The Role of Ethnographic Studies in Empirical Software Engineering

Helen Sharp, *Member, IEEE*, Yvonne Dittrich, and Cleidson R. B. 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; to improve 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 studies of software practice are used to illustrate the points being made.

Index Terms—Design tools and techniques, human factors in software design, software engineering process, computer-supported collaborative work

1 Introduction

Ethnography is a research method recognized as a significant qualitative empirical approach suited to understanding 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 Computer-Supported Co-operative Work (CSCW) and HCI communities to conduct studies of the workplace and inform the design of computer applications e.g., [8]. Ethnographic studies are also used every 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 ethnography and hence there is little support in the literature for applying ethnography to software practice. This in turn leads to a lack of awareness in the community and hence unfamiliarity with the approach. This paper aims to address this situation.

Several articles and journal collections focusing on the use of qualitative methods in software engineering research e.g., [37], [116], [120], and several sets of guidelines for empirical work in software engineering have been published, e.g., [122] and [45], but ethnography receives 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 engineering. Kitchenham et al.'s [74] guidelines for empirical research 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 culture (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 practice 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 engineering researchers. In addition to the lack of awareness and familiarity, applying ethnographic methods as part of software engineering research is challenging, partly because ethnology and sociology – the homesteads of ethnography – are descriptive and analytical disciplines, whereas software engineering, as an engineering discipline, is interested in improving the way software is developed and built. Practitioners and fellow researchers expect empirical software engineers to discuss possibilities for improvement or new methods and technologies [30], [37], [115] and are puzzled by ethnography that appears to focus only on understanding and describing a situation. Dittrich [34] reports how the software engineering researchers' role was co-constructed by practitioners and management to be associated with

- *H. Sharp is with the Open University, Walton Hall, Milton Keynes MK7 6AA, UK. E-mail: helen.sharp@open.ac.uk.*
- *Y. Dittrich is with the Software and Systems Section, IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark. E-mail: ydi@itu.dk.*
- *C.R.B. de Souza is with the Vale Institute of Technology and the Federal University of Pará, Tv. Boaventura da Silva, 955, Belém 66055-090, PA, Brazil. E-mail: cleidson.desouza@acm.org.*

Manuscript received 17 July 2015; revised 21 Dec. 2015; accepted 11 Jan. 2016. Date of publication 19 Jan. 2016; date of current version 19 Aug. 2016. Recommended for acceptance by D. Damian.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2016.2519887

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 misunderstood or misapplied, leading to results being dismissed with a "so what?" response. In our experience, other forms of field study such as observational study, case study and fieldwork are more commonly used in empirical software engineering. An ethnographic stance would add a different perspective to these approaches. Indeed, case study research is observational rather than transformational, but Runeson et al. [102] does not regard ethnography 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 studies with large amounts of participant-observer data. We will address these aspects throughout the paper.

Empirical software engineering aims to improve software 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 confident in the feasibility of proposals. Two of the fundamental 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 software engineering, because capturing both the 'what' and the 'why' of practice provides a solid foundation for identifying 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 conducted 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 engineering research. Furthermore, and to facilitate a wider uptake of ethnography, we present a set of five dimensions 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 software engineering: addressing the significance of ethnography's analytical stance; how to know when ethnography is an appropriate research method; why an empirical software engineer should consider being the ethnographer; 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 software engineering community, were not necessarily conducted by software engineers, and in some cases researchers 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.

2 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 features of ethnography are presented, and five dimensions across which ethnographic studies vary are introduced. These dimensions have been distilled from existing ethnographic studies and related literature, and tailored for presentation to a software engineering audience in order to 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 anthropology, 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 became valued. At the same time, understanding of humanity and its variety of cultures and languages was growing [65].

Ethnography, as we know it today, goes back to Malinowski's research of an island outside Australia during the first World War [76]. The central tenet of this approach is 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 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 understanding 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

TABLE 1
Overview of Example Ethnographic Studies in Software Engineering

Example's short name	Main Publication(s)	Focus question(s)	Context	Main finding(s)	Implication for empirical software engineering
Agile	[110], [111], [112]	How agile methods (specifically XP) are put into practice	Various organisations including small start-ups and large investment banks	Explication of agile methods in practice Mechanisms of co-ordination and collaboration The role of physical artefacts	Practice varies from written descriptions Specific new research questions
Architecture	[129], [130]	How can the software architecture and the architecture practices in the company be improved	The redevelopment of a software product line for hydraulic simulation software of one-dimensional water systems like rivers, water supply and sewers. The development team and their cooperation with engineers developing specific modelling systems	An understanding of the development practices and the corresponding architectural practices. Adaptation of architectural practices to the agile development context	A set of methods for light weight architecture. When proposing and introducing methods and tools supporting architectural practices, the change of the social practices that are meant to be supported by the tools need to be considered explicitly
Awareness	[26], [28], [29]	How do software engineers manage technical dependencies as part of their day-to-day development	Two different organizations and teams. One in NASA/Ames Research Center, the other is one of the largest software development companies in the United States with products ranging from operating systems to software development tools	APIs play multiple roles in collaborative software development: contracts between developers, reification of organizations boundaries and communication mechanisms	Management of dependencies is an activity undertaken by software developers on a daily basis
Bug report	[18], [19]	How do various paper-, board- and computer-based mechanisms supported the coordination of debugging.	The beginning of the 1990s in the IT unit of Foss Electric a company providing process technology before web-based issue handlers like Bugzilla were widely used.	Developers use a coordination mechanism consisting of a physical form (bug report) and social protocol	The research informed the concept of 'coordination mechanism' as consisting of social protocols supported by an artefact. This concept has widely informed the CSCW discourse, and is increasingly cited in research on coordination in (global) software engineering.
Coordination	[58]	What are the practices necessary to coordinate the integration and built procedure of huge software products	Three organisations developing software products of different sizes. The software engineers, the project managers, architects and built managers of the projects were studied.	To prevent development problems, software architects, project managers and also software engineers apply preventive strategies, like designing project organization and meeting structures, involving relevant actors bottom up in the architectural design, or communicating changes that effect others widely, that both take organizational contingencies into account and shape the organization of the project	Re-composition of software from components and modules is not just a technical problem to be solved by integration, built tools and procedures but needs to be addressed in project management, meeting structures and last but not least the architecture
PyPy	[117], [118], [119]	How to adapt the community's processes, practices and constitutions to welcome new developers to the OSS project	The way the PyPy open source community discusses and implements changes to its practices in order to address problems with the current organization identified by the community	The strategies developed by especially distributed software projects to cope with changes in the environment	Software projects – open source as well as corporate ones – consciously adapt their practice to accommodate changing circumstances. The need for reflection and improvement of the usage of methods and tools becomes visible as an integral part of software engineering
Scientist	[105], [106]	How to integrate software engineering practices into scientific software development	Joint development of a Laboratory Information Management System (LIMS) between several UK labs working with related kinds of experiments. In this project, professional software developers were cooperating with scientific end-user developers, with scientists as the customers/users also by-and-large having experience of scientific end-user development	Change of problem focus away from distribution and onto culture clashes between scientists and software engineers	Collaborations between professional and end-user developers can be problematic as these two groups might have differing values and practices. Software engineering needs to adapt its methods, tools and processes to relate to different software development cultures
Testing	[78], [105]	What testing actually involves	The fieldwork covers several organisations. Researchers focused on day-to-day 'run of the mill' testing. Different forms of testing were observed in each project, including user acceptance testing, unit and regression testing	Testing is a cooperative activity that cannot be improved by technical means alone	Testing is a co-operative activity and improvements need to take social and cooperative aspects into account besides devising technical innovations

directly, and captures the ways in which the foreign culture differs from her home culture. These differences 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, ethnography became an approach to social research in general, not only of foreign cultures. Hammersley and Atkinson [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 research" (p. 9). Especially the Chicago School of Sociology [10] started to use ethnography to research and understand urban sub-cultures and their rationalities. Furthermore, 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 computer applications e.g., [2], [24], [107]. This was led by Lucy Suchman's [123] seminal ethnographic work on Xerox'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 computational systems". Note that Suchman was not the first one to adopt approaches from social sciences, but reinforced the interest in them [42].

