Coordination and Learning in Global Software Development

Articulation Work in Distributed Cooperation of Small Companies

Dissertation

von

Alexander Boden

zur Erlangung des Doktorgrades Dr. rer. pol.

an der Fakultät III:
Wirtschaftswissenschaften,
Wirtschaftsinformatik und Wirtschaftsrecht

der Universität Siegen

Abstract

Software offshoring has been established as an important business strategy over the last decade. While research on such forms of Global Software Development (GSD) has mainly focused on the situation of large enterprises, small enterprises are increasingly engaging in offshoring, too. Representing the biggest share of the German software industry, small companies are known to be important innovators and market pioneers. They often regard their flexibility and customer-orientation as core competetive advantages. Unlike large corporations, their small size allows them to adopt software development approaches that are characterized by a high agility and flat hierachies. At the same time, their distinct strategies make it unlikely that they can simply adopt management strategies that were developed for larger companies.

Flexible development approaches like the ones preferred by small corporations have proven to be problematic in the context of offshoring, as their strong dependency on constant communication is strongly affected by the various barriers of international cooperation between companies. Cooperating closely over companies' borders in different time zones and in culturally diverse teams poses complex obstacles for flexible management approaches. It is still a matter of discussion in fields like Software Engineering and Computer Supported Cooperative Work how these obstacles can be tackled and how they affect companies in the long term. Hence, it is agreed that we need a more detailed understanding of distributed software development practices in order to come to feasable technological and organizational solutions.

This dissertation presents results from two ethnographically-informed case studies of software offshoring in small German enterprises. By adopting Anselm Strauss' concept of *articulation work*, we want to deepen the understanding of managing distributed software development in flexible, customer-oriented organizations. In doing so, we show how practices of coordinating inter-organizational software development are closely related to aspects of organizational learning in small enterprises. By means of interviews with developers and project managers from both parties of the cooperation, we do not only

take into account the multiple perspectives of the cooperation, but also include the socio-cultural background of international software devlopment projects into our analysis.

Based on an analysis of the management practices we found in the field, we present theoretical as well as practical implications for the management of software offshoring in small companies. Furthermore, we contribute implications for the design of supportive technology, and discuss the methodological issues that we encountered while doing field research in the field of distributed software development.

Acknowledgements

I am deeply grateful to my thesis advisor Volker Wulf for guiding my research activities at the intersection between Software Engineering and Computer Supported Cooperative Work. As a cultural anthropologist doing my PhD in the field of Information Systems, his support and advice was especially important and valuable for me.

Also, I cannot thank Bernhard Nett enough for his tutorage. Many of the ideas and insights of my research are based on the interesting discussions we had during the time I worked on my thesis.

Special thanks goes also to my former advisor Gunther Hirschfelder, who helped me in finding my way into the field of Information Systems. Without his support, I would hardly have taken this step.

I am especially grateful to Sebastian Draxler and Gunnar Stevens for their valuable feedback and input especially during the finishing phase of my study, as well as to Claudia Müller for her advice and co-authorship. I would also like to thank my other colleagues from the University of Siegen and Fraunhofer FIT who gave me feedback and inspiration in uncountable discussions.

Furthermore, I thank Gabriela Avram from the University of Limerick for the fruitful cooperation on various papers and other research activities. Thanks a million!

I also thank all the participants of my study, who welcomed me at their workplaces and allwed me to conduct interviews and observations, and who have chosen to remain anonymous in this dissertation.

Last but not least, I also feel deep gratitude towards my family and especially to my fiancée, who have always supported me morally through the years that it took to write this book; without their support, patience and encouragement, this dissertation would not have been possible.

Contents

List of Figures

List of Tables

1. Introduction

Software is more and more developed in distributed teams, be it in the context of outsourcing, offshoring, or forms of open source software development—a transition which has been labelled as *Distributed Software Development* or *Global Software Development* (GSD). While the former term highlights the distributed character of the cooperation between developers and teams working at different places for example in the context of outsourcing, the latter focuses on internationally distributed cooperation. Software offshoring is one of the various forms of Global Software Development. It describes business strategies of sourcing software development tasks to teams in foreign countries, usually for saving development costs, getting access to new markets, or streamlining business processes [113]. Even though it is hard to estimate the exact economic impact of software offshoring due to limited statistic data and varying definitions of the phenomenon in different studies [115, 152], it is commonly agreed that Global Software Development has been of growing importance for the software branch since the late 1990s [10, 164].

Software offshoring comes with high risks, due to the temporal, geographical, organizational and cultural barriers which have to be bridged for successful distributed team coordination. The discussion on the value of offshoring as a business strategy is controversial. On the one hand, there are many reports about failed projects of software offshoring in the industry (including dramatic examples such as an estimated 100 million dollar loss at AT&T due to a failed upgrade of an offshored CRM system [129]). Apart from operational risks, there are ethical and ecologic considerations, for example low working standards and a high need for travelling, which make it doubtful that offshoring of software development can be a sustainable and expedient business strategy for western firms [195]. On the other hand, offshoring is often discussed as being without alternative in times of global economic competition. At the same time, there are many reports about successful projects, indicating that it is possible to successfully deal with the challenges of distributed software development. As a result, the question that is discussed extensively is how such distributed projects can be successfully planned, organized, and upheld despite the numeruous obstacles and risks of globally distributed cooperation.

Global Software Development has emerged as a sub-discipline of Software Engineering that strives to understand and leverage the implications of developing software in distributed contexts. Early research in the field of GSD focused on supporting formal software development models in distributed settings, usually regarding offshoring as a "make-or-buy" decision to be taken by the management in the context of business-economic reasoning [126]. Today, GSD is increasingly understood as a dynamic process with complex implications for software development work due to the related organizational, temporal, spatial and cultural barriers of international cooperation (just to name a few). Research questions are centered on how practitioners organize and perform their work in distributed teams, how cooperation arrangements can be re-adjusted to emerging necessities, and how software developers can be supported in dealing with the challenges and problems of international cooperation.

Small and medium-sized enterprises (SMEs) have proven to be of high, albeit heterogeneous economic relevance for the German software industry. They constitute a large share of the enterprises in the software branch and are known to be important product innovators and market pioneers (see for example [13]). Software SMEs usually employ only a small number of developers; thus, it would be more precise to refer to them only as small companies. Research into small companies has shown that these often stick to self-composed mixtures of practices instead of straightforwardly adapting software engineering approaches [7]. At the same time, small companies often have only limited resources, which narrow their business opportunities considerably. As small companies are known to follow quite different software development approaches as compared to larger enterprises [78], it is questionable whether they can simply adapt approaches that have been identified in studies that are focused on large corporations [179, 40]. Hence, it is important to learn more about how small companies deal with the challenges of distributed software development in practice, what implications such strategies have for their economic success, and how strategies that have been developed for local teams or for larger forms of distributed software development can be adjusted to fit the needs of small enterprises [104, 205].

In order to answer these questions, it is agreed that more case studies into the work practices of software developers in distributed teams are needed [126]. As we do not know much about the practices of small companies for dealing with the challenges of offshoring, this dissertation wants to shed light on the question as to how these companies organize their work in the specific context of distributed software development, as well as how these practices are mediated and can be supported. In doing so, we will adopt the concept of

articulation work [219], which provides a fine grained view on coordination practices and has proven to be of high value in the field of Computer Supported Cooperative Work [200] (see chapter 2.2).

The dissertation is organized as follows: We will first develop a conceptional view on software offshoring in SMEs and present our methodology and case studies in the following part I of this book. Part II will then present our findings with regard to the research questions which will be developed in the next sections (see chapter 3 for an outline of the study). Part III concludes the dissertation with a summary as well as an analysis of our findings with regard to their implications for theory and practice, and for the design of supportive technologies.

Parts of this dissertation have already been published as conference or journal papers. See appendix I for an overview.

Part I.

Concept

2. Related Work

Offshore outsourcing (short: offshoring) of software development is an option of strategic business management which has been intensively discussed over the last years [134, 114, 34, 126, 10]. Offshoring is associated with high risks that are related to the temporal, spatial, organizational, and cultural barriers of such forms of international cooperation [171]. While many projects are known to have failed, risks have often been described as being manageable if the proposed offshoring models are followed—usually formal, waterfall-based models with a clear focus on documentation, architecture, and plans [59].

Outsourcing is a coinage of "outside", "resource", and "using" and basically denotes the delegation of formerly internal functions and structures to external service providers. The term subsumes a broad field of different sourcing strategies which are sometimes hard to discern from other forms of business relationships such as for example joint ventures [34].

Offshoring, on the other hand, is a term which highlights the location of the service provider. Offshoring basically means that the outsourcing provider is located in a different country "offshore"—usually in a low-wage country. Offshoring is sometimes distinguished from nearshoring and onshoring implying different levels of distance (and wage differences) between employer and provider. From a European perspective, India and Asia are usually seen as offshoring regions while Eastern Europe and Russia are regarded as nearshore locations [41]. As nearshoring has very similar qualities, the term is often subsumed under the broader term offshoring [194].

2.1. Global Software Development

This section provides an overview of the literature on offshoring in general, with a focus on software offshoring in SMEs. After a broad introduction into the history and background of offshoring in the industry, we will cover the perspective of Information Systems on the topic and then focus on the Software Engineering literature with regard to managing

distributed software development in the context of SMEs that have offshored parts of their software development.

2.1.1. Background

The recent and current interest in software offshoring is related to general trends of the economy, especially branches like the textile or automotive sectors, which became apparent in the second half of the 20th century. For example, the in-house production depth of automotive companies is usually less than 25% today; most of the parts and systems are built and assembled by external suppliers [246].

Offshoring as a significant part of a global economic strategy has been discussed since the 1960s, for example by Stopford and Wells [217]. The increasing impact of offshoring is mainly based on a change of business strategies towards a concentration on core competencies. The underlying assumption is that the depletion of international trade restrictions in combination with cheap means of international travelling, transport and communication offers access to new markets and makes it possible for companies to implement offshoring projects in almost all parts of their value chains [236]. In contrast to such economical arguments, there are also references to ethical and macroeconomic accounts in the discussion which question the sustainability (and validity) of offshoring strategies as short-sighted instruments to cut costs, leading to losses of jobs and competencies on a national level [195]. In the long term, it has been argued that the competition of companies would spread offshoring strategies further until the rising wages in offshore regions would deplete the benefits of offshoring [247].

