperative Work (CSCW)*, *18*(5-6), 393-399
Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C. and Lindeberg, O. (2008) 'Cooperative method development' *Empirical Software Engineering*, *13*(3), 231-260.
Dourish, P. and Button, G. (1998) "On technomethodology: Foundational relationships between ethnomethodology and system design." *Human-Computer Interaction* 13.4 (1998): 395-432.
Dourish, P. (2004) *Where the Action Is: The Foundations of Embod-ied Interaction*. Cambridge: MIT Press.
DSDM (2015) *DSDM and UX Design*, DSDM Consortium, ISBN 978-1-910961-00-1
Dybå, T., Kitchenham, B.A., and Jørgensen, M. (2005) Evidence- based Software Engineering for Practitioners, *IEEE Software*, 22(1): 58-65.

Easterbrook, S., Singer, J., Storey, M-A and Damian, D. (2008) 'Selecting Empirical Methods for Software Engineering Research' in *Guide to Advanced Empirical Software Engineering*, F.Shull, J. Singer and D. Sjøberg (eds), pp 285-311, Springer.
Engeström, Y. (1999). Expansive visibilization of work: An activity-theoretical perspective. *Computer Supported Cooperative Work (CSCW)*, *8*(1), 63-93. Engeström, Y., Miettinen, R, Punamäki, R.-L. (1999) *Perspectives on Activity Theory*. Cambridge University Press.
Ferreira, J., Sharp, H. and Robinson, H. (2012) 'Agile Develop- ment and User Experience Design Integration as an On-going Achievement in Practice', in Proceedings of Agile 2012
Fetterman, D. M., (1998) *Ethnography*. Second Ed. Sage Publica-tions, Thousand Oaks, CA.
Floyd, C. (1987) Outline of a Paradigm Change in Software Engineering. In: Bjerknes, G., Ehn, P., Kyng, M. (Eds.) *Comput- ers and Democracy*. Aldershot 1987, pp. 192-210.
Floyd, C. (1992) Software Development as Reality Construction. In: Floyd, C., Züllighoven, H., Budde, R., Keil-Slawik, R. (Eds.) *Software Development and Reality Construction.* Springer Verlag,
Berlin, pp. 86-100
Garfinkel (1967) *Studies in Ethnomethodology*, Wiley
Geertz, C. (1988) *Works and Lives: The Anthropologist as Author*. Stanford University Press.

Geiger, R.S. and Ribes, D. (2011) Trace ethnography: Following coordination through documentary practices. In 44th Hawaii International Conference on System Sciences (HICSS), pp 1–10.IEEE.
Giuffrida, R., Dittrich, Y. (2011) Exploring the role of instant messaging in a global software development project. In: *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, IEEE, pp. 103-112
Goh, D.P.S. (2007) States of Ethnography: Colonialism, Resistance, and Cultural Transcription in Malaya and the Philippines, 1890s–1930s. Comparative Studies in Society and History, 49, pp 109-142. doi:10.1017/S0010417507000424.
Green, T.R.G. (1989) Cognitive dimensions of notations. In: Sutcliffe, A., Macaulay, L. (Eds.), People and Computers V. Cambridge University Press, pp. 443–460.
Grinter, R. E. (2003). Recomposition: Coordinating a web of software dependencies. *Computer Supported Cooperative Work (CSCW)*, *12*(3), 297-327
Hammersley, M. and Atkinson, P. 1995 What is Ethnography. In: Ethnography: Principles in Practice. Routledge, London 9- 25.
Heath, C., Luff, P. (1992) Collaboration and control: crisis man- agement and multimedia technology in London underground line control rooms. *Computer Supported Cooperative Work* (CSCW) 1 (March (1–2)), 69–94.

Hine, C. (2000) *Virtual ethnography*. Sage Publications Ltd.
Holtzblatt, K. and Jones, S. (1993) Contextual Inquiry: a participatory approach for systems design', in D. Schuler and
A. Namioka (eds) Particiopatory Design: Principles and practice. Lawrence Erlbaum and Associates, Hillsdale, NJ, pp177-210.
Hovenden, F.M., Walker, S., Sharp, H.C. and Woodman, M. (1996) 'Building quality into scientific software', *The Software Quality Journal,* **5**, 25–32, Chapman & Hall, ISSN 0963-9314