2.2 Ethnographic Studies Outside Sociology

Ethnography originated in sociology but it has been successfully applied and adopted in other disciplines too. In Information Systems and Organizational Studies, ethnographic studies are rather unproblematic to the communities, 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 commercial 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 'rapid' 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 people collaborating have become a major source for understanding 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 collaborative practices observed. For instance, the study of the London underground control room [60] showed how operators made their action visible to each other, thus promoting 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 Design, ethnography has been widely accepted as a resource in the design process [9].

Altogether the experiences from neighbouring disciplines show that ethnographic methods can be successfully 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 development practices and thus provide a base to develop and devise adequate processes, tools and techniques.

2.3 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 descriptions' for academic accountability. These features distinguish ethnographic studies from other observational research. In the following sections, each one is described generally first, and then it is interpreted in the context of empirical software engineering research. Along the way, some important misunderstandings about ethnographic studies are clarified.

2.3.1 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 *rather than what he thinks is important*. In the anthropological tradition, this can only be achieved by participant observation: 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 also means that ethnographies traditionally take a long time to be conducted – from months to years. The reason for this is partially explained in

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, but it means that using interviews on their own yields only a partial view of the picture, no matter how skilled is the interviewer [96].

the previous section where we describe the context in which ethnography arose: the study of exotic cultures [42]. In this context, ethnographers need to learn a new language, a new way of living, as well as learn to adapt to new food and health conditions which often resulted in ethnographers getting sick in their new "lifestyles". Because of all these aspects, ethnographies took longer [84]. Nowadays, ethnographers are being employed to study informants that are not exotic, who are from the same culture, and live in similar conditions. 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 already have some understanding about the setting being studied, e.g., since they understand software development, then ethnographies might be conducted in shorter periods of time. For instance, in Example Agile the ethnographic study of agile practices was conducted over a period of three weeks, with one week of full-time observation 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 both an 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 knowledge to 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 talk about 'bracketing' their assumptions (i.e., putting assumptions 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.

2.3.2 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 details of what members of a culture do in their daily actions, 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 important 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, including concerns, frustrations, expectations, preferences and the like, and therefore will be able to identify aspects that might be more interesting, relevant, and worth exploring for the informant than the original research questions. 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 sticking 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, Dittrich 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 focus evolved. They started looking *not only* at these tools, but more importantly at how they were used in the context 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.

2.3.3 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 Malinowski 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 hundreds of miles across dangerous seas simply to exchange in very strictly ritualised ways apparently worthless 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 associated activities may be carried out in association with Kula; what magic governs the journeys; and so forth, Malinowski demonstrates how Kula promotes social integration. In carrying out the Kula, Trobrianders, willy-nilly, are solving one of the central problems of all societies. 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. In fact, this feature has also been neglected by researchers in human-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 ethnography will not only present the evidence acquired in the field, but will provide an analysis of the results explaining how this evidence is, or is not, relevant for a particular purpose.

Example Awareness demonstrates this analytical aspect of ethnography. De Souza et al. [30] discuss the roles of application programming interfaces (APIs) in the coordination of software development work in a large-scale organization. This account reports what was seen during the study, but most importantly, it is reported in an analytical 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 developers' 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 built. An ethnographic study in empirical software engineering will aim not simply to observe activity but to determine rationalities and explanations: what causes frustration, stops progress, or makes the team work well; why certain notations are used for certain tasks; why are different 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 collaboration; and so on. An ethnographic analysis in empirical software engineering needs to extend findings into insights that can inform the improvement of software development practice through tools or processes, although not all ethnographic work looking at software practice has achieved this.

2.3.4 Thick Descriptions for Academic Accountability

The result of an ethnography is often more comprehensive and detailed when compared to results using other research methods. This is because it aims to communicate the broad picture. This comprehensive and detailed set of data 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 ethnography are realistic[2], have high internal validity, but, on the other hand, have weak external validity, i.e., do not necessarily 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 dissection of even one swallow may provide a great deal of reliable information about swallow anatomy"

2. Realistic in as far as the situation or context within which the evidence is gathered is comparable to the contexts in which you want your evidence to apply [80].

2.4 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 five dimensions along which ethnographic studies differ. These dimensions have been distilled from existing ethnographic studies and related literature, and tailored specifically for a software engineering audience. The balance of dimensions chosen for any one study will be influenced 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 any empirical software engineering researcher interested in conducting an ethnography study. Table 2 maps these dimensions to the eight example studies used throughout the paper.

2.4.1 Participant or Non-Participant Observation

Observation is key to ethnographic studies and both participant 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 effectively performing the same actions as the informants [69]. In contrast, non-participant observation involves the ethnographer observing the actions, but not necessarily doing them as well [69]. Participant observation can be impractical in a work-based situation such as software engineering 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 meaningful role within the community and engaging with the community's everyday business. For instance, when observing 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 documentation tasks [129] or support the project through administration [75].

2.4.2 The Duration of the Field Study

Traditionally, an ethnographic study in ethnology would require a long-term participation in the field and extended 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 (four days a week) of a water utility company and Prior et al.'s [94] 20 month-long study (45 days in total) of a group of professional software developers. An example of short-term ethnography in software engineering 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

TABLE 2
Dimensions of Ethnography Mapped to Example Studies

Short name	Degree of participation	Duration	Space and location	Theoretical underpinning	Ethnographer's intent
Agile	mixed degrees of participation across studies	at least one iteration per study (one to three weeks)	co-located (mostly), and dispersed	distributed cognition, cognitive dimensions and technological frames	to produce a detailed account of a new area, and identify practitioner-relevant research questions
Architecture	participatory observation and participation, workshops, light-weight intervention.	the whole action research took place over two years, with periods of intense fieldwork (4 days a week) but also rather sparse visits (once every second week)	co-located	none	to improve practice, develop and evaluate method development
Awareness	participation at the first and non-participation at the second	8 weeks and 11 weeks (two studies)	co-located (mostly), and dispersed	none	to understand and provide improved tool support
Bug report	participation	fieldwork consisting of 21 interviews, participation in 10 project meetings and about 75 hours of observation over 6 months	co-located	ethnomethodology, practice theory	understanding
Coordination	participation and non-participation	the study analyses field work from 3 different sites: Field site 1: 3 months part observation over 100 interviews; Field site 2: 3 days participatory observation 14 interviews; Field site 3: consulting 3 projects about integration procedures.	not stated	none	understanding
PyPy	mixed degrees of participation across study timeline	7 days of observation (one sprint) by several researchers; prepared beforehand by reading documentation about the community; 4 months of participant observation of the community.	distributed	Social Practice Theory, social theory of learning, ethnomethodology	understanding the reflective aspects of the observed practices that led to change decided and implemented by the practitioners
Scientist	no participation	4 + years, although the observation was limited to project meetings.	distributed	none	to help practitioners understand and solve a problem
Testing	no participation	30 days fieldwork over 10 months (in agile project)	co-located	ethnomethodology	to understand how testing is conduc ed and to emphasise the importance of CSCW research to SE

located full-time at the study 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).

2.4.3 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 researching 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 fieldsite 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 multi-site ethnography for studies that involve more than one fieldsite 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 developers 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 environment and collecting and analyzing documents of different kinds [68], [109], [119].