The trend of software outsourcing (as a predecessor of offshoring) has been around since the 1990s. The reason for this relatively late development is that there were few specialized providers in the early days of computing, and companies began to set up their own departments. One of the earliest deals known was the outsourcing of the IT of the Blue Cross in Pennsylvania to the company Electronic Data Systems (EDS) in 1963. However, the symbolic breakthrough of IT outsourcing was the 1 billion dollar deal between IBM and Kodak in 1989. For the first time, a renowned company outsourced a whole business division (including 300 employees), an incident which some researchers see as the beginning of an era of "outsourcing mega-deals", as it raised the trust in the feasibility of large outsourcing projects, legitimized such strategies and encouraged other companies to follow. This has been labelled as the "Kodak effect" [190], the ignition spark of today's software outsourcing and offshoring strategies.

There are two reasons why IT departments and functions are increasingly in the focus of offshoring strategies today. On the one hand, software is increasingly understood as a service that can be provided from anywhere in the world due to negligible transfer costs (transaction cost view). On the other hand, IT is often seen as a pure cost-factor with very limited strategic relevance (resource-based view) [59]. The main driver for the increasing impact of IT offshoring, however, is the expectation to save costs. Although a shortage of skills on the domestic labor market as well as the chance for accessing foreign markets are often cited as the reasons for offshoring, cost savings are clearly dominating related surveys amongst companies [10]. Larry Ellison, CEO of Oracle, summarized the IT offshoring philosophy as follows: "Why should every automaker, publisher, or doctor's office have to be a tech company too, employing high-paid staff who spend all of their time fiddling around with computers?" (citation taken from [55]).

While IT offshoring includes a broad variety of different forms and arrangements (for example including models like "software as a service"), this dissertation focuses on a special form: the offshoring of parts of the software development work (that has formerly been done in-house) to a partner company in a low-wage country, resulting in a distributed form of two teams in different locations developing software together.

2.1.2. Offshoring of Information Systems

In Information Systems research, software offshoring is seen as a special form of Information Systems offshoring [136, 55, 114]. IS offshoring has been discussed from many perspectives in relation to economic, strategic, and cultural factors (just to name a few) [58]. The main focus of research has often been the identification of concepts and best practices for successful management of offshoring projects [135, 50], especially with regard to the multitude of different barriers of offshoring cooperation. For example, spatial and temporal distance is known to reduce the frequency of communication, while social and organizational distance can lead to misunderstandings and a lack of trust between the cooperating companies, thus leading to delays and requiring adequate management of offshoring cooperation [239].

The theoretic foundation of the discussion can be divided into three categories [58]:

1. *Economic approaches* like the transaction cost theory offer criteria for the planning and coordination of outsourcing/offshoring projects according to the relation of transaction and production costs [241, 240]. The main focus is the legal arrangement of offshoring contracts as part of the transaction costs. With regard to

software development, this is problematic as technical requirements can not always be fully anticipated, and as the time and effort of development and maintenance can hardly be estimated beforehand. This leads to problems regarding the legal organization of outsourcing/offshoring arrangements, since companies have to take into account hardly detectable opportunistic behavior on behalf of the service provider (principal-agent problem) [125], while at the same time avoiding an over-specification of the contract in order to leave room for unforeseen innovations [148]. For software offshoring arrangements, related considerations indicate the need to organize offshoring not in the form of mere service providing but rather in the form of a close cooperation like in a joint venture or intra-organizational form of cooperation, thus indicating the need to overcome the related barriers between the teams.

2. *Strategic approaches* like the resource-based theory focus on providing instructions for company development according to specific, non-imitable competencies of enterprises such as knowledge about processes or patents [49]. The main focus is placed on questions such as how to identify resources in companies that are suited for outsourcing/offshoring [4]. The theory suggests that companies should outsource parts that are either strategically unimportant (and keep the core resources in-house), or use outsourcing for freeing resources that can be used for innovation in more important areas [188]. With regard to software offshoring, the problems discussed include the questions as to how to distinguish strategically important parts from unimportant ones, and whether software development should be seen as an strategic asset, or a common service with little strategic importance in this regard. From this view, software offshoring only makes sense when software development work is seen as being distributable between different specialized roles: software architects and requirements engineers, who do the actual development work, and mere "coders", who "just" carry out the plans derived by these specialists (who would thus be easily interchangeable).

3. *Social/organizational approaches* like the power theory focus on relationships and dependencies between actors apart from economic reasoning. Such approaches take into account the influences of different interests, conflicts, and distribution of power in companies [156, 136]. The basic assumption is that stakeholders like an IT department continuously try to secure their power. As a loss of employees due to software offshoring could possibly decrease the influence of such a department, actors are expected to oppose related plans of the management in some cases. On

the other hand, powerful actors of small departments could see offshoring as a way to increase their influence in the company [60].

Other common approaches include the focus on reducing the risks of offshoring by developing adequate sourcing strategies, or on supporting offshoring decisions by providing different perspectives in relation to multiple theories at the same time [3, 61]. In general, it can be summarized that the approaches in the IS literature are very heterogeneous. At the same time, the focus is usually on problems of planning and organizing offshoring-related decisions of the management [58], for example by providing suggestions regarding the selection of business functions for selective offshoring [137].

Recently, the focus of the research has been shifted more towards seeing outsourcing and offshoring as a process, focusing on the (often unanticipated) changes that can be necessary in offshoring projects. As software markets are volatile and offshoring partners can not be changed arbitrarily, it is necessary to take into account long-term effects of sourcing strategies [140, 189], especially as these often involve a restructuring of the organization. In this regard, offshoring should not be seen as a mere "make-or-buy" decision but has to be considered as a dynamic process, which needs to be continuously re-adapted to emerging demands. This is especially important for software development, which on the one hand is hard to manage and plan ahead, and on the other hand can be regarded as an strategic "resource" in many cases [62]. For the situation of small companies, these aspects seem to be especially important, as the next sections will illustrate.

2.1.3. Software Offshoring in SMEs

The German software sector is dominated by small companies that offer not only software development but also services for its customization and appropriation support for their customers [205, 68, 78]. SMEs often attempt to compete on the market by addressing individual needs of particular customers, instead of making products for mass markets (as large companies often do). As a consequence, their business models include many different elements such as technology consultancy, help desk support, maintenance services etc. Thus, the seemingly trivial provision of custom-built solutions may involve very different and complex forms: SMEs may provide individual products by individualizing generic adaptable software to individual demands (tailoring, mass customizing, re-engineering, etc. [161]), but they may also develop or assemble modules for specific demands (similar to bricolage in Cultural Studies [103])

While large companies often are seen as the entrepreneurs of offshoring strategies and have been in the focus of scientific research so far, the pressure to adapt offshoring is growing for small and medium-sized enterprises, too [87, 104]. Due to their often limited financial possibilities, problems of planning and conducting offshoring projects are especially relevant for SMEs. Even for small projects, choosing the wrong partner can have severe consequences, as cancelling an offshoring contract (backsourcing) or changing the service provider can lead to high transaction costs.

In order to reduce such risks, the GSD literature often advocates careful planning and standardizing product development processes as is suggested for example by the CMM [187, 60, 126]. In this regard, development processes that divide labor into single, predetermined steps are meant to make software development predictable, transparent, and traceable while improving performance and productivity at the same time. However, predefining of and sticking to workflows is not always possible in dynamic, innovative environments [163]. Hence, SMEs usually do not attempt to certify their development processes [37, 78]. Instead, they stick to their own, self composed practices (which are often neither stable nor documented). Working in small groups allows for a very flexible handling of coordination mechanisms, division of labour, and hierarchies. The possibilities to quickly adapt to changing market demands, implement innovations in an ad-hoc way, and flexibly react to new wishes of the customer are counted amongst the most important competitive advantages of SMEs compared to larger competitors. Hence, such strategies seem to be promising for SMEs that focus on flexible adaptation, and not on highly formalized decision processes [237, 72]—thus implying a far greater proximity to agile methodologies than plan-based models of software development.

Furthermore, SMEs often do not follow one particular school but make use of practices which may belong to different software development methods that are adapted in such a manner that they fit into the specific domain of the company (if such methodologies are followed at all) [7]. From a Software Engineering point of view, such a mix of approaches is seen as problematic when SMEs adjust or only partially implement such methodologies in their daily practices [2]. On the other hand, if SMEs attempt to formalize their software development practices, the question arises how this affects their ability to adapt flexibly to emerging demands of their business, and how small companies can remain agile while engaging in offshoring [192, 193, 175]. Software development practice is facing severe challenges in this regard, as agile methods with their strong dependency on intensive communication and local teamwork are especially vulnerable to the disadvantages of distributed work [2, 165].

2.1.4. Flexibility and Learning as Core Competencies of software developing SMEs

If flexibility can be regarded a strategic resource of SMEs in the sense of the resource-based theory, then this flexibility has to be secured when such companies engage in offshoring projects [69]. Apart from the ability to coordinate software development efficiently across organizational borders, this implies that companies need to be able to continuously adjust their cooperation to emerging demands. When complex products have to be tailored to the individual needs of particular customers, companies need a high level of reactivity in order to dynamically adjust to emerging requirements.

Especially in innovative, context-sensitive fields like the software branch, such requirements become apparent only in the process of the development [182, 77, 248]. The reason is that software development can be seen as knowledge work, which is firm-specific and highly related to tacit knowledge [172, 121]. From such a view, knowledge can be shared only in the process of negotiating experiences amongst individuals or groups in the course of a common practice [229]. Successful examples of knowledge exchange like for example in the context of *communities of practice* are highly interwoven with practices of social interaction [238, 139], which should be taken into account for offshoring projects. Hence, SMEs do not just need efficient means of communication, but also trust and possibilities to continuously monitor and review their performance [140].

From this perspective, SMEs require ongoing learning and knowledge exchange in order to flexibly adapt their cooperation to the emerging technical and organizational necessities of their projects [177]. With regard to offshoring, it has to be stressed that this cooperation does not only include the (local) development teams but also the (foreign) partner company as well as the customer and other possible stakeholders of the project at hand.

Argyris et al. [9] have introduced the concept of double-loop learning in this context. According to this view, the ability for self-organized work does not only require an ongoing reflection of the performance (single-loop learning), but also a reflection of the underlying aims and assumptions (double-loop learning, see figure 2.1).