Hughes, J., King, V., Rodden, T., & Andersen, H. (1994). Moving out from the control room: Ethnography in system design. In *Proceedings of the 1994 ACM conference on Computersupported cooperative work* (pp. 429-439). ACM.
Humboldt, W. v. (1836) *The Heterogeneity of Language and its Influence on the Intellectual Development of Mankind*. 1836. (orig.*Über die Verschiedenheit des menschlichen Sprachbaus und ihren Einfluss auf die geistige Entwicklung des Menschengeschlechts*). 1836. New edition: On Language. On the Diversity of Human Language Construction and Its Influence on the Mental Development of the Human Species, Cambridge University Press, 2nd rev. edition 1999.
Hutchins, E. (1995) *Cognition in the Wild.* MIT Press, Cambridge,MA.
Hutchinson, H., Mackay, W., Westerlund, Bederson, B.B., Druin, A., Plaisant, C., Beaudouin-Lafon, M., Conversy, S., Evans, H., Hansen, H., Roussel, N., Eiderbäck, B., Lindquist, S., and Sundblad, Y. (2003) 'Technology Probes: Inspiring Design for and with Families' In Proceedings of CHI '03 pp 17-24.
Jordan, B. (2009) Blurring boundaries: The "real" and the "virtual" in hybrid spaces. *Human Organization*, 68(2):181–193.
Jorgensen, D.L. (1989) *Participant Observation: A Methodology for Human Studies*, Sage Publications Inc.
Kitchenham, B., et al., Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 2002. 28(8): p. 721-734.
Klischewski, R. (2002) 'Reaching out for Commitments: Systems Development as Networking' Dittrich, Y., Floyd, C. and Klischewski, R. (eds.) *Social Thinking – Software Practice*, MIT Press, pp 309-329.
Latour, B. (1987).*Science in Action: How to Follow Scientists and Engineers Through Society.* Milton Keynes: Open University Press.
Latour, B. (1994). Where are the missing masses? The sociology of a few mundane artifacts. Shaping Technology / Building Society: Studies in Sociotechnical Change. W. Bijker and J. Law. Cambridge, MA, MIT Press**:** 225-258.
Law, J., & Hassard, J. (1999) 'Actor network theory and after'
*Sociological review*.
Low, J., Johnson, J., Hall, P., Hovenden, F., Rachel, J., Robinson,
H. and Woolgar, S. (1996) 'Read this and change the way you feel about software engineering', *Information and Software Technology* **38** 77-87
Malinowski, B. 1922. Argonauts of the Western Pacific. London: Routledge (1967). Marcus, G.E. (1995) Ethnography in/of the world system: theemergence of multi- sited ethnography. *Annual review of anthropology*, pages 95–117.
Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2007) 'Good'Organisational Reasons for'Bad'Software Testing: An Ethnographic Study of Testing in a Small Software Company. In *Software Engineering, 2007. ICSE 2007. 29th International Conference on* (pp. 602-611). IEEE.
Mathiassen, L., (1998) Reflective Systems Development.
*Scandinavian Journal of Information Systems,* 10(1&2), 67-118.
McGrath, J.E. (1994) Methodology Matters: Doing Research in the Behavioral and Social Sciences, in *Human-computer interac- tion*, pp 152-169, Morgan Kaufmann, San Francisco.
Mi, P. and Scacchi, W. (1991) 'Modeling articulation work in software engineering processes', in Proceedings of First Interna- tional Conference on the Software Process, pp 188-201, IEEE.
Millen, D. (2000) 'Rapid Ethnography: Time Deepening Strate- gies for HCI Field Research' in Proceedings of DIS '00, pp 280- 286.
Mursu, A. S., Luukkonen, I., Toivanen, M., & Korpela, M. J. (2006). 'Activity Theory in information systems research and practice: theoretical underpinnings for an information systems development model', *Information Research* **12**(3), 3.

Nardi, B. (1997) The use of ethnographic methods in design and evaluation. In Handbook of Human- Computer Interaction, M. Helander, T. K. Landauer, e P. Prabhu, Eds., pp. 361-366.
Elsevier Science
Naur, P. (1985) 'Programming as Theory Building.'
*Microprocessing and Microprogramming* 15(1985), 253-261.
Naur, P. and Randell, B. (1969) *Software Engineering.* Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO.
Nicolini, D. (2013) *Practice theory, work, and organization: An introduction*. Oxford University Press.
Nygaard, K. (1986). Program Development as a Social Activity. In: H.J. Kugler (eds.) *Information Processing 86*. IFIP, 189-198.
O'Riain, S. (2000) Net-working for a living: Irish software developers in the global workplace. The Blackwell cultural economy reader, pages 15–39.
Passos, C., Cruzes, D.S., Dybå, T. and Mendonça, M. (2012) 'Challenges of applying ethnography to study software practices'in *Proceedings of ESEM '12, the ACM-IEEE international symposium on Empirical software engineering and measurement*,
Pages 9-18, ACM
Poderi, G. (2012). Simple conversational practices in the case of free and open source software infrastructure. In *Proceedings of the*