TABLE 3
The Theoretical Underpinnings Most Used in Software Engineering and Related Scientific Discourses

Theoretical underpinnings	Basic concept	Focus of analysis	Main References	Examples in SE and related areas (not necessarily ethnographic studies)
Ethnomethodology	The social order of a group or community is continually reconstructed through the social interaction of the group.	Fine grained interaction analysis of specific encounters; how do members of the team, group or community (re)-establish their social order in the interaction.	[52]	[15], [78], [99], [117]
Distributed Cognition	Most cognitively challenging tasks are achieved through a 'cognitive system,' which entails interactions among people, the artefacts they use, and the environment they are working in. This contrasts with other cognitive approaches, by focusing on what is happening across a system of individuals and artefacts rather than what is happening inside the head of one individual.	How is information propagated through interactions between different media, i.e. how is information represented and re-represented as it moves through the cognitive system between individuals and artefacts that are used during activities, e.g. code, diagrams, sketches, spoken word.	[66]	[110], [111], [112]
Activity Theory	Human activity is purposeful, motivated by an anticipated outcome. However, all activity is mediated through technical or semiotic tools and organisational structures	Analysis focuses on identifying interacting activity systems consisting of actors, objects and outcomes, mediating tools, written or unwritten rules governing the collaboration, the community the actors belong to and the norms of this community, and division of labor involved in achieving the intended outcome	[47],[46]	[126], [83]
Actor Network Theory	Understanding social institutions, like science and academia or technology development, and their evolution can be done by studying the interaction of human and non-human actors. The latter can be, e.g. artefacts, tools, methods, or documents. The non-human actors represent black-box results of similar heterogeneous systems.	Analysis of empirical material focuses on the interaction of human and non-human actors (material objects like artefacts, tools, documents, and the like). Human and non-human actors are analyzed similarly.	[73], [72], [74]	[71], [27]

2.4.4 The Use of Theoretical Underpinnings

Ethnographic research does not come with a specific theoretical underpinning that should be used. However, there are several that are commonly used. Table 3 presents the theoretical underpinnings most used in software engineering and related scientific discourses. These theories act as lenses that focus on specific aspects of the field while de-emphasizing others. This means that a theoretical framework strongly influences not only how data analysis is conducted, but also which data is collected. For example, an Activity Theory-informed ethnography would use the notion of *activity* as the scope for observations and focus on the central elements of an activity system 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 outcome. In constrast, ethnography using a distributed cognition underpinning would focus on how the environment, 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 underpinning, 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 depends on the research question and the ethnographer's intent (see below). As ethnography applied in software engineering focuses on understanding software engineering 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 engineering context.

2.4.5 The Ethnographers' Intent in Undertaking the Study

Traditionally, ethnography investigates an unfamiliar culture and aims to understand it. In this case, how that understanding is acted upon, if at all, was not part of the ethnographers' intent. However in the early days of ethnography, there were some cases of commercial exploitation of the understanding achieved through ethnographic studies, e.g., with the results being used for targeted marketing. To

guard against this and other misuses, many proponents of ethnography today reject the notion of ethnographic results being used to exploit or change the culture it observes [56]. However, in the context of CSCW for instance, ethnographic studies are intended to provide not only an in depth understanding of a work practice and organizational culture, but also support for the design of specific functionality which may then change ways of working.

In the context of software engineering, it may be valuable 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 government or industry standards, such as software quality management 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 practitioners under study, may use that information to improve, 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 empirical researchers, understanding observed practices is only the starting point to improve them, maybe through evaluating methods, or developing tools and so on. Practitioners being studied may expect help with improving their practices, or addressing their problems and be puzzled if the researchers do not help them to improve, as they are perceived as peers with access to relevant knowledge about methods and tools.

The intent of the ethnographers will change the interaction between the researcher and the field [34]. It is therefore important that the researcher both is open about her research interest and her intent and also takes into account the way her research might impact the observed software development practice.

2.4.6 Designing the Ethnographic Study

When designing an ethnographic study the dimensions discussed above need to be taken into account. These dimensions 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 if the observed community meets for a sprint, participatory observation might require travelling and participation for as 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 resultant adaptation of ethnography into an identifiable approach. Examples include rapid ethnography [82] and contextual inquiry [62], mentioned in 2.2 above, and cognitive ethnography [5]. This last one is tailored for goal-directed development of technology through data collection in time-slices of observation, tight focus and verifiability through multiple data sets. Making such compromises 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 ethnography, which is to uncover the participants' viewpoint.

3 Ethnography in Software Engineering

Some of the most well-known research using ethnography in software engineering tried to integrate ethnographic studies into software engineering as a technique for requirements engineering [131]. This paper takes a different stance, it focuses on *ethnographic studies as a technique to study the community of software engineers and improve the way in which they work*.

The first studies using ethnographic methods to investigate software practice, conducted from a software engineering perspective, were implemented and published in the mid 1990s, often by interdisciplinary groups of researchers 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 recently (see Table 2). Another example is the work of Hovenden 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 ethnographic studies in empirical software engineering. The discussion is illustrated using the eight example studies introduced above; the way in which they relate to each of these four roles is shown in Table 4. These four roles are:

1. To strengthen investigations into the social and human aspects of software engineering
2. To inform the design of software engineering tools
3. To improve process development
4. To inform research programmes by articulating more specific research questions and complementing other research methods, such as code analysis and quantitative studies, thereby providing context grounded in practice

One study or series of studies may address more than one of these roles.

3.1 To Strengthen Research on Social and Human Aspects of Software Practice

People and their interactions are central to software development, 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 engineering tool development on empirical studies and findings from sociology [7], [81]; and Naur [85] who questioned the value of documentation as a sole communication tool, and emphasized the need to directly communicate the rationale behind the design.

TABLE 4
Example Studies Mapped to Their Main Role in Empirical Software Engineering

Example's short name	Social and human aspects	Software engineering tools	Process improvement	Inform research programmes	
				New research questions	Complement other methods
Agile	X			X	
Architecture			X	X	
Awareness	X	X			X
Bug report	X	X			X
Coordination	X				X
PyPy			X	X	X
Scientist			X	X	X
Testing	X			X (but only later)	

Research on the human and social aspects of software development has quite a broad scope, but its key characteristic 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 considered under this theme. Given its focus on culture and the informants' perspectives, it is not surprising that ethnography can strengthen investigations of the social and human aspects of software practice.

Research in this area often relies on interview data. Interviews are a good approach to use because they can elicit data from people relating to their thoughts and rationales, 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 account 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 interviews. Together with ethnography's emphasis on the informants' perspective, this approach overcomes any limitations of using self-reporting instruments alone. Questions in this area which might be answered using ethnography 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 methods?*

Example Agile indicates one set of studies where an ethnographic approach has been used to explore the social and human aspects of software engineering, in particular 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 artefacts [111], [112]. This finding was based on a number of individual studies in different settings, all with XP (extreme programming) practitioners and a range of underpinning theories including distributed cognition [66] and cognitive dimensions [57]. One of the consequences of this finding for practice is the need to compensate for social discipline when the social context of the team changes, 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 software development is a social process as much as a technical 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 promote user-developer cooperation. Their findings particularly highlight how the dynamics of a participatory and evolutionary process can be accommodated while at the same time making the process accountable and manageable from an organizational perspective. Avram et al.'s study emphasises the role of ethnography in uncovering the rationale for certain practices [4]. They used participant observation, among other methods, to investigate workflow at the project level and at local level, within a distributed team. Their results confirmed that imposing processes or workflows does not necessarily support the organization in meeting its goals, and that local arrangements may be supportive rather than obstructive.

In short, ethnographic studies of software practice focused on social and human aspects can expose the mechanisms used to 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.

3.2 To Inform the Design of Software Engineering Tools

The relationship between ethnographic research and design is by no means a simple one. In fact, it has been discussed 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 concurrent ethnography, which means that the ethnographic investigation of the work and the systems design proceed in parallel (see Fig. 1). Debriefing meetings between ethnographers and designers are held to "identify, discuss, and elaborate issues of relevance to design" [22: 91].