As actors have only a limited picture of the existing relations of their field of work, organizational learning and dynamic self-organization require ample knowledge exchange and a continuous reorganization of coordination mechanisms [1, 74]. In this regard, it seems to be important to learn more about how these aspects of software development work are affected by the circumstances of offshoring in practice.

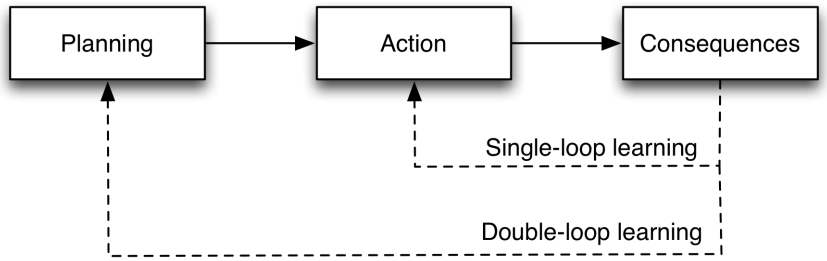

Figure 2.1.: Argyris et al.'s concept of single- and double-loop learning.

A deep understanding of the particular practices of SMEs is even more important as recent studies have shown that the needs of global software teams are fundamentally different from the ones of local teams in many aspects, and that research findings can not be transferred arbitrarily from local to trans-local contexts [54, 110].

2.2. Computer Supported Cooperative Work

In order to obtain a detailed understanding of the practices of software development in the context of offshoring in SMEs, this dissertation will adopt approaches from the field of Computer Supported Cooperative Work. With regard to our research agenda, CSCW offers a diversity of instruments and concepts that are useful for understanding software development as a complex, creative, and knowledge-intense practice—such as the notion of *articulation work* and *coordination mechanisms*, which will be introduced in the following sections.

2.2.1. Articulation Work

The concept of *articulation work* has been of high importance in the field of Computer Supported Cooperative Work. It has been initially introduced by the sociologist Anselm Strauss, also co-developer of the methodological approach of *Grounded Theory*, in the late 1980s [220, 219, 221]. Strauss contributed considerably to the school of Chicago sociology, and his work has been well received by organizational studies as well as technology research [222, 224]. His work on the role of perspectivity and process-oriented nature of *sociality* has offered important insights for the analysis of complex social systems.

Strauss developed his concept of articulation work in the context of medical work. Based on a long-term study on work in a hospital, Strauss identified different kinds of work, for

example *comfort work* (i.e. comforting patients), *technical work* (i.e. operating technical medical devices) and *articulation work* as a kind of meta- or supra-work that is needed to coordinate the different tasks and responsibilites of cooperating actors (i.e. nurses, doctors, technicians, relatives) in the course of treating patients (so called trajectories, or *arcs* of work) [219]. Articulation work was further developed by Strauss in several follow-up articles [220, 221], and also picked up as a fruitful perspective for understanding the intricacies of cooperative work arrangements in the foundation of the field of CSCW in the early 1990s [200].

According to the articulation work concept, cooperative work is inherently distributed in the sense that arrangements (like for example offshoring projects) contain arcs of work (like for example software projects) that include a division of labor with regard to both tasks and actors [201]. These interdependencies—including task-to-task, task-to-person, and person-to-person dependencies—constitute cooperative work, as they form the context in which (semi-autonomous) actors follow individual strategies, heuristics, perspectives, aims, and motives while interacting with their co-workers [196]. As these observations are centered at a highly granular understanding of work, its forms of distribution are analyzed on a much more basic level as compared to the higher levels of temporal and spatial isolation of globally cooperating teams.

Strauss has introduced articulation work as a form of meta- or supra-work that is needed to handle these inherent aspects of cooperative work. He characterizes articulation work as "the coordination of lines of work. This is accomplished by means of the interactional processes of working out and carrying through of work-related arrangements. Articulation varies in degree and duration, depending upon the degree to which arrangements are in place and operative" [222]. Articulation is needed mainly to coordinate the distribution of tasks amongst the actors: who does what, when, how, where, in which quality, until when etc. However, articulation work is more than just "coordination" in the sense that it also includes the negotiation of the inherent distribution of work. Apart from a top-down allocation of resources and responsibilities, articulation work includes the mediation of the different, individual activities with regard to their related arcs of work, for example corresponding individual interpretations and perspectives on work [201].

Articulation work has been developed further in the context of CSCW research towards the conceptions of *coordination mechanisms* [201], and recently the notion of *ordering systems* [202]. A coordination mechanism is defined as a

> "specific organizational construct, consisting of a coordinative protocol imprinted upon a distinct artifact, which, in the context of a certain cooperative

> work arrangement, stipulates and mediates the articulation of cooperative work so as to reduce the complexity of articulation work of that arrangement" [201].

Coordination mechanisms thus are at the core of an artifact-centered view on coordinative protocols, focusing on questions as to how to support articulation work by providing actors with adequate (computational) coordination mechanisms. As a means to design such systems, Schmidt and Simone have developed ARIADNE as a generic, semantic notation for building malleable and interoperational coordination mechanisms [206]. With regard to software development work, an example is discussed in form of a bug tracking tool that can be used to classify and describe bugs, then assigning them to developers with specific deadlines and priority levels. While such coordination mechanisms are usually thought to work on the level of the cooperation of a particular team (which can nonetheless be distributed), ordering systems take a step back and focus on coordinative work happening outside of a team. By taking into account boundary objects between different cooperating communities, ordering systems shift the attention towards more open, evolving artifacts in the flow of complex work trajectories in distributed, fragmented fields of work [202].

With regard to software offshoring in SMEs, articulation work and the related concepts like coordination mechanisms can be seen as analytic lens for understanding the work that is needed to coordinate—or rather: articulate—software development projects against the background of the specific needs of SMEs. The broad understanding of cooperative work in this regard makes it possible to analyze the role of informal re-adjustments of projects, which are likely to occur often in software SMEs due to their flexible approaches towards product development for specific market niches (see above). In this regard, we will also look at the role of articulation work for organizational learning, i.e. how single- and double loop learning is realized in small firms, and how the related practices of learning are articulated. This is especially important as software projects need to be continuously re-adjusted to emerging demands, and as articulation work is very likely to occur intensively when unexpected occurrences lead to unforeseen problems and contingencies that have to be solved in order to proceed with the project (thus offering important relations to organizational learning).

2.2.2. Articulation Work in Distributed Software Projects

In local projects with low complexity, coordination can often be upheld without much efforts in the course of the daily social interaction [165, 108]. An important concept in this

regard is *awareness* [102], which describes an implicit form of coordination based on "an understanding of the activities of others, which provides a context" for their own activities [67]. Awareness is often divided into different categories like informal awareness, social awareness or workplace awareness [98], and can reduce the need for explicit articulation work, as actors monitor each others work progress sub-consciously and react to changing states of the project without the need for explicit discussions. Furthermore, many issues can be handled informally, as upcoming questions can be worked out in ad-hoc discussions and plans can be readjusted easily. However, if projects are getting more complex, for example because new developers are involved that are working at an offshore site in a different country, these implicit forms of coordination may not be sufficient anymore. With growing distance, articulation work can become more complex as communication frequency and awareness are likely to decrease, thus creating barriers between the teams which can lead to delays, misunderstandings or even project breakdowns [97, 106, 251].

At the same time, articulation work can be a kind of "invisible work" in the sense that actors are sometimes not aware of the articulation work they are performing, or at least often do not regard it as being part of their work [211]. Hence, if SMEs decide to engage in offshoring, they can sometimes be surprised by the enormous complexity that is introduced by the change, which can exceed the need to talk in a foreign language and deal with time differences considerably [88]—especially if the company previously has followed a very ad-hoc, informal way to organize software development. Hence, companies need to find organizational and often also technical means for dealing with the challenge of coordinating their work in distributed contexts [95, 47, 128].

A common approach for reducing the need for articulation work is optimizing the division of labor and the corresponding workflows. A huge part of the research on GSD is centered on developing corresponding management strategies, development models, or software tools [109, 180]. From an articulation work perspective, formal constructs like plans or workflow models can be seen as coordination mechanisms that are meant to reduce complexity and make articulation work easier. While they can reduce the need for constant coordination, one characteristic of articulation work is that it can never be fully substituted. "The important thing about articulation work is, that it is invisible to rationalized models of work" [209]. As it is not possible to anticipate all possible eventualities and contingencies, formal constructs can never be complete. Hence, their use in practice always requires articulation work itself [89]. Even when deterministic plans reach a relatively high level of accuracy, the necessity remains to synchronize the different fragments of the divided labor [170, 177]—a problem which can not be solved by plans. Hence, while

plans can be seen as resources for cooperative work, they can never predefine it fully and will always require articulation work in practice—an understanding which Lucy Suchman has labelled as "situated action" [226]. Schmidt recently has pointed out that plans still play an important role for cooperative work [199]. Suchman's account of situated action was meant as criticism against cognitive science (which basically assumed that rational action would be *determined* by plans). However, while some action is definately ad-hoc in the sense that it is spontaneous and improvised, other action is planned in the sense that there is an obligation to follow certain predefined steps in a particular sequence, at certain times etc. At the same time, plans do play an important role for action in the sense of normative constructs providing guidelines for what is correct, and what is not. Of course, action is still situated in so far that plans and rules have to be applied, and that actors can choose to alter, abandon or ignore them in certain situations—thus indicating a need for articulation work that has to be supported.

With regard to supporting coordination mechanisms in distributed software teams, research has shown that the requirements for coordinating software development can be volatile and exceed the borders of teams [44]—especially in the case of small companies with their flexible approaches towards software development. This narrows the possibilites of adopting formal development models and indicates a need for flexible coordination mechanisms that can be easily adapted to emerging necessities (see chapter 2.1.3). Related approaches are often based on agile methodologies which are increasingly tested in the context of distributed software projects [228, 14]. While they seem to be generally well suited for the needs of SMEs, their application requires ample articulation work and thus can be problematic in distributed contexts. Hence, the literature has suggested approaches to support the necessary articulation work of distributed actors by providing adequate means of communication media, often embedding them into the development tools [90, 54, 167]. At the same time, attempts of using such concepts in the context of offshoring often attempt to formalize the related articulation work in order to make them more predictable and less dependent on frequent communication [194]. As this focus on planning seems to contradict the basic assumptions on which agile methodologies are based [17], it is interesting to investigate the possible outcome of such strategies [175, 168]. This is especially important as a detailed understanding of the organizational issues of distributed software development is important for guiding the design of practicable solutions for companies that engage in this type of international cooperation [95]. Right now, we know too little about the needs and practices of SMEs in order to assess the applicability of existing approaches in this context (like for example the conception

of Continuous Coordination that tries to bring together formal and informal aspects of coordination [177]).