*12th Participatory Design Conference: Exploratory Papers, Workshop Descriptions, Industry Cases-Volume 2* (pp. 45-48). ACM.
Poderi, G. (2013) *Making Sense of Users Participation in Open Source Projects: The case of a Mature Video Game*. Diss. University of Trento.
Preece, J., Rogers, Y. and Sharp, H. (2015) *Interaction Design*,Fourth edition, John Wiley.
Prior, J.R., Robertson, T.J. & Leaney, J.R. 2008, 'Situated Software Development: Work Practice and Infrastructure are Mutually Constitutive', Australian Software Engineering Conference, Perth, Western Australia, March 2008 in *Proceedings of 19th Australian Software Engineering Conference*, edvon Konsky, B.; Grundy, J., IEEE Computer Society, United States, pp. 160-169. Robinson, H.M., J. Segal, and H. Sharp (2007) Ethnographically- informed Empirical Studies of Software Practice. *Information and Software Technology,* 2007. **49**(6): pp. 540-551.
Robson, C. (2011) *Real World Research* (3rd ed), John Wiley andSons
Rönkkö, K. (2010) 'Ethnography', in Laplante, P. (ed.) Encyclopedia of Software Engineering, Taylor and FrancisGroup, New York, 2010
Rönkkö K., Dittrich Y. and Lindeberg O. (2002) "Bad practice" or "bad methods"—are software engineering and ethno- graphic discourses incompatible? Proceedings 1st InternationalSymposium on Empirical Software Engineering (ISESE'02),
Nara, Japan, pp 204–210
Rönkkö, K., Dittrich, Y., & Randall, D. (2005). When plans donot work out: How plans are used in software development projects. *Computer Supported Cooperative Work (CSCW)*, *14*(5),433-468.
Rönkkö, K., Hellman, M., Kihlander, B. and Dittrich, Y., (2004) Personas is not Applicable: Local Remedies Interpreted in a Wider Context, Proceedings of the Participatory Design Con- ference, PDC '04, Artful Integration: Interweaving Media, Ma- terials and Practice, Toronto, Canada, July 27-31, pp. 112-120.
Rooksby, J., Rouncefield, M. and Sommerville, I. (2009) 'Testingin the wild: the social and organizational dimensions of real world practice', CSCW, 18:559-580
Runeson, P. Host, M., Rainer, A. and Regnell, B. (2012) *Case Study Research in Software Engineering: Guidelines and Examples*,Wiley.
Sach, R.J., Sharp, H., Petre, M. (2011) 'Software Engineers' Perceptions of Factors in Motivation' in *Proceedings of ESEM2011*
Seaman, C. (1999) Qualitative Methods in Empirical Studies of Software Engineering, In *IEEE Transactions on Software Engineering*, 25(4), 557-572.
Segal, J. (2009) Software development cultures and cooperation problems: A field study of the early stages of development of software for a scientific community. *Computer Supported Cooper-ative Work (CSCW)*, *18*(5-6), 581-606.
Segal, J. and Clarke, S. (2009) 'Software engineers don't know everything about end-user programming', *IEEE Software*, Sep- tember/October
Shapiro, D. 1994. The Limits of Ethnography: Combining SocialSciences for CSCW. Proceedings of the CSCW 1994, ACM Press.
Sharp, H. (2007) 'The role of physical artefacts in agile software development team collaboration*', Proceedings of Physicality 2007*,pp 61-64, UWIC Press, ISBN 978-1-905617-60-9
Sharp, H., Giuffrida, R. and Melnik, G. (2012) 'Information flow in a dispersed agile team: a distributed cognition perspective', in Proceedings of XP 2012, Malmo, Sweden.
Sharp, H. and Robinson, H.M. (2004) 'An ethnographic study ofXP practices', *Empirical Software Engineering,* **9**(4) 353-375
Sharp, H. and Robinson, H.M. (2008) 'Collaboration and Co-ordination in mature eXtreme Programming teams'
*International Journal of Human-Computer Studies,* **66,** 506-518
Sharp, H., Robinson, H.M. and Petre, M. (2009) 'The Role of Physical Artefacts in Agile Software Development: two complementary perspectives', *Interacting with Computers*, **21**(1-2) 108-116
Sharp, H., Robinson, H. and Woodman, M. (2000) 'Software Engineering: Community and Culture', *IEEE Software*, **17**(1), 40□47, ISSN 0740-7459
Sharp, H., Robinson, H. and Woodman, M. (2000a) 'Using Ethnography and Discourse Analysis to Address Software Quality Issues' *Proceedings of ICSE 2000 workshop Beg, Borrow or*
*Steal: Using Multi-disciplinary Approaches in Empirical SoftwareEngineering Research*
Sharp, H., Woodman, M. and Hovenden, F. (2004) 'Tensions in the adoption and evolution of software quality management systems: a discourse analytic approach' *International Journal of Human Computer Studies,* **61**(2), 219-236
Shull, F., J. Singer, D. Sjoberg, Shull, (Eds.) (2008) Advanced topics in empirical software engineering: An edited volume.Springer.
F. Shull and J. Singer (eds) *Guide to Advanced Empirical Software Engineering*, Springer.
Sigfridsson, A. (2010) *The purposeful adaptation of practice: an empirical study of distributed software development*. PhD thesis at the University of Limmerick, Ireland.
Sigfridsson, A., and Sheehan, A. (2011). On qualitative methodologies and dispersed communities: Reflections on the process of investigating an open source community. *Information and Software Technology*, *53*(9), 981-993.