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 and other modes of social science might mean and just what their investigative methods can contribute. I do not believe this is a straightforward, quick, or painless process." Dourish and Button [42]

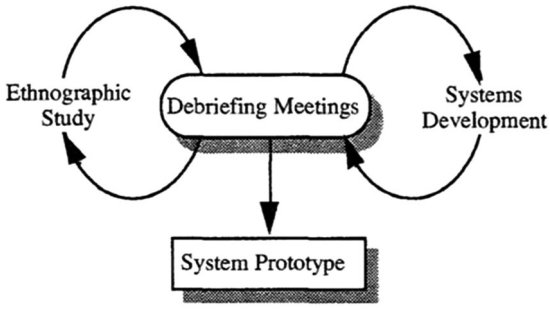

Fig. 1. Concurrent ethnography and design [64].

highlighted an issue which is crucial to software engineering research too: can a detailed 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 and the 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 discussion assumes that ethnograhers do not possess design skills [22: 95], and therefore, that ethnographers and designers are different stakeholders. This translates the relationship 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 designer are one and the same person. Example Awareness illustrates 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 communication mechanisms, therefore both facilitating and hindering 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 ethnographic 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 dependency analysis are used to support the identification of software developers who are working on similar issues, 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 colleagues [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 ethnographic 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 observed in these companies.

Being a software engineering researcher and an ethnographer in the context of tool design is not without problems, however. The most important problem to be addressed 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 implementing something, when it is unclear which of these hypotheses is true. Instead, the researcher should hypothesize 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. Furthermore, the appropriation of the design will provide additional information about which aspects of the supported practice interact with the task the tool is designed to support. Another risk associated with early design decisions 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 research progressed they realized it was more important to focus on the overall communication among the different teams. Combining these two tools would be only one limited part of the solution. Again, they reached this conclusion by hypothesizing about solutions and tools, and later on validating them in following data collection periods.

3.3 To Inform Software Process and Method Improvement

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 becomes involved in the deliberation of change and the exploration of methods, though, the role of the researcher changes from a pure observer to an actor and sometimes even a change agent.

The way methods are used informs practice in a different, less visible and less accountable form than tool design and usage. This has been discussed in research

focussing on cooperative aspects of software engineering, e.g., [51] and [79]. As other ethnographic research has shown, the application of methods is not as straightforward as many software engineering textbooks imply [15]: methods, processes and techniques, despite their prescriptive nature, are still subject to interpretation in the specific context. Applying ethnographic research to focus on the situated implementation and adaptation of methods, processes and tools in software practice leads to an understanding of how methods, processes and techniques actually influence that practice, what specifically influences the interpretation and appropriation of methods, what makes them work, and what prohibits their application. Two examples that illustrate this role of ethnography 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 circumstances [100]. The former is an example of how different processes and techniques together facilitate a participatory 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 interested in improving their practice by applying these innovations. This sometimes leads to the situation that software practitioners as well as their managers expect recommendations 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 recommendations to improve control. In turn, the practitioners involved 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 action research as a framework for making the research and intervention accountable both towards the research community and the research setting [34], [40]. Their Cooperative 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:

1. an action research cycle consisting of three phases: empirical studies, deliberation of changes, and introduction and research of the changes decided;
2. the empirical research is ethnographically inspired;
3. focusing on the everyday shop floor software development practice;
4. improvements are deliberated and evaluated from the shop floor software development perspective; and
5. the changes are deliberated and implemented together with the practitioners.

Example Architecture illustrates this role for ethnographic studies in empirical software engineering. The study focused on the processes and practices around agile architecture, and applied the above-described Cooperative Method Development approach [40]. The study provided an understanding of the architectural practices that were used in this context, and showed how heavyweight architecture practices that focus on comprehensive documentation 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 architecture 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.

3.4 To Inform Research Programmes

Ethnographic accounts have the potential to highlight under-researched areas of software engineering, to identify more specific research questions, and to complement other research methods by providing context for them. For example, Example Agile highlighted the importance of physical artefacts plus social practices to maintain discipline 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 dispersed, 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 Sections 3.1–3.3) or it may be used to generate new research questions or complement other methods by providing context.

3.4.1 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 common 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 distributed nature of the project. It was hoped that formal and informal discussions prompted by the researchers would support team members to reflect on their own practices. However this initial focus on distribution was overridden by the research analysis which pointed to conflicting cultural values and practices between the professional software developers and the scientists as scientific end-user developers. The analysis of Example PyPy began with an industrial case study looking at how Open Source teams handle the

challenges of distributed development. This initial study led to a more specific focus on how changes to common practice are negotiated, and implemented in order to address these challenges.

3.4.2 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 under study, useful contextual information was available to help 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 useful context, but it is of a different nature.

Instead of collecting just measurements or reported activities, the ethnographic investigation reveals how the measurements come about and what the reported activities look like in practice. Observations may also support making connections between various data sets. For instance, when investigating the use of instant messaging in distributed 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 ethnography to complement other research methods. Example Bug report used interviews and elements of case study research; Example Coordination complemented ethnography with interviews and document analysis; and Example 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 understanding of why experienced practitioners might deviate. Based on ethnographic studies and complementary interviews, Example Architecture proposes that there might be good reasons why software architects do not use architectural 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.

4 Discussion

The examples throughout this paper illustrate what can be achieved when ethnography is used to investigate software 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 ethnographic approach is taken.

4.1 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 software practitioners achieve useful and usable software in an imperfect environment: what difficulties they are facing and what, for them, has proven beneficial to address these difficulties. This fine-grained activity often only becomes visible when it becomes problematic, e.g., when coordinating software engineering practices fail over distances [25].

Understanding the fine-grained activity of software practice, and its rationalities increases the chance that improvements in software practices built on these findings will be sustainable, because they take account of local conditions and embed existing local expertise. Specifically, this understanding helps to: identify what not to disturb when introducing new methods, tools and techniques; 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 successfully.

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 require 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 distributed 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 mechanisms 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 understand as a proper implementation of this method. However, the appropriation takes place for good professional reasons. Rather than being read as a sign of incompetence, these appropriations can be seen as a reaction to the mismatch of the rationalities of practice and the method to be applied [98].

How to design software tools. Example Awareness illustrates the different roles played by APIs in the coordination 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 collaborative software development practice.

How projects are kept on track. Example PyPy investigated how distributed development projects, private as well as open 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 practice 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.

4.2 When is Ethnography Appropriate

Ethnography is not the only, nor the best qualitative approach to be used in all circumstances, but the more thorough understanding of software practice that ethnography 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 appreciation of practice and practitioners' points of view will improve 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 software engineering to promote qualitative research with this idea. However it is also appropriate even within areas 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 collaboration arise. Similar practices of articulation and meta-work were thereafter observed in the PyPy project. Such renegotiation of coordination and cooperation practices have not been reported previously from studies in empirical software engineering. Similarly, Example Awareness looked at a technical construct (APIs) and uncovered their role in the coordination of software development work, which was a new insight. In both examples we see that the ethnographic research unearthed unexpected, 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 aspects of software engineering practice in the centre, and is therefore very well-suited to any research question focusing 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 answered [95]. For example, asking 'How do software practitioners develop systems using XP?' rather than 'Is single programmer coding more productive than pair programming?' 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?'

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 improvements are sought, because they can then take account of local conditions and embed existing local expertise.

There are many (empirical) software engineering questions that an ethnographic study may address: What is so interesting about discussions of APIs (Example Awareness)? How do software engineers use graphical representations 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 distributed 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 development 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. Radical changes in tools, techniques, methods and processes, cannot be *supported* by ethnographic research. Ethnographic methods rely on existing practices. If a tool, technique or method is supposed to radically change a practice, ethnographic research is of little help in effecting that change, although it can be used to *evaluate* the proposed practices in pilot studies.

4.3 Why a Software Engineer Should Consider Being the Ethnographer

The relationship between the ethnographer and her informants is key. If the ethnographer is a software engineer, 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 respect is a significant force in the software practitioner community, and we have found that software practitioners 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 engineer 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 stance and a suitable scientifically accountable manner. One framework to handle this (the Co-operative Development Method) was discussed in Section 3.3.