In addition to such approaches, there are also attempts to develop novel assistant systems for software development teams, for example for raising awareness of possible breakdowns by means of statistical analysis of the development IS. Projects in this area are often based on data mining or network analysis methods to investigate information flows in companies. These findings are then compared for example with technical dependencies in the source code [177] in order to make predictions about possible information barriers. Possible issues can then be vizualized for the team members to raise awareness and prevent breakdowns before they occur [191]. While such approaches are able to support the (implicit) coordination of software teams [150], their application domain is usually rather narrow. The problem is, that they are usually centered on a couple of indicators that are strongly dependent on the available data from the development IS (like for example up-to-date bug tracking system or an actively used mailing list). However, recent research has shown that Information Systems often do not represent the current state of affairs in software teams very well [8]. Hence, such tools have to be applied carefully in order to be successful. At the same time, they cannot simply be deployed in software companies but need to be tailored to the specific practices of the teams they are supposed to support.

For our study, it will thus be important to learn how coordination mechanisms and plans are used by the practitioners in terms of how they are developed, negotiated, implemented, and modified in practice (also in the sense of organizational learning)—and also how distributed software development teams in small companies can be supported in this endeavor.

3. Study Outline

In the following section, we will present our research questions and put them into perspective. Then, we will describe the methodology of our study as well as the case studies that we have conducted.

3.1. Perspective

This dissertation aims at providing a detailed perspective on software offshoring in small software companies with regard to the related challenges of distributed work, and the strategies that small enterprises have developed for dealing with them. As agility and customer orientation have been identified as the main competetive advantages of small software enterprises, it is especially important to focus on the question as to how such small and volatile teams deal with the challenges of flexibly *coordinating* work in distributed cooperation. Furthermore, it is important to understand how small companies *learn* in the context of offshore software development in terms of quickly reacting to changing customer demands (operational learning) and developing the cooperation with regard to the emerging necessities of distributed firm cooperation (strategic learning). Both aspects (coordination and learning) are closely interrelated, but their connection and interdependencies have hardly been studied so far in the context of offshoring [126, 22]. We have identified the conception of articulation work as a promising framework for the analysis of these aspects, especially with regard to the question how distributed software teams use (and develop) coordination mechanisms in practice to deal with the challenges of their distributed cooperation.

Researching articulation work requires a sound analysis of the socio-political and cultural contexts of the practices to be investigated [227]. Hence, we will not only center on the normative side of offshoring (i.e. on the question as to how teams should organize their work in order to be successful from a management perspective), but also on the practical side of managing and doing distributed software development on the shop floor. In this regard, routinized actions have to be distinguished from situated ones, and contextual

norms have to be identified without violating them [200]. We will also have to take into account the particular social and cultural context of offshoring cooperations with regard to the roles of aspects like trust, social capital, and culture [52]. In this respect, culture has to be seen as an interpretive concept that includes socially embedded sensemaking and negotiation processes in different layers [85]. In order to address these aspects appropriately, we will take an ethnographically-informed approach in our study.

A close investigation of the articulation work in the context of software offshoring in SMEs will allow us to contribute to the discussion on coordination and learning in the context of offshoring (with regard to the applicability of management conceptions like Continuous Coordination [177] or agile development methods [228] for small companies) as well as to the discussion on the role and nature of articulation work and coordination mechanisms in this context [199]. Furthermore, a detailed understanding of the practices of managing distributed software development will allow us to provide recommendations for practitioners [155], as well as to derive implications for the design of technologies that support and mediate the articulation work of small software teams in practice (especially with regard to the different approaches that we identified in section 2.2.2).

In the following sections, we will first provide details on the methodology of our study as well as on the two cases of software offshoring to Russia which were at its focus: the companies Alpha and Beta.

Part II of the dissertation will then center on answering the questions outlined above. The five chapters of part II have already been published and resemble the accepted versions of the related journal or conference papers (see appendix I).

- Chapter 4 will provide a methodological reflection of our work. Conducting research in the field of distributed software development turned out to be subject to challenges similar to the ones the software developers had to face. Hence, we will discuss how we tried to meet those challenges in our study.

The two subsequent chapters will then provide a comparative analysis of the coordination and learning practices we found in the two case studies:

- Chapter 5 will focus on the actual practices of coordinating distributed software development in the two companies. It will be centered on the question as to how the actors in the two cases organized their software development work, and how the companies tried to warrant their operative agility in the context of their offshoring projects.

- Chapter 6 will discuss how the different approaches chosen by the companies affected their ability of organizational learning in the long term. For doing so, we will investigate the relation between single- and double-loop learning and articulation work in the two cases with regard to how the companies were able to strategically adapt their cooperation to emerging demands.

The last two chapters of part II will analyze the socio-cultural context of the practices of distributed coordination and learning that we have found in the field:

- Chapter 7 will focus on the roles that social capital and trust played for the decline of the cooperation in one case (Beta). By revisiting an offshorig failure story, the chapter will highlight how the organization of the cooperation co-evolved with social capital and trust during its different stages.
- Chapter 8 will then center on the impact of cultural aspects on the cooperation of company Alpha. By examining the constant re-negotiation of the socio-cultural context of coordination and learning in distributed teams, we will highlight the complexity of the conception of culture and draw conclusions for related management approaches.

Part III will bring the different perspectives together and analyze them with regard to their theoretical and practical implications (see chapter 9).

3.2. Methodology

Our study on offshoring in SMEs started in early 2006 and consisted of several stages.

First, we conducted an interview study with 15 managers and developers from 10 SMEs of the German software branch in order to get a general overview on the different attitudes towards offshoring, i.e. the related assumptions, experiences, and strategies. In addition, we also conducted expert interviews with a division manager from a large company, with a manager from an Eastern European offshoring provider, and with a consultant from a business association in order to get a professional assessment of the phenomenon we were interested in. Most interviews were held at the respective companies (in some cases by phone), and most of them followed the same interview guide-line (which evolved during the study as we learned more about the subject). All interviews were recorded and transcribed, except for one case where we had to rely on handwritten notes because the interview partner did not allow us to record the interview.

Based on this sample, we chose two cases which displayed very different attitudes towards the organization of their distributed development for a comparative study (the two companies will be described in greater detail below). Both companies were visited for ethnographically oriented on-site observations over a period of three weeks each. During the visits, we stayed at the companies during their usual working hours and observed how the developers were working, especially with regard to the cooperation with the offshore teams. When something interesting happened, we took fieldnotes and also conducted ethnographic interviews (see for example [91]) with most of the employees about their views and experiences, asking them to explain what they were doing with regard to the cooperation with their offshore partners, how they organized the cooperation, and why they took a particular decision. Observations and interviews were documented in the form of field notes which were taken directly or soon after, on the same day. We also asked for access to the development IS, chatlogs, and to artifacts like specification documents, and included them in our analysis.

In order to get a better view on the perspective of the offshore team, we also visited the Russian partner company of one of the two companies for one week. We were able to accompany the visit of two German project managers to the team in Tomsk, Siberia. At the offshore site, we followed a similar methodology as at the German companies. During our stay, we were also able to conduct interviews with managers of three other companies in the Tomsk region regarding their experiences with offshoring cooperations. We also planned to visit the partner company of the other company, but unfortunately this was not possible due to an unexpected termination of the whole cooperation. In order to compensate, we arranged an interview with the manager of the Russian partner company (see chapter 7 for details).

All in all, we conducted 33 semi-structured interviews and 38 days of on-site observations (supplemented by ethnographic interviews). See table 3.1 for an overview.

For the analysis, we oriented on Strauss' and Corbins Grounded Theory approach [223]. Material was coded in an iterative process that was conducted directly after each field contact (e.g. after the transcription) and involved several steps. First, we tagged the transcripts with regard to different codes that we identified in relation to the aspects we were interested in. For example, we used codes like "Asking for help", "Customer contact", and "Knowledge exchange" to mark positions in the material that covered these aspects. Then, these codes were refined and assembled into categories (like "coordination" and "learning"), and we tried to connect them to each other in order to build a theory of

Period	Method	Field
02/2006 - 01/2007	Semi-structured interviews	15 developers and managers, 1 Eastern European offshoring provider, 1 consultant, 1 large company
07/2006, 5 days	On-site observation, ethnographic interviews	Company Alpha, Bonn
12/2006, 2 days	On-site observation, ethnographic interviews	Company Alpha, Bonn
01/2007, 5 days	On-site observation, ethnographic interviews	Company Beta, Berlin
02/2007, 5 days	On-site observation, ethnographic interviews	Parter company of Alpha as well as three offshoring providers in Tomsk, Siberia
07/2007, 5 days	On-site observation, ethnographic interviews	Company Beta, Berlin
07/2007	Semi-structured interview	Partner company of Beta, Saint Petersburg (telephone interview)
01/2008, 2 days	On-site observation, ethnographic interviews	Company Alpha, Bonn
03/2008, 2 days	On-site observation, ethnographic interviews	Company Beta, Berlin
10/2008	Kick-off meeting	Company Alpha, Bonn
11/2008-02/2009, 11 days	Semi-structured interviews, on-site observations	11 employees of Alpha in Berlin and Tomsk
06/2009	Workshop	Company Alpha, Bonn
07/2009, 1 day	On-site observation, ethnographic interviews	Company Alpha, Bonn

Table 3.1.: Overview on the research activities.