Sigfridsson, A., Avram, G., Sheehan, A., & Sullivan, D. (2007). Sprint-driven development: working, learning and the process of enculturation in the PyPy community. *Open Source Development, Adoption and Innovation*, 133-146.
Sim SE, Singer J, Storey M-A (2001) Beg, borrow, or steal: using multidisciplinary approaches in empirical software engineering research, an ICSE 2000 workshop report. *Empirical Software Engineering* 6(1):85–93
Singer, J., Lethbridge, T., Vinson, N. and Anquetil, N. (1997). An examination of software engineering work practices. *Proc. CASCON*. IBM Toronto.
Sjøberg, D.I.K., Dybå, T. and Jørgensen, M. (2007) The Future of Empirical Methods in Software Engineering Research Future of Software Engineering (FOSE'07) 0-7695-2829-5/07
Suchman, L. (1987) *Plans and Situated Actions*, Cambridge University Press.
Suchman, L. and Trigg, R.H. (1993) Artificial Intelligence as craftwork, in S. Chaiklin & J. Lave (eds). *Understanding practice: Perspectives on activity and context*, Cambridge UK.
Teasley, S., Covi, L., Krishnan, M.S. and Olson, J.S. (2000) How does radical collocation help a team succeed?, Proceeding of the ACM 2000 Conference on Computer supported cooperative work, pp.339-346, Philadelphia, Pennsylvania, United States.
Tell, P., & Babar, M. A. (2012). 'Activity theory applied to global software engineering: Theoretical foundations and implications for tool builders' Proceedings of the Seventh IEEE International Conference on Global Software Engineering (ICGSE), IEEE, pp. 21-30.
Trainer, E., Quirk, S., de Souza, C. and David Redmiles (2005) Bridging the gap between technical and social dependencies with Ariadne. In Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange (eclipse '05). ACM, New York, NY, USA, 26-30
Trimble, J., Wales, R. and Gossweiler, R. (2002) NASA position paper for the CSCW 2002 workshop on Public, Community and Situated Displays: MERboard
Unphon, H. (2009) *Re-engineering for Evolvability* PhD Thesis. IT University of Copenhagen.
Unphon, H. and Dittrich, Y. (2010) Architecture Awareness in Long-Term Software Product Evolution. *Journal for Systems and Software 83*, 2211-2226.
Viller, S. and Sommerville, I. (1999) Coherence: an approach to representing ethnographic analyses in systems design. *Human—Computer Interaction 14,* Special issue on representations in interactive systems development.
Walsham, G. (2002). Cross-cultural software production and use: a structurational analysis. *MIS Quarterly*, 26(4), 359–380.
Weber, J. and Cheng, J. (2013) 'Making the most of ethnographic research' in UX magazine (online), downloaded from http://uxmag.com/articles/making-the-most-of-ethnographic-research, accessed on 13.06.2015
Weinberg, G. (1971) *The Psychology of Programming*. Dorset House.
Yoon, Y. and Myers, B. (2014) "A Longitudinal Study of Pro- grammers' Backtracking" in Proceedings of VL/HCC 2014