4.4 How the Dimensions and Roles Relate to Each Other

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 engineering (Section 3). So how do these dimensions and roles relate 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 important 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 human aspects of software development 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 collection on site.

In contrast, using ethnographic studies to inform software 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 and focus 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 ethnographic research allows the adaptation of the research design based on initial findings, therefore the kind of engagement and participation often will need to be adapted during the course of the study.

5 Conclusion

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

1. a focus on the members' point of view allows the rationalities of practice to be made explicit, and hence exposes *why* software engineers do what they do;
2. a focus on the ordinary detail of life emphasises *local* context and *local* expertise which can be overlooked when using other research approaches;
3. the analytical stance makes the results of an ethnographic study more than a simple account of activity; and
4. the production of 'thick descriptions' supports academic accountability.

Section 3 highlighted four significant roles that ethnographic studies of software practice can and have fulfilled. 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 studies may play in empirical software engineering (see Table 4).

By investigating and articulating both the work behaviour and the rationalities behind that behaviour, meaningful and sustainable improvements can be researched, devised 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 decisions about the usage of the ethnographic method within the specific context and circumstances. In Section 2.4, we discussed five dimensions along which the studies reviewed for this article differ: the degree and kind of participation, the duration of the study, the space and location of the study, the theoretical underpinning applied, and the researcher's 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 investigated, the context of the fieldwork and the characteristics of the main focus of the study. As mentioned above, ethnographic 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 their ability to uncover the rationalities of the observed practices. Therefore, ethnographic studies provide an important complement to other empirical research methods, like experiments, that rely on prior formulation of hypotheses. Ethnographic studies help understand *how*, and more importantly, *why*, software teams do the things that they 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 findings 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.

Ackowledgments

The authors wish to thank attendees at their tutorials during ICSE over the last 5 years, and all the collaborators who have supported our ethnographic studies. The third author would like to thank the funding from CNPq (process numbers 440880/2013–0 and 310468/2014–0).

References

[1] U. Abelein, H. Sharp, and B. Paech, "Does involving users in software development really influence system success?" *IEEE Softw.*, vol. 30, no. 6, pp. 17–23, Nov./Dec. 2013.

[2] B. Anderson, *Work, ethnography and system design. The encyclopedia of microcomputers*, A. Kent and J. G. Williams, Eds. New York, NY, USA: Marcel Dekker, 1997, pp. 159–183.

[3] Y. Anzai and H. A. Simon, "The theory of learning by doing," *Psychological Rev.*, vol. 86, pp. 124–140, 1979.

[4] G. Avram, L. Bannon, J. Bowers, A. Sheehan, and D.K. Sullivan, "Bridging, patching and keeping the work flowing: Defect resolution in distributed software development," *Comput. Supported Cooperative Work*, vol. 18, nos. 5/6, pp. 477–507, 2009.

[5] L. J. Ball and T. C. Ormerod, "Putting ethnography to work: The case for a cognitive ethnography of design," *Int. J. Human-Comput. Stud.*, vol. 53, no. 1, pp. 147—168, Jul. 2000.

[6] R. M. Belbin *Team Roles at Work*, 2nd ed. London, U.K.: Butterworth Heinemann, 2012.

[7] S. Bendifallah and W. Scacchi, "Work structures and shifts: An empirical analysis of software specification teamwork," in *Proc. 11th Int. Conf. Softw. Eng.*, 1987, pp. 260–270.

[8] J. Blomberg and H. Karasti, "Reflections on 25 years of ethnography in CSCW," *Comput. Supported Cooperative Work*, vol. 22, pp. 373–423, 2013.

[9] J. Blomberg and H. Karasti, "Positioning ethnography within participatory design," in *Routledge International Handbook of Participatory Design*, Simonsen, J. and Robinson T., Eds. London, U.K.: Routledge, 2012.

[10] M. Blumer, *The Chicago School of Sociology: Institutionalization, Diversity, and the Rise of Sociological Research*. Chicago, IL, USA: Univ. Chicago Press, 1984.

[11] A. Boden, G. Avram, L. Bannon, and V. Wulf, "Knowledge sharing practices and the impact of cultural factors: Lessons from two case studies of offshoring in SME," *Softw Maintenance Evol.: Res., Practice*, vol. 24, no. 2, pp. 139–152, 2012.

[12] A. Boden, F. Rosswog, G. Stevens, and V. Wulf, "Articulation spaces: Bridging the gap between formal and informal coordination," in *Proc. 17th Conf. Comput. Supported Cooperative Work Social Comput.*, 2014, pp. 1120–1130.

[13] O. Bondarenko and R. Janssen, "Documents at hand: Learning from paper to improve digital technologies," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2005, pp. 121–130.

[14] G. Button, "The ethnographic tradition and design," *Design Stud.*, vol. 21, pp. 319–332, 2000.

[15] G. Button and W. Sharrock, "Occasioned practices in the work of software engineers," in *Requirements Engineering: Social and Technical Issues.* M. Jirotka and J. Goguen, Eds. New York, NY, USA: Academic, 1994, pp. 217–240.

[16] A. Capiluppi, J. F. Ramil, J. Higman, H. Sharp, and N. Smith, "An empirical study of evolution patterns for an agile system, that combines qualitative and quantitative approaches," in *Proc. 29th Int. Conf. Softw. Eng.*, 2007, pp. 511–518.

[17] J. Carroll and M. B. Rosson, "Better home shopping or new democracy? Evaluating community network outcomes," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2001, pp. 372–379.

[18] P. H. Carstensen and C. Sørensen, "From the social to the systematic," *Comput. Supported Coop. Work*, vol. 5, no. 4, pp. 387–413, Dec. 1996.

[19] P. H. Carstensen, C. Sorensen, and T. Tuikka, "Let's talk about bugs," *Scandinavian J. Inform. Syst.*, vol. 7, pp. 33–33, 1995.

[20] M. Cataldo, P. A. Wagstrom, J. B. Herbsleb, and K. M. Carley, "Identification of coordination requirements: Implications for the design of collaboration and awareness tools," in *Proc. 20th Anniversary Conf. Comput. Supported Cooperative Work*, 2006, pp. 353–362.

[21] M. L. Cohn, S. E. Sim, and C. P. Lee, "What counts as software process? Negotiating the boundary of software work through artifacts and conversation," *Comput. Supported Cooperative Work*, vol. 18, no. 5–6, pp. 401–443, 2009.

[22] A. Crabtree, *Designing Collaborative Systems*. New York, NY, USA: Springer, 2003.

[23] A. Crabtree, T. Rodden, P. Tolmie, and G. Button, "Ethnography considered harmful," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2009, pp. 879–888.

[24] A. Crabtree, M. Rouncefield, and P. Tolmie, *Doing Design Ethnography*. New York, NY, USA: Springer, 2012.

[25] D. Damian, L. Izquierdo, J. Singer, and I. Kwan, "Awareness in the wild: Why communication breakdowns occur," in *Proc. IEEE 2nd Int. Conf. Global Softw. Eng.*, 2007 pp. 81–90.

[26] C. R. B. De Souza, "On the relationship between software dependencies and coordination: Field studies and tool support," Ph.D. dissertation, Donald Bren School *Inform. Comput. Sci.*, Univ. California, Irvine, CA, USA, 2005.

[27] C De Souza, J. Froehlich, and P. Dourish, "Seeking the source: Software source code as a social and technical artifact," in *Proc. Int. ACM SIGGROUP Conf. Supporting Group Work,* 2005, pp. 197–206.

[28] C. R. B. De Souza and D. F. Redmiles, "An empirical study of software developers management of dependencies and changes," in *Proc. Int. Conf. Softw. Eng.*, 2008, pp. 241–250.