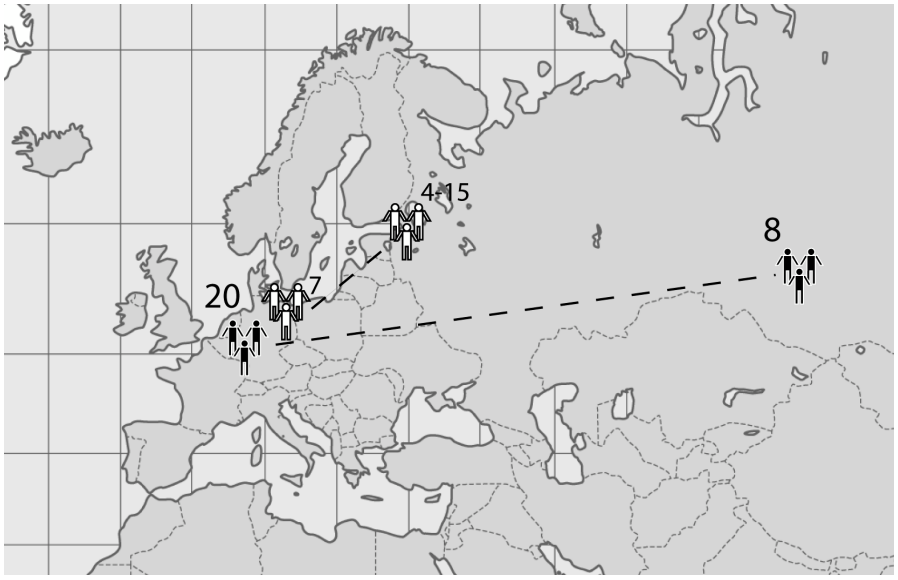

Figure 3.1.: The two case studies Alpha (Bonn-Tomsk, black) and Beta (Berlin-Saint Petersburg, white). The numbers indicate the team sizes during our research (worldmap by I.K. Sankakukei, http://english.freemap.jp, distributed under a CC-BY 3.0 license).

offshoring expecations from the material. During the following research steps, we further developed and refined our concepts on the basis of the new material.

In 2008, we changed our methodological focus from explorative ethnography towards a more design-oriented action research approach of *Business Ethnography* [214, 159]. This involved conducting exhaustive semi-structured interviews with most of the employees of company Alpha as well as further on-site observations. After the analysis of the material with regard to the multi-perspectivity of the actors, we discussed our results with the employees with regard to possible implications for their organizational practice in the context of a shared workshop. Details on this part of our study and the related methodologcial implications will be discussed in the chapter 4 below.

3.3. Case Studies

In the next sections, we will provide an overview on the two case studies that formed the basis of our research: the companies Alpha and Beta (see figure 3.1 and table 3.2 for an overview).

	Company Alpha	**Company Beta**
Field	Statistics and documentation products and services for archives and museums	Business Process Modeling tools and services
Headquarters	Bonn	Bonn (development team in Berlin)
Size of development team (Germany)	20	7
Size of offshore team	4-8	4-15 (varied during research)
Offshoring experience	ca. 1998 -	2002 - 2008

Table 3.2.: Overview on the two case studies.

3.3.1. Company Alpha

Company Alpha offers products and services in the field of statistics and documentation. Customers of the company are mainly public institutions like archives and museums. Generally, the company is interested in long-term cooperations with their customers, aiming rather at negotiating service and consultancy contracts than selling software products. As the working procedures of German archives and museums have a distinct (and complex) tradition and can be quite different between institutions in Germany, customers usually need a high level of tailoring to fit the desired software solutions into their working domains. The company maintains two core products (one for archives, and one for museums), which are each in the responsibility of project managers who sometimes branch the code in case of bigger projects. The company also has several small projects which are usually in the responsibility of the German developers. The headquarters is in Bonn, where 20 employees work.

The company has been cooperating with developers in Russia since the mid-1990s. Basis for the cooperation was the internship of a talented Russian developer from Tomsk, Siberia, in the German company. As Alpha wanted to hire developers in this period but was unable to pay the high wages demanded by the German job market at that time, the project manager saw a potential and hired him. He then asked him to find further Russian developers, and established a branch in Tomsk with the help of another German company led by a former manager of a large automotive enterprise. The reason was that the German manager feared the legal liabilities of founding a Russian company. Getting help from the experienced manager allowed him to run his branch simply as a cost center. Later, this set-up was changed in order to reduce costs but the former intern

still works for the company and has (temporarily) migrated to Germany in the meantime (see chapter 8 for details).

The first project was an update of an older software product of the company with adaptations to new frameworks in C++. Despite some initial problems with this project, which took much longer than initially planned, the cooperation was extended also to smaller projects which are usually led by German project managers. The only exception to this rule is the case of the product which was initially re-engineered. This project is now led by a Russian project manager, who reports directly to the German manager and owner of the company. The size of the Russian team varied during our research between 4 and 8 developers.

3.3.2. Company Beta

Company Beta develops a standard software product for Business Process Modeling (BPM). This product is used and advertised by a group of business consultants, who are in close contact with the company and use to send bug reports and feature requests. The company follows a static development cycle, with fixed release dates for new versions every year. There are also a number of smaller projects for customers who need features which are not suitable for the main branch and which are implemented in form of Visual Basic plugins by the German company. The headquarters of the company is in Bonn, but the development team is situated in Berlin and has 7 employees. Beta is part of a holding that is based in Düsseldorf.

The actual development of the standard software product is performed by a Russian team which has been cooperating with the German company since 2002. Basis for this cooperation were the high development costs of the German company at that time. This motivated the German manager to look for possible saving potentials by means of outsourcing parts of the software development work to low wage countries. Based on personal contacts the company hired a Russian developer and instructed him to look for colleagues in his home region in Saint Petersburg. After hiring a team of four Russian developers, the company invited them over to Germany for a couple of months, in order to make them them familiar with the codebase.

Afterwards, the Russian team took over the software development from the German company (apart from the smaller projects decribed above, which are still carried out by the German team). Later, the Russian team size was increased up to about 20 developers, then reduced again to about 4 due to financial and organizational problems

with the coordination of the large offshore team. In early 2008, the company terminated the cooperation with the Russian team. This decision was based on the high development costs and rising wages in Saint Petersburg, but also based on the fact that the holding was not satisfied with the performance of the cooperation. The exact reasons and implications of this offshoring failure story are described in chapter 7.

Part II.

Findings

4. Conducting Business Ethnography in Global Software Development Projects of Small German Enterprises[1]

Context: Studying work practices in the context of Global Software Development (GSD) projects entails multiple opportunities and challenges for the researchers. Understanding and tackling these challenges requires a careful and rigor application of research methods.

Objective: We want to contribute to the understanding of the challenges of studying GSD by reflecting on several obstacles we had to deal with when conducting ethnographically-informed research on offshoring in German small to medium-sized enterprises.

Method: The material for this paper is based on reflections and field notes from two research projects: an exploratory ethnographic field study, and a study that was framed as a Business Ethnography. For the analysis, we took a Grounded Theory-oriented coding and analysis approach in order to identify issues and challenges documented in our research notes.

Results: We introduce the concept of Business Ethnography and discuss our experiences of adapting and implementing this action research concept for our study. We identify and discuss three primary issues: understanding complex global work practices from a local perspective, adapting to changing interests of the participants, and dealing with micro-political frictions between the cooperating sites.

Conclusions: We identify common interests between the researcher and the companies as a challenge and chance for studies on offshoring. Building on our experiences from the field, we argue for an active conceptualization of struggles and conflicts in the field as well as for extending the role of the ethnographer to that of a learning mediator.

[1]This chapter has been published as an article in the Journal on *Information and Software Technology* 2011 [28]. Reprinted from Information and Software Technology, 53/9, Alexander Boden, Claudia Müller, Bernhard Nett, Conducting a Business Ethnography in Global Software Development projects of small German enterprises, pp. 1012-1021, Copyright (2011), with permission from Elsevier. See appendix I.

4.1. Introduction

Globalization has led to an increasing spread of geographically distributed software development teams [105]. Cooperating over barriers of different organizations, nations, languages, time-zones and cultures is a multifaceted field of partially inter-related problems, including communication, knowledge exchange, and the coordination of international work groups [126].

Studies in the field of Global Software Development (GSD) face a variety of methodological problems, as researchers entering this complex field face similar challenges to those faced by the practitioners themselves. Understanding and tackling the challenges of researching in distributed software development work requires novel research approaches as well as a careful application of data-gathering and analysis methods. Thus, reflecting on alternative methodological approaches and their applicability to a distributed context is an important topic for the growing GSD community.

In this paper, we discuss several challenges we had to deal with while conducting research in the context of offshoring in small to medium-sized enterprises (SMEs) of the German software industry. Our work is based on the methodological experiences we gained from a research project that used two different research designs to address various problems within the context of offshored software development work in SMEs [26, 29, 31, 32]. The first part of the project was an exploratory, ethnographically-informed field study on coordination practices in offshore software development in SMEs. The second part was conceptualized as a *Business Ethnography*[2] [214] and aimed at supporting flexible coordination practices and organizational learning in distributed software teams. This second study is going to be the focus of our paper, while we are also providing a brief discussion of the first study in order to draw some comparisons. It has to be noted that we applied the Business Ethnography research cycle only partially in our project. However, our analysis showed that even our incomplete implementation had relevant impact on the field site we were conducting our research in. By analyzing the related challenges and opportunities, we argue for actively conceptualizing struggles and conflicts in the field, as well as for extending the role of the researcher to that of a learning process mediator.

The paper is organized as follows: Section 4.2 gives an overview over related work with respect to literature on research methodology and introduces the conception of Business Ethnography. Section 4.3 describes the aims of our study and the methods we applied in relation to our research agenda. Section 4.4 describes how we implremented our Business

[2]See http://www.business-ethnography.org.

Ethnography in practice, while section 4.5 identifies challenges that we encountered during our research, which are then analyzed with regard to the related literature in section 4.6. Section 4.7 concludes the paper.

4.2. Related Work

4.2.1. Qualitative Research in Software Engineering

Over the last decades, qualitative methods focusing on understanding work practices have received much attention in Software Engineering. Practices in relation to the use of software applications are still a major research area, as are the development of software and the deployment of related methods by development teams. Research focusing on software development teams has revealed that quantitative methods have their own limitations as the development work is mostly framed by specific situational and organizational contingencies [63]. In this regard, qualitative methods are employed for gaining highly detailed insights into the contexts in which software development work takes place, which would not be accessible, for example, by applying quantitative data mining approaches [8].

The adoption of qualitative methods in Software Engineering has been largely influenced by related disciplines such as Human Computer Interaction (HCI), Computer Supported Cooperative Work (CSCW), and Participatory Design (PD). Using qualitative methods often involves following an ethnographic research methodology, entailing (participant) observation and ethnographic interviews with the aim of understanding what activities mean to the people who do them [185]. In this context, some researchers opted for ethnomethodological approaches, as well as for sociological accounts of ethnography aiming at understanding social structures in the workplace [204].

Software Engineering research based on (or informed by) ethnography and ethnomethodology has the potential to offer an understanding of software development practices "from within". Such research is interested in how project team members order their work using their own methods, "that is, their mundane processes of interaction and action" [184]. However, understanding development practices is only one of the aims of qualitative research in Software Engineering. This is complemented by efforts to actually improve poor or ineffective practices, for example by designing and introducing new information technology systems into an organization [33]. Qualitative research can be used in this respect for informing design with regard to the organizational context as well as gaining knowledge about the users' work practices that are to be supported by the system [176].

However, the aim to support organizations (instead of "just" studying them) has many implications for the choice of the research approach, as well as the application of research methods [65]. As traditional ethnographic research usually wants to avoid having any influence on the practices to be investigated, technology development projects are often deploying action research approaches that aim explicitly to construct a cooperative workspace encompassing the researchers and the development team. The underlying rationale is that one can understand social processes best by introducing changes and observing the effects of these changes [15]. While the descriptive nature of ethnography has been criticized for stressing the status quo instead of leading to new ideas in this regard, findings from the Participatory Design community have shown that the deep understanding that comes from ethnographically informed action research can also help to find and develop innovative tools [185].

Using ethnographic methods in organizational contexts entails many challenges, like negotiating access and getting immersed in the field under study, which are inherently linked to the relationship between the researchers and the development and management teams [63]. When conducting ethnographic research in the context of Global Software Engineering, additional research challenges exist, which are related to the temporal, spatial, cultural and organizational barriers between the cooperating sites [97]. Dealing with such barriers while conducting research is everything but trivial, and the researchers can be subjected to similar challenges as the practitioners themselves. Known problems of conducting research on GSD teams include getting access to the research field in the first place [5] as well as dealing with constraints such as organizational hierarchies and conflicting interests that may hinder a deeper immersion of the researchers in the environment [245]. With regard to action research approaches in GSD, further considerations have to be given to the implementation of changes across organizational borders (i.e. in both of the cooperating companies), or even in the same company across different projects [184]. This issue is extremely complex, as the success or failure of recommended changes does not only rely on the validity of the research findings, but also on the framing socio-political factors at work in the respective organizations [64].

As the implications and challenges of studying GSD projects have hardly been discussed in Software Engineering literature so far [184], we want to present our experiences with conducting research in this field in our paper. In doing so, we will focus on our experiences with conducting ethnographically-informed action research in the context of software offshoring in a German SME. Our research project was oriented on the conception of Business Ethnography, which will be introduced in the next subsection.

4.2.2. The Concept of Business Ethnography

Business Ethnography is based on the Intergrated Organization and Technology Development (OTD) approach that was introduced by Wulf and Rohde in the 1990s [249]. On a theoretical level, Business Ethnography is based on conceptions of *American Pragmatism*, especially on the work of the logician Peirce [169], which implies that what appears to be fully consistent from one perspective can look very different from another [203]. For instance, the way a software developer understands a requirement may not necessarily be identical with how a user or stakeholder understands it [74]. In a similar fashion, the way a project manager understands a task or aim may not match the views of his colleagues, especially in the case of GSD projects where team members are working in different cultural and organizational contexts. Business Ethnography (BE) tries to overcome such problems by supporting reflexivity in making project decisions [158]. As aspects of organizational development are also at the center of the related reflections, Business Ethnography is also applicable to GSD studies that are not aimed at developing new technologies, but at analyzing and improving practices of distributed software development work.

In practice, Business Ethnography follows a typical Action Research cycle, consisting of the following stages: research, analysis, and feedback workshops (see figure 4.1). Formally, Business Ethnography involves the initiation by the researchers of a joint project with the practitioners, and a related anticipation of the mutual aims. For example, a shared project could be a development activity of a software company, whereas the anticipation would be that of the software to be developed. Or, as in our case, the project could be focused on learning to improve the GSD cooperation of a company, with the aim of finding better ways of cooperating across temporal, spatial and organizational barriers (this case will be described in greater detail later).

During the research phase, the researchers attempt to identify the different perspectives of the project participants on the aims of the joint project by applying a focused form of ethnography [127] to the related sense making processes, deconstructing them into the multiple individual contributions. By conducting interviews with all participants, the researchers attempt to capture and illustrate the practical implications (i.e. anticipations, expectations, fears, etc.) of the shared intentional frame, which may impact the joint project. In order to get a complete picture, the researchers deploying Business Ethnography rely on intensive, individual interviews with all participants involved in the project. Other ethnographic methods like on-site observation can be used to complement

the interviews, helping the researchers to get a rich understanding of the project context that is related to the different perspectives, intentions and expectations of the actors.

In the analysis phase of Business Ethnography, the researchers actively look for a plurality of views and analyze their meanings and implications for the joint project. The desired result of the analysis phase is a refined, multi-faceted description of the field, including the different perspectives (including hopes, concerns etc.) of the participants. Since intentions are expressions of self-determination and need to be respected, Business Ethnography is not trying to evaluate the intentions of the participants from some absolute perspective; instead, it studies the complex inter-relation between the intentions and expectations of the project participants in order to allow for better informed decision making. In this regard, the researchers also have to take into account their own roles in the project, and constantly reflect on them during the research.

In the feedback phase, the results of the analysis are shared in a common workshop, confronting practitioners with "their" own perceptions (as well as with the related synergies and conflicts) from a distanced, alienated ("*verfremdet*") point of view [214]. Taking this presentation as basis for decision making, the partners may re-appropriate their project as a more concrete and situated relation between the different intentions and expectations, unveiling implicit contradictions and differing perspectives on the project procedures, as well as the interpretations and assumptions of the participants. Thus, project actors are enabled to reflect upon their own decisions from a wider perspective [77]. By doing so, Business Ethnography, as an action research process [9], supports the ability for self-organization and enriches the expertise of the project participants. The results of the workshop, as well as the agreed actions can become the subject of another Business Ethnography cycle, in order to validate and to further improve the practical findings.

4.3. Research Project: Articulation Work in Offshoring of small to medium-sized Software Companies

This section provides an overview of the research design approaches used in the two studies we conducted for this research project. It also describes our methodology for comparing the two research approaches (section 4.3.2), as well as the organization we focused on in our research project: company Alpha (section 4.3.3). A detailed discussion of how the Business Ethnography was conducted in practice will be presented in section 4.4.

	First Study	**Second Study**
Aims	Understanding coordination practices in the context of offshore software development in SME	Supporting coordination and organizational learning in distributed teams
Research Design	Ethnographically-informed field studies	Business Ethnography
Methodology	- 15 semi-structured interviews in ten German SMEs - 5 weeks of on-site observations in 2 German SMEs, as well as one Russian partner company in Tomsk, Siberia. - Grounded Theory analysis	- 11 open interviews with almost all employees of company Alpha. - 2 weeks of on-site observations at the company - Grounded Theory-oriented analysis - 1-day workshop at the company

Table 4.1.: Details on the two studies in our research project.

4.3.1. Aims and Research Design

Our research project included of two studies following different research designs on different problems within the context of software offshoring in German SMEs. The first one was an exploratory ethnographically-informed field-study in several German SMEs. The second study aimed at studying and supporting articulation work and organizational learning in distributed software development teams, and was conducted in the form of a Business Ethnography in company "Alpha", a German software SME (see table 4.1 for a brief overview). The focus of this paper will be on the second study. However, we will briefly discuss the first study, which formed the basis of our Business Ethnography, in order to draw comparisons between our experiences with the two approaches.

In both studies, the first author (who has a background in Cultural Anthropology) was invoved in field research (observations, interviews). The data analysis, as well as the preparation and execution of the workshop were undertaken collaboratively be all authors, with the support of other members of the research group and the involvement of a number of student volunteers.

The initial aim of our research project was to understand coordination practices in SMEs belonging to the German software sector, in the context of offshore software development. As we wanted to learn how SMEs try to secure their agility in Global Software Development, we decided to conduct an exploratory ethnographically-informed field study.

First, we conducted semi-structured interviews with fifteen managers and developers of ten different German SMEs. From the sample, two companies were chosen as sites for a deeper analysis of the work practices; in these, we conducted on-site observation. We spent two weeks in each of the two German SMEs. A third on-site observation period was conducted over one week at the Russian partner company of company Alpha in Tomsk, Siberia.

Our analysis showed that the SMEs of our sample used a broad variety of different coordination tools and artifacts in order to coordinate their distributed work. While formal tools (like bug-tracking applications and project plans) played an important role for the development work, it became clear that their use happened in the context of complex articulation work practices [219], which required time-consuming chat sessions and expensive personal visits. The related limitations (in combination with social and cultural issues) led to frequent misunderstandings and limited the ability of the companies to adjust their working arrangements in case of emerging problems [29].

Our first study was conducted in the context of a bigger research project funded by the Federal Ministry of Education and Research (BMBF), the VSEK project [116]. When VSEK reached its conclusion in 2007, study one was still in an early stage. At that point, we had already gained some interesting insights into the practices and challenges of distributed development work in German SMEs. However, as a design oriented group we were interested in actually deriving practical recommendations and tangible ideas for support tools from our exploratory material. Hence, we needed to collect more fine-grained data in order to identify solutions grounded in practice, and thus we reconsidered the opportunities for further research. Being in contact with two SMEs involved in offshoring was a natural point to start, having been granted access before both companies. However, as we contacted them again it quickly became clear that the gatekeepers, i.e. the managers of the companies, both regarded their role in the study as fulfilled. Even though they both agreed to grant us a couple more days of access for participant observation and interviews in their companies (an offer we gladly accepted), they made it very clear that their willingness to grant further access was limited. As we intendet to study their work practices in much greater detail, this hesitance to give us further access created problems for us.

In search for external funding, we submitted several new project applications that also included funding for the two companies as project partners. In this phase, it became clear that the manager of one of the companies ("Alpha") was especially interested in getting involved in academic research projects. At the same time, the offshoring project of the

other company was terminated, thus making further research on this topic impossible (although we managed to conduct interviews with several participants on the reasons and outcomes of this decision).