[29] C. R. B. De Souza and D. F. Redmiles, "On the roles of APIs in the coordination of collaborative software development," *Comput. Supported Cooperative Work*, vol. 18, pp. 445–475, 2009.

[30] C. R. B. De Souza, D. Redmiles, L.-T. Cheng, D. Millen, and J. Patterson, "How a good software practice thwarts collaboration - The multiple roles of APIs in software development," in *Proc. 12th ACM SIGSOFT Twelfth Int. Symp. Foundations Softw. Eng.*, Newport Beach, CA, USA, 2004, pp 221–230.

[31] C. R. B. De Souza, D. F. Redmiles, G. Mark, J. Penix, and M. Sierhuis, "Management of interdependencies in collaborative software development," in *Proc. ACM-IEEE Int. Symp. Empirical Softw. Eng.*, Rome, Italy, 2003, pp. 196–203.

[32] C. R. B. De Souza, H. Sharp, J. Singer, L.-T. Cheng, and G. Venolia, "Guest Editors' Introduction: Cooperative and human aspects of software engineering," *IEEE Softw.*, vol. 26, no. 6, pp. 17–19, Nov./Dec. 2009.

[33] F. Detienne and F. Bott, *Software Design-Cognitive Aspects*. New York, NY, USA: Springer-Verlag, 2002.

[34] Y. Dittrich, "Doing empirical research on software development: Finding a path between understanding, intervention, and method development," in *Social Thinking – Software Practice*, Y. Dittrich, C. Floyd, and R. Klischewski, Eds. Cambridge, MA, USA: MIT Press, 2002, pp. 243–262.

[35] Y. Dittrich, "What does it mean to use a method? Towards a practice theory for software engineering," *Inform. Softw. Technol.*, vol. 70, pp. 220–231, 2016.

[36] Y. Dittrich and R. Giuffrida, "Exploring the role of instant messaging in a global software development project," in *Proc. 6th IEEE Int. Conf. Global Softw. Eng.*, 2011, pp. 103–112.

[37] Y. Dittrich, M. John, J. Singer, and B. Tessem, "Editorial for the special issue on qualitative software engineering research," *Inform. Softw. Technol.*, vol. 49, no. 6, pp. 531–539, 2007.

[38] Y. Dittrich and O. Lindeberg, "How use-orientated development can take place," *Inform. Softw. Technol.*, vol. 46, pp. 603–617, 2004.

[39] Y. Dittrich, D. W. Randall, and J. Singer, "Software engineering as cooperative work," *Comput. Supported Cooperative Work*, vol. 18, no. 5–6, pp. 393–399, 2009.

[40] Y. Dittrich, K. Rönkkö, J. Eriksson, C. Hansson, and O. Lindeberg, "Cooperative method development," *Empirical Softw. Eng.*, vol. 13, no. 3, pp. 231–260, 2008.

[41] P. Dourish and G. Button, "On technomethodology: Foundational relationships between ethnomethodology and system design," *Human-Computer Interaction*, vol. 13, no. 4, pp. 395–432, 1998.

[42] P. Dourish, *Where the Action Is: The Foundations of Embodied Interaction*. Cambridge, MA, USA: MIT Press, 2004.

[43] DSDM: H. Sharp, L. Plonka, P. Gregory, K. Taylor and, M. Rowlands. (2015). *DSDM and UX Design*, DSDM Consortium, ISBN 978–1–910961–00–1.

[44] T. Dybå, B. A. Kitchenham, and M. Jørgensen, "Evidence-based software engineering for practitioners," *IEEE Softw.*, vol. 22, no. 1, pp. 58–65, Jan./ Feb. 2005.

[45] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. Sjøberg, Eds. New York, NY, USA: Springer, 2008, pp. 285–311.

[46] Y. Engeström, "Expansive visibilization of work: An activity-theoretical perspective," *Comput. Supported Cooperative Work*, vol. 8, no. 1, pp. 63–93, 1999.

[47] Y. Engeström, R Miettinen, and R.-L. Punamäki, *Perspectives Activity Theory*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[48] J. Ferreira, H. Sharp, and H. Robinson, "Agile development and user experience design integration as an on-going achievement in practice," in *Proc. AgileConf.*, 2012, pp. 11–20.

[49] D. M. Fetterman, *Ethnography*, 2nd ed. Thousand Oaks, CA., USA: Sage, 1998.

[50] C. Floyd, "Outline of a paradigm change in software engineering," in *Computers and Democracy*, G. Bjerknes, P. Ehn, M. Kyng, Eds. Aldershot, U.K.: Avebury, 1987, pp. 192–210.

[51] C. Floyd, "Software development as reality construction," in *Software Development Reality Construction*, C. Floyd, H. Züllighoven, R. Budde, and R. Keil-Slawik, Eds. Berlin, Germany: Springer-Verlag, 1992, pp. 86–100

[52] Garfinkel, *Studies in Ethnomethodology*. New York, NY, USA: Wiley, 1967.

[53] C. Geertz, *Works and Lives: The Anthropologist as Author*. Stanford, CA, USA: Stanford Univ. Press, 1988.

[54] R. S. Geiger and D. Ribes, "Trace ethnography: Following coordination through documentary practices," in *Proc. IEEE 44th Hawaii Int. Conf. Syst. Sci.*, 2011, pp. 1–10.

[55] R. Giuffrida and Y. Dittrich, "Exploring the role of instant messaging in a global software development project," in *Proc. 6th IEEE Int. Conf. Global Softw. Eng.*, , 2011, pp. 103–112

[56] D. P. S. Goh, "States of ethnography: Colonialism, resistance, and cultural transcription in Malaya and the Philippines, 1890s-1930s," *Comparative Stud. Soc. History*, vol. 49, pp. 109–142, 2007.

[57] T. R. G. Green, "Cognitive dimensions of notations," in *People and Computers V.*, A. Sutcliffe, L. Macaulay, Eds. Cambridge, U.K.: Cambridge Univ. Press, 1989, pp. 443–460.

[58] R. E. Grinter, "Recomposition: Coordinating a web of software dependencies, *Comput. Supported Cooperative Work*, vol. 12, no. 3, 297–327, 2003.

[59] M. Hammersley and P. Atkinson, "What is Ethnography," in *Ethnography: Principles in Practice*. London, U.K.: Routledge, 1995, pp. 9–25.

[60] C. Heath and P. Luff, "Collaboration and control: Crisis management and multimedia technology in London underground line control rooms," *Comput. Supported Cooperative Work*, vol. 1, pp. 69–94, 1992.

[61] C. Hine, *Virtual Ethnography*. Thousand Oaks, CA, USA: Sage, 2000.

[62] K. Holtzblatt and S. Jones, "Contextual Inquiry: A participatory approach for systems design," in *Particiopatory Design: Principles and Practice*, D. Schuler and A. Namioka, Eds. Hillsdale, NJ, USA: Lawrence Erlbaum Assoc., 1993, pp. 177–210.

[63] F. M. Hovenden, S. Walker, H. C. Sharp, and M. Woodman, "Building quality into scientific software," *Softw. Quality J.*, vol. 5, pp. 25–32, 1996.

[64] J. Hughes, V. King, T. Rodden, and H. Andersen, "Moving out from the control room: Ethnography in system design," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 1994, pp. 429–439.

[65] W. V. Humboldt (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, 2nd rev. ed. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[66] E. Hutchins, *Cognition in the Wild*. Cambridge, MA, USA: MIT Press, 1995.

[67] H. Hutchinson, W. Mackay, B. B. Westerlund Bederson, A. Druin, C. Plaisant, M. Beaudouin-Lafon, S. Conversy, H. Evans, H. Hansen, N. Roussel, B. Eiderbäck, S. Lindquist, and Y. Sundblad, "Technology probes: Inspiring design for and with families," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2003, pp. 17–24.

[68] B. Jordan, "Blurring boundaries: The "real" and the "virtual" in hybrid spaces," *Human Org.*, vol. 68, no. 2, pp. 181–193, 2009.