In 2008, we successfully acquired funding for the ARTOS[3] project from the German Research Foundation (DFG). Framed as an exploratory study on the practices of coordinating distributed software development work in SMEs, ARTOS was also meant to support practitioners by identifying useful practices, as well as means for tool support. Methodologically, we had decided to use a particular action research design that had already proven to work well in several of our research projects: Business Ethnography [159]. Following the Business Ethnography concept, we needed to establish a joint project together with a company practicing offshoring as a basis for our research activities, study the multiple facets of the situation in the field by means of ethnographic methods, analyze them and mirror the related findings back to the participants.

As the ARTOS project did not include funding for enterprises, we needed other argments to convince possible project partners to join in. As we knew that the manager of Alpha was still interested in further cooperation, we contacted him and informed him about our new research project, asking if he was willing to participate. In this discussion, we presented the intention to support the company in organizing their inter-team cooperation and learning. This proposition was based on our experiences from study one, which had revealed that the theme "learning in distributed organizations" was an important issue for the company (see above). For example, in the first study, some developers in company Alpha had reported to be unhappy about parts of the company's working arrangements, finding it difficult to maintain an overview of what was going on in the company. At the same time, it was difficult for them to improve the situation, as the company lacked resources and procedures for systematically discussing and implementing changes [32]. Hence, we offered to support Alpha's capabilities of self-organization by giving them the opportunity to conjointly address specific practical problems of inter-team coordination and knowledge exchange if they decided to participate in our study. In line with the Business Ethnography approach, we explained that what we were offering was not consultancy, but rather an attempt to initiate mutual learning processes between the company and us as researchers by stipulating a tentative and exemplary implementation of organizational learning procedures. The manager of Alpha gladly agreed to our idea, although his interests in our study turned out to be a little bit different from ours in the course of the project (more about this issue later).

[3]See http://www.artos.uni-siegen.de.

Details on how we conducted our second study will be presented in section 4.4.

4.3.2. Analysis of our Research Methodology

The material for this paper is based on our methodological reflections and notes from the field that we had accumulated. For the analysis, we followed the approach outlined in section 4, using Grounded Theory-oriented coding and analysis in order to identify issues and challenges documented in our research notes.

As we had constantly deliberated on our choice of methods, as well as on the appropriateness of their application, we had a rich basis for a systematic reflection on the two different phases of our study. While the notes concerning our research reflected the research rationale of our project, we complemented the analysis with a systematic reflection on our own role in the field (as required by Business Ethnography). For doing so, we scrutinized our documentation (field notes and interview transcripts) concentrating on how our application of methods and our research focus affected our perspective on the practices, as well as the conclusions we were able to draw. We also cross-searched the interview transcripts and field notes for related evidence.

4.3.3. Case Study: Company Alpha

Alpha, the company at the center of our Business Ethnography, provides data processing software and services for statistics and documentation. Most of the approximately 20 employees of the company are software developers who work in several teams on different projects (see table 4.1 for details). The products include databases, documentation and presentation systems used by cultural institutions, like archives or museums. The services that are offered are concerned with the use and adaptation of these products. Since the mid-1990s, the company has employed a project manager as well as three to seven software developers in Tomsk, Siberia. At the origin of this decision was an internship spent by a Russian developer at the German company. Due to the positive experiences they had working with him, the German manager decided to expand the cooperation.

The first offshoring project aimed at re-engineering an existing software product and was led independently by an offshore project leader, who directly reported to the German manager. The offshore project leader was also responsible for the communication with the German customers, providing support and investigating bug reports. Despite unexpected delays in the development of that first project, offshoring was expanded to several

German Site: Bonn	Offshore Site: Tomsk
- 1 CEO	- 1 project manager
- 3-4 project managers	- 3-7 developers
- ca. 8 developers	

Table 4.2.: Details of company Alpha (as our research spanned several years, the numbers of developers employed varied).

other projects, which involved a closer cooperation between German project leaders and offshore developers.

4.4. Conducting a Business Ethnography in our Project

After our idea of conducting a Business Ethnography in company Alpha had been approved by the German manager, our research started with a 1-h kick-off workshop at the German company where we presented our ideas and our approach to the employees in order to facilitate the emergence of mutual aims. After the manager of Alpha had introduced us to his employees (most of which already knew us from the first study) and had stated his support for our project, we gave a talk explaining the research topic and our expectations, as well as the foreseeable benefits for the company. In doing so, we briefly presented some findings regarding possible problems and challenges from the first study, and explained how the company could benefit from collaborating in our study. We also stated that we, the researchers, would take the initiative and undertake most of the work, while the employees would be expected to grant us some time for conducting interviews, as well as to allow us to engage in participant observation. The main aim of this shared endeavor was described as the initiation of an organizational development process, during which the company would have the opportunity to address identified problems and improve their organizational practices (if they found our findings to be relevant). In order to inspire confidence, we were very open about the limitations of our study, and stressed out that we did not intend to do any evaluation of individual work; we also emphasized that all the results of the research would be anonymized and that data collected will be kept in strict confidence and will not be shared with the management. We ensured that the Russian team was informed about our endeavor through the Russian team manager. He was sent a set of slides in English and we had a phone conversation with him to explain our intentions for the study.

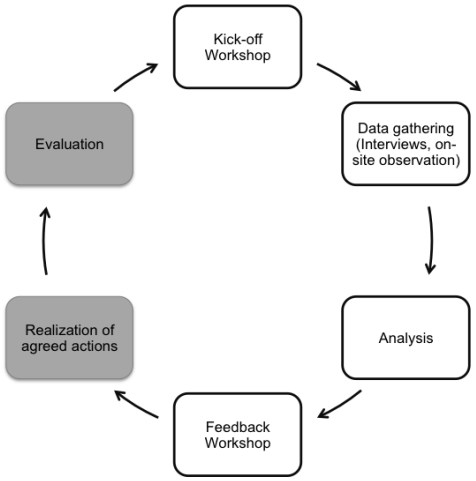

Figure 4.1.: Business Ethnography circle (future steps are grey).

In order to understand the perspectives of all involved actors, we conducted semi-structured interviews with the majority of employees (developers, testers, secretaries, project leaders) of both the German and the Russian team. Each interview lasted between 45 and 120 min., depending on the role and involvement of the interviewee in the cooperation between Bonn and Tomsk. During the interviews, we focused on discussing issues we had unveiled during our previous study with the company, and on aspects that were related to our research question: how to support the inter-team coordination and knowledge exchange practices. During a second round, we conducted telephone interviews with the Russian team manager as well as with his developers in Tomsk.

In addition to the interviews, we conducted additional on-site observation at the German company. Over a period of three months, we visited the company on a regular basis (1-2 days/week). We did not actively take part in the work, but we were allowed to accompany the developers during their working days. In this phase, we also got access to some internal documents which were relevant at that point of time (for example to project plans, customer agreements , and requirements documents, as well as to chat-logs—as far as they were relevant for our research topic). Thus, we had many opportunities to observe local and distributed articulation work practices in the context of meetings, individual work situations and cooperative tasks. Again, our fieldwork was documented by taking field notes and pictures, as well as by collecting artifacts.

For the analysis part, we chose a Grounded Theory-oriented approach for investigating the different perspectives, perceptions, and views of the participants on the cooperation with the other team [223]. After each research step (e.g. after each interview or period of on-site observation), the transcripts of the material (field notes, chat-logs, specification documents or interview data) were coded in an open process using the Qualitative Data Analysis (QDA) software Atlas.ti. In doing so, we analyzed the coordination and knowledge exchange practices we had documented with regard to the problems (and ideas for improvements) that the interviewees reported during the interviews. Three main categories emerged which were focused during the subsequent analysis: social, organizational, and technical aspects of the onshore-offshore cooperation. On this basis, we prepared a workshop for company Alpha, in which we presented the results of our research to the participants in Bonn for discussion. The aim was to develop a set of specific measures, in order to improve inter-team coordination and knowledge exchange within the company. The workshop was organized as a half-day event and consisted of two parts.

The first part began with a presentation of our findings. It ended with some explicit suggestions for improvement, divided into three main areas that needed improvement: technology, organization, and social aspects. While we drew on scientific concepts like articulation work [219] to explain our findings on a theoretical level, we attempted to be as specific and comprehensible as possible in describing the practices in the context of different perceptions, needs and concerns of the participants. In doing so, we took care to anonymize the presentation as much as possible. Subsequent to the presentation, we discussed our findings with the practitioners, asking for feedback and clarifications. Our aim was to come to an agreement with regard to possible improvements of the onshore-offshore cooperation. During this part, the German manager offered to leave the room at some point, giving us some time to get feedback from the employees only. This unplanned opportunity led to a critical and interesting discussion, which continued even after the manager had returned to the room after about 20 min.

The second part of the workshop aimed at a specific topic the company had a particular interest in: opportunities for expanding into the Russian market. As this issue had not been targeted by our research, the German manager chaired this part, explaining his interest and aims as well as comparing and assessing possible strategies and dangers. The practitioners and researchers in the room then discussed this topic. The workshop ended with a final round of feedback, every participant being asked to summarize the key points from his perspective.

The researchers then summarized the results of the workshop in minutes—including action items—, which were sent to all the participants. For example, at the organizational level, the results were centered on changing the form of the weekly meetings the company held in order to support better the knowledge exchange between employees. During the interviews, these meetings had been described as often being too time consuming and unfocused. At the same time, many developers found the occasion useful to get hooks for later discussions, for example in the kitchen during coffee breaks. After discussing these aspects at the workshop, the participants agreed to limit the time allocated to each speaker and shift the focus towards brief overviews about current projects and problems, while deferring the details for later project team discussions. On the technical level, one of the action items stipulated the use of instant messaging (which was not used by all members of the German team despite the many benefits its users reported), and another was the installation of a digital board to serve as an extension for the minutes of the weekly meetings. Last, but not least, the social level action items were centered on the role of „bridges", i.e. the capacity of human mediators to support mutual understanding and communication between the teams, to be realized by exchanging knowledgeable team members for longer periods of time.

4.5. Challenges of GSD Research

This section describes several challenges we had to deal with as we conducted our research project. While we focus on both our experiences with the second study and the adopted Business Ethnography approach, we also look at general issues we have found while conducting our first study.