[69] D. L. Jorgensen, *Participant Observation: A Methodology for Human Studies*. Thousand Oaks, CA, USA: Sage, 1989.

[70] B. Kitchenham, et al., "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, Aug. 2002.

[71] R. Klischewski, "Reaching out for commitments: Systems development as networking," in *Social Thinking – Software Practice*, Y. Dittrich, C. Floyd, and R. Klischewski, Eds. Cambridge, MA, USA: MIT Press, 2002, pp. 309–329.

[72] B. Latour, *Science in Action: How to Follow Scientists and Engineers through Society*. Milton Keynes, U.K.: Open Univ. Press, 1987.

[73] B. Latour, "Where are the missing masses? The sociology of a few mundane artifacts," in *Shaping Technology/Building Society: Studies in Sociotechnical Change*, W. Bijker and J. Law, Eds. Cambridge, MA, USA: MIT Press, 1994, pp. 225–258.

[74] J. Law and J. Hassard, "Actor network theory and after," in *Sociological Review*, Hoboken, NJ, USA: Wiley, 1999.

[75] J. Low, J. Johnson, P. Hall, F. Hovenden, J. Rachel, H. Robinson, and S. Woolgar, "Read this and change the way you feel about software engineering," *Inform. Softw. Technol.*, vol. 38, pp. 77–87, 1996.

[76] B. Malinowski, *Argonauts of the Western Pacific*. London, U.K.: Routledge (1967), 1922.

[77] G. E. Marcus, "Ethnography in/of the world system: The emergence of multi-sited ethnography," *Annu. Rev. Anthropol.*, vol. 24, pp. 95–117, 1995.

[78] D. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville, 'Good'organisational reasons for 'Bad' Software testing: An ethnographic study of testing in a small software company," in *Proc. IEEE 29th Int. Conf. Sof. Eng.*, 2007, pp. 602–611.

[79] L. Mathiassen, "Reflective systems development," *Scandinavian J. Inform. Syst.*, vol. 10, nos. 1/2, pp. 67–118, 1998.

[80] J. E. McGrath, "Methodology matters: Doing research in the behavioral and social sciences," in *Proc. Human-Comput. Interaction*, San Francisco, CA, USA: Morgan Kaufmann, 1994, pp. 152–169.

[81] P. Mi and W. Scacchi, "Modeling articulation work in software engineering processes," in *Proc. 1st Int. Conf. Softw. Process*, 1991, pp. 188–201.

[82] D. Millen, "Rapid ethnography: Time deepening strategies for HCI field research," in *Proc. 3rd Conf. Designing Interactive Syst.: Processes, Practices, Methods, Techn.*, 2000, pp. 280–286.

[83] A. S. Mursu, I. Luukkonen, M. Toivanen, and M. J. Korpela, "Activity theory in information systems research and practice: Theoretical underpinnings for an information systems development model," *Inform. Res.*, vol. 12, no. 3, p. 3, 2006.

[84] B. Nardi, "The use of ethnographic methods in design and evaluation," in *Handbook of Human-Computer Interaction*, M. Helander, T. K. Landauer, and E. P. Prabhu, Eds. Amsterdam, The Netherlands: Elsevier, 1997, pp. 361–366.

[85] P. Naur, "Programming as theory building," *Microprocess. Microprogram.*, vol. 15, no. 1985, pp. 253–261, 1985.

[86] P. Naur and B. Randell, *Software Engineering. Report of a Conference Sponsored By the NATO Science Committee*, Garmisch, Germany: Brussels, Scientific Affairs Division, NATO, Oct. 7–11, 1968.

[87] D. Nicolini, *Practice Theory, Work, and Organization: An Introduction*. London, U.K.: Oxford Univ. Press, 2013.

[88] K. Nygaard, "Program development as a social activity," in *Proc. Inform. Process.86. IFIP*, 1986, pp. 189–198.

[89] S. O'Riain, "Net-working for a living: Irish software developers in the global workplace," in *Blackwell Cultural Economy Reader*, Oxford, U.K.: Blackwell, 2000, pp. 15–39.

[90] C. Passos, D. S. Cruzes, T. Dybå, and M. Mendonça, "Challenges of applying ethnography to study software practices," in *Proc. ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2012, pp. 9–18 .

[91] G. Poderi, "Simple conversational practices in the case of free and open source software infrastructure," in *Proc. 12th Participatory Design Conf.: Exploratory Papers, Workshop Descriptions, Industry Cases-Volume*, 2012, vol. 2, pp. 45–48.

[92] G. Poderi, "Making sense of users participation in open source projects: The case of a mature video game," Ph.D. dissertation, Univ. Trento, Trento, Italy, 2013.

[93] J. Preece, Y. Rogers, and H. Sharp *Interaction Design*, 4th ed. Hoboken, NJ, USA: Wiley, 2015.

[94] J. R. Prior, T. J. Robertson, and J. R. Leaney, "Situated software development: Work practice and infrastructure are mutually constitutive," in *Proc. 19th Australian Softw. Eng. Conf.*, 2008, pp. 160–169.

[95] H. M. Robinson, J. Segal, and H. Sharp, "Ethnographically-informed empirical studies of software practice," *Inform. Softw. Technol.*, vol. 49, no. 6, pp. 540–551, 2007.

[96] C. Robson, *Real World Research*, 3rd ed. Hoboken, NJ, USA: Wiley, 2011.

[97] K. Rönkkö "Ethnography," in *Encyclopedia of Software Engineering*, P. Laplante, Ed. New York, NY, USA: Taylor & Francis, 2010.

[98] K. Rönkkö, Y. Dittrich, and O. Lindeberg, "'Bad practice' or 'bad methods'—are software engineering and ethno-graphic discourses incompatible?" in *Proc. 1st Int. Symp. Empirical Softw. Eng.*, Nara, Japan, 2002, pp. 204–210.

[99] K. Rönkkö, Y. Dittrich, and D. Randall, "When plans do not work out: How plans are used in software development projects," *Comput. Supported Cooperative Work*, vol. 14, no. 5, pp. 433–468, 2005.
[100] K. Ronkko, M. Hellman, B. Kihlander, and Y. Dittrich, "Personas is not applicable: Local remedies interpreted in a wider context," in *Proc. 8th Conf. Participatory Design Conf. Artful Integr.: Interweaving Media, Mater. Practice*, Toronto, ON, Canada, Jul. 27–31 2004, pp. 112–120.
[101] J. Rooksby, M. Rouncefield, and I. Sommerville, "Testing in the wild: The social and organizational dimensions of real world practice," *Comput. Supported Cooperative Work*, vol. 18, pp. 559–580, 2009.
[102] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. New York, NY, USA: Wiley, 2012.
[103] R. J. Sach, H. Sharp, and M. Petre, "Software engineers' perceptions of factors in motivation," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Sept. 2011, pp. 368–371.
[104] C. Seaman, "Qualitative methods in empirical studies of software engineering," in *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 557–572, Jul./Aug. 1999.
[105] J. Segal, "Software development cultures and cooperation problems: A field study of the early stages of development of software for a scientific community," *Comput. Supported Cooperative Work*, vol. 18, nos. 5/6, pp. 581–606, 2009.
[106] J. Segal and S. Clarke, "Software engineers don't know everything about end-user programming," *IEEE Softw.*, vol. 26, no. 5, p. 55, Sep./Oct. 2009.
[107] D. Shapiro, "The limits of ethnography: Combining social sciences for CSCW," in *Proc. Conf. Comput. Supported Cooperative Work*, 1994, pp. 417–428.
[108] H. Sharp, "The role of physical artefacts in agile software development team collaboration," in *Proc. Physicality*, 2007, pp. 61–64.
[109] H. Sharp, R. Giuffrida, and G. Melnik, "Information flow in a dispersed agile team: A distributed cognition perspective," in *Proc. 13th Int. Conf. Agile Process. Softw. Eng. Extreme Program.*, Malmo, Sweden, 2012, pp. 62–76.
[110] H. Sharp and H. M. Robinson, "An ethnographic study of XP practices," *Empirical Softw. Eng.*, vol. 9, no. 4, pp. 353–375, 2004.
[111] H. Sharp and H. M. Robinson, "Collaboration and co-ordination in mature eXtreme programming teams," *Int. J. Human-Comput. Stud.*, vol. 66, pp. 506–518, 2008.
[112] H. Sharp, H. M. Robinson, and M. Petre, "The role of physical artefacts in agile software development: Two complementary perspectives," *Interacting Comput.*, vol. 21, no. 1–2, pp. 108–116, 2009
[113] H. Sharp, H. Robinson, and M. Woodman, "Software engineering: Community and culture," *IEEE Soft.*, vol. 17, no. 1, pp. 40–47, Jan./Feb. 2000.
[114] H. Sharp, H. Robinson, and M. Woodman, "Using ethnography and discourse analysis to address software quality issues," in *Proc. Workshop Beg, Borrow Steal: Using Multi-Disciplinary Approaches iEmpirical Softw. Eng. Res.*, 2000.
[115] H. Sharp, M. Woodman, and F. Hovenden, "Tensions in the adoption and evolution of software quality management systems: A discourse analytic approach," *Int. J. Human Comput. Stud.*, vol. 61, no. 2, pp. 219–236, 2004.
[116] F. Shull, J. Singer, and D. Sjoberg, Shull, Eds., "Advanced topics in empirical software engineering: An edited volume," in *Guide to Advanced Empirical Software Engineering*, New York: Springer, 2008.
[117] A. Sigfridsson, "The purposeful adaptation of practice: an empirical study of distributed software development," Ph.D. dissertation, Univ. Limerick, Limerick, Ireland, 2010.
[118] A. Sigfridsson and A. Sheehan, "On qualitative methodologies and dispersed communities: Reflections on the process of investigating an open source community," *Inform. Softw. Technol.*, vol. 53, no. 9, pp. 981–993, 2011.
[119] A. Sigfridsson, G. Avram, A. Sheehan, and D. Sullivan, "Sprint-driven development: Working, learning and the process of enculturation in the PyPy community," in *Open Source Development, Adoption Innovation*, New York, NY, USA: Springer, 2007, pp. 133–146.
[120] S. E. Sim, J. Singer, and M.-A. Storey, "Beg, borrow, or steal: Using multidisciplinary approaches in empirical software engineering research, an ICSE 2000 workshop report," *Empirical Softw. Eng.*, vol. 6, no. 1, pp. 85–93, 2001.
[121] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil, "An examination of software engineering work practices," in *Proc. Conf. Centre Adv. Stud. Collaborative Res. IBM*, 1997, p. 21.
[122] D. I. K. Sjøberg, T. Dybå, and M. Jørgensen, "The future of empirical methods in software engineering research," in *Proc. Future Softw. Eng.*, 2007, pp. 358–378.
[123] L. Suchman, *Plans and Situated Actions*. Cambridge, U.K.: Cambridge Univ. Press, 1987.
[124] L. Suchman and R. H. Trigg, "Artificial Intelligence as craftwork," in *Understanding Practice: Perspectives on Activity and Context*, S. Chaiklin and J. Lave, Eds. Cambridge UK: Cambridge Univ. Press, 1993.
[125] S. Teasley, L. Covi, M. S. Krishnan, and J. S. Olson, "How does radical collocation help a team succeed?" in *Proc. ACM 2000 Conf. Comput. Supported Cooperative Work*, 2000, pp. 339–346.
[126] P. Tell and M. A. Babar, "Activity theory applied to global software engineering: Theoretical foundations and implications for tool builders," in *Proc. 7th IEEE Int. Conf. Global Softw. Eng.*, 2012, pp. 21–30.
[127] E. Trainer, S. Quirk, C. de Souza, and D. Redmiles, "Bridging the gap between technical and social dependencies with Ariadne," in *Proc. OOPSLA Workshop Eclipse Technol. eXchange*, New York, NY, USA, 2005, pp. 26–30.
[128] J. Trimble, R. Wales, and R. Gossweiler, *NASA position paper for the CSCW 2002 workshop on Public, Community and Situated Displays: MERboard*, 2002.
[129] H. Unphon, "*Re-engineering for Evolvability*," Ph.D. disertation, IT Univ. Copenhagen, Copenhagen, Denmark, 2009.
[130] H. Unphon and Y. Dittrich, "Architecture awareness in long-term software product evolution," *J. Syst. Softw.*, vol. 83, pp. 2211–2226, 2010.
[131] S. Viller and I. Sommerville, "Coherence: An approach to representing ethnographic analyses in systems design," *Human—Computer Interaction 14*, Special issue on representations in interactive systems development, vol. 14, pp. 9–41, 1999.
[132] G. Walsham, "Cross-cultural software production and use: A structurational analysis," *MIS Quart.*, vol. 26, no. 4, pp. 359–380, 2002.
[133] J. Weber and J. Cheng. (2013). Making the most of ethnographic research. UX magazine [Online]. Avaliable: http://uxmag.com/articles/making-the-most-of-ethnographic-research
[134] G. Weinberg, *The Psychology of Programming*. New York, NY, USA: Dorset House, 1971.
[135] Y. Yoon and B. Myers, "A longitudinal study of programmers' backtracking," in *Proc. IEEE Symp. Visual Languages Human-Centric Comput.*, 2014, pp. 1–8.

Helen Sharp received the BSc (Hons), MSc, PhD, and PGCTLHE degrees in 1981, 1982, 1988, and 2003, respectively. She is currently a Professor of software engineering at the Open University. Her research focuses on the study of professional software practice and the effect of human and social aspects on software development. She has been conducting qualitative studies of software practice since the early 1990s and has produced over 100 peer-reviewed articles. She is very active in both the software engineering and interaction design (HCI) communities and has had a long association with practitioner-related conferences. She is a joint author of one of the leading textbooks on Interaction Design (id-book.com), now in its fourth edition. She is an associate editor of TSE, on the advisory board of *IEEE Software*, the editorial board of JSS, and reviews for many journals and conferences. She is a member of the *IEEE Computer Society*, BCS, and *ACM*.

Yvonne Dittrich received the Dipl-Inform, PhD degrees in (1989), and (1997), respectively. She was at the Blekinge Institute of Technology, Sweden. She is currently an associate professor at the IT University of Copenhagen, Denmark. Her research interests are use-oriented design and development of software and software development as cooperative work. She has been applying ethnographically inspired empirical methods in research cooperation with industrial partners since 1997 and developed "Cooperative Method Development," a research approach that combines ethnography and software process improvement. Through editing special issues in leading journals she contributed to the establishment of the research area "Cooperative and Human Aspects of Software Engineering." She is contributing to software engineering, computer supported cooperative work and participatory design communities. Her publication list counts over 60 peer-reviewed articles. She is regularly reviewing for major software engineering journals and is member of the ACM, the German Informatics Society, and the Forum InformatikerInnen für Frieden und Gesellschaftliche Verantwortung.

Cleidson R. B. de Souza received the BSc, MSc, and PhD degrees 1996, 1998, and 2005, respectively. He was at IBM Research Brazil. He is currently a Senior Researcher at the Vale Institute of Technology and an Associate Professor at the Federal University of Pará, Brazil. His main research interest is computer-supported cooperative work (CSCW) as applied to software engineering—that is, how software engineers work together to develop software systems. He uses research approaches such as ethnographic studies, surveys, and tool development and evaluation to understand and augment software developers' work. His work has appeared at top conferences and journals in both Software Engineering and CSCW, and he has helped to organize the series of Cooperative and Human Aspects of Software Engineering (CHASE) workshops. He is a member of the ACM, the Brazilian Computer Society and an affiliate member of the Brazilian Academy of Science.