4.5.1. Studying Global Work Practices through a Local Lens

A general challenge we had to deal with during both studies was that our observation of inter-organizational work practices was only possible through a local lens. Since it was not possible in our project to deploy several researchers to observe the work practices synchronously at the two different sites, cooperation processes could only be observed from the perspective of one team at a time. This sometimes resulted in lacking awareness of what was going on at the remote site, which was hard to maintain during our on-site observations in Bonn. Since we were dependent on the same media and tools for communication as the software developers and the managers in Bonn were, our research

was subject to the same difficulties the software developers located in Bonn had to face for conducting distributed development work. The limitations of the communication technology also made it hard for us to conduct interviews with the remote team members, as the VoIP connection to Russia was slow and not always stable. The following excerpt from our field notes gives an idea about how this could lead to problems:

> "[During the interview] there was some sort of misunderstanding, as suddenly [the interview partner] went offline and did not sign in again over the rest of the day. I wrote several emails to him and hoped that he would soon return. (...)
>
> [After the weekend] it turned out that [the interview partner] had left the office at 4 pm to pick up his kids from the kindergarden (...). Apparently, he had told me so but I did not understand it due to the bad connection."

Despite such technical problems (which also made the transcription of the recorded interviews more cumbersome in some cases), it was also very difficult to even get an idea of the physical layout, not to mention the local unwritten rules, local practices and cultural differences of the remote site (Tomsk) we were interested in. Even though it helped that we were able to refer to our experiences from the first study, during which we had visited the remote site, it was hard to obtain clear information concerning some hard facts such as the actual size of the remote team during the interviews. Since no one from the local team had regular contact with every team member abroad and since the size and structure of the offshore team had changed several times during the year prior to the study, the local perception of the remote team differed. As one of the German developers explained:

> "Well, I am not quite sure about the exact number of colleagues (in Tomsk). That's a clear sign that I probably don't know everyone yet. And, yes, so far I know the three who have been here in Germany (...). But the others, I don't know them."

Also, there was no clear evidence available of the exact reasons and dates of changes in the cooperation. Even here, the explanations of team members differed considerably. For example, our inquiry ignited a dispute at the company between a senior developer and the manager on the question of when the whole cooperation had started.

Another challenge was related to the high complexity of the articulation work practices in the GSD projects. This involved the simultaneous use of several inter-related synchronous and asynchronous media (Skype, email, ICQ, etc.), tools (bug-tracking-systems, etc.), and artifacts (specification documents). This turned out to be a problem

for the developers, too: "(...) one notices again and again that information is there, but is distributed in a way that makes collating it cumbersome. . ." . As it was hardly possible to understand the complex interrelationships by referring solely to interviews, our research required monitoring a broad set of media and artifacts, as well as their complex inter-relations. This turned out to be quite difficult, as the time frames for observing the inter-site cooperation were limited. Even when trans-local work had to be done, the actors often omitted to involve us when they contacted the remote team, as they did not always plan for such incidents, and as these occasions were usually related to emerging problems that required their attention more than paying attention to the interests of the researcher. As a result, we were sometimes involved too late or not at all, so that we had to conduct interviews in order to reconstruct the missing events. In addition, documenting the context of the collected data and linking different kinds of information (artifacts, field notes, pictures etc.) with each other required great efforts from us, especially since we were usually not granted full access to log-files and databases.

In particular, documenting complex work practices was challenging, requiring extensive and accurate field notes on the role of several inter-related media and artifacts. As the observed work trajectories were complex and hard to predict, it turned out to be difficult to relate the various sources of data to each other during the analysis. Basically, the problems were related to the high granularity of data that was needed to reconstruct the complex interactions of the developers and teams, and our limited resources in terms of research time and the limited access to data posed complex obstacles that we had to deal with.

4.5.2. Adapting to Changing Interests of the Company

Despite the initial presentation of our research interests at the beginning of our study, the aims and even the schedule of the shared project we agreed with the company on turned out to be not considered important by several of the participants. For example, developers who were not involved in the management of software projects were willing to give interviews, but during the negotiation of possible dates for interviews they often said they would not redeem themselves as being important and stated they did not or only hardly remember the aims of the research project. For example, during the interviews we often got answers like "I will be happy to help you with this", indicating that they ignored the intention of the joint Business Ethnography project. An excerpt from our field notes, taken after an interview with a Russian developer in Germany, further illustrates this issue:

> "Again I had the impression that the Business Ethnography aspect of my research was not really appreciated by the company. [The Russian developer] told me much about his attempts to improve the cooperation between Tomsk and Germany, without referring to the planned workshop. He didn't seem to regard the interview as an opportunity for him to impact the cooperation."

This became even more apparent when the interview phase finished, since it was hard for us to remain in constant contact with the field site during the analysis phase.

As the company struggled to deal with the consequences of the financial crisis during this period, and as our own scientific work life imposed other engagements to be prioritized (like writing papers or attending conferences), the shared commitment deteriorated more or less and we had to make additional efforts to get back in touch with everyone again, and even to negotiate the aims and scope of the shared workshop again. It took us two months to transcribe all the interviews and to analyze the transcripts with regard to the focus of our research project. Then, finding a possible date for the workshop turned out to be difficult due to conflicting commitments, for instance the company had to finish a product and the researchers were writing papers. As a result, we had to postpone the workshop for another month.

During this 3-month period, we visited the company on a regular basis in order to keep in touch. However, our involvement in the company declined during this phase, fact that became apparent when we finally came to plan and prepare the workshop. As we approached the German manager in order to set the date, he suggested to change the topic of the workshop into "Expansion to the Russian market", which had not been at the focus of our work. In order to satisfy his interests, we decided to divide the workshop into two parts, as described in section 4.4. However, we found it quite hard to deal with this question not having much interview material on this aspect, as we had focused on a totally different topic.

But even with regard to our core topic, improving the cooperation between the sites, the workshop revealed that the company had already begun to change several aspects of the cooperation. The following excerpt of our field notes from the workshop illustrates this issue:

> "Interestingly, [Alpha] has already begun to implement some of the changes we have suggested during the workshop [...]. Hence, there are change and learning processes in place which might have been triggered by the interviews. On the one hand, this is an encouraging result, but it implies that

> we might have been too slow with our analysis or that we did not succeed to communicate our own role in the learning processes well enough (or both)."

As described above, the weekly meetings of the company had often been described as being not very useful, as many employees considered them too long and too formal. When we suggested changing the organization of the meetings to a more focused and efficient modus operandi, it turned out that the company had already decided to abolish the unpopular meetings. Even if our suggestions suggested a discussion about reintroducing the meetings in a different form, we had not been aware of these changes (which also affected other aspects of the cooperation we wanted to discuss).

4.5.3. Dealing with Micro-Political Conflicts between the Sites

Another important challenge we had to deal with while conducting our Business Ethnography was getting access to the field. As we were already in contact with company Alpha, due to our first study, this turned out to be less difficult for the German site.

However, it turned out to be very challenging for us to get involved in the Russian team. As it turned out, the relationship between the sites was constrained by ongoing conflicts regarding duties and responsibilities, as well as by social aspects of the cooperation. Russian and German developers reported different estimations concerning reasons for the failure of a shared project. As teams blamed each other it was very interesting to analyze the different views that revealed different perceptions of what constitutes "good software development", influencing the cooperation practices. However, the different perspectives also led to problems. In this regard, a Russian developer who lived in Germany at the time of our research and who was acting as some kind of a bridge between the teams told us that most of the communication of his German colleagues with the Russian team would consist of criticism, and positive developments were not acknowledged properly. As the Russians usually were in a sort of inferior position when it came to conflicts, they were sometimes very sensitive to criticism. He pointed this out with an example:

> "I think it was the day before yesterday, when we (...) requested to our Tomsk colleagues to temporarily take down the Tomsk website. We explained that the website content should be more strictly controlled (...). Our colleagues in Tomsk interpreted it as an authoritarian command, and found it very rude. They asked us in a quite emotional way: why you are starting war on this?" Due to the asymmetric power relationship, developers from both sides took care to avoid discussing problems too directly. As a German em-

> ployee explained: "When dealing with Tomsk, you have to take care to find the right words (...)".

This also had consequences for our research, as the Russians were rather reluctant to talk openly about the problems with the homepage in their interviews. Asked about general problems of the cooperation, they avoided talking about problems and stated that everything was running quite well (in contrast to the German developers who openly reported problems between the teams [26]), for example: "So for me, everything is ok." When explicitly asking the Russians about some of the problems with the homepage, the answers were usually elusive, like "there had been a set of bad steps" that had led to some "minor problems and misunderstandings". The following statement describes an experience we described in our field notes:

> "[When interviewing Russian team members,] one has to act very carefully. It happens that the interviewee seems to feel cornered if one interprets vague suggestions about changes as an indication for lower-level problems."

It was challenging for us to convince the Russians that like in our first study we still had a more or less neutral role, and that they could use the intended workshop to influence the offshore-onshore cooperation arrangement and to address problems that were relevant for them (and not necessarily for the German team). This problem was probably aggravated by the geographical distribution between the researchers and the offshore site, as well as by cultural and language related issues. For example, some of the new developers from Russia had problems to answer our questions in English during the interview, which prevented us having an open discussion on change opportunities.

Visiting the remote team before the workshop might have helped in this regard, but organizational issues and funding limitations prevented us from organizing more than one visit to Tomsk. Similarly, these issues prevented the Russians from participating in the workshop, somewhat limiting its scope to the perspective of the German team.

4.6. Discussion

The presentation of the challenges illustrates the complexity of doing research on work practices in the context of GSD projects. The following discussion aims at analyzing our experiences in order to develop a strategy for dealing with these challenges. As pointed out in section 4.5, we encountered several challenges while conducting our Business Ethnography. These were related to the local view on global work practices (4.5.1),

the changing interests of the participants (4.5.2) as well as possible frictions in the field (4.5.3).

Dealing with challenges, such as observing the global through a local lens or tackling the high complexity of practices involving the simultaneous use of several media are essential concerns and are probably inherent for most studies looking at Global Software Development practices. Methodologically, these challenges hold a clear danger to acquire a biased view on the cooperative work arrangement. The many obstacles in such fields makes it desirable to spend as much time as necessary at the field site in order to observe the given situation long enough to understand it (even though such observations are necessarily biased towards the local perspective). In order to deal with the complexity of distributed development work, we expected that referring to a single source of information (like observations or interviews only) would not be sufficient [8]. Hence, apart of conducting interviews and observations, we also wanted to access artifacts and media, including server and communication log-files, email repositories and bug-databases (just to name a few). In practice, observations and interviews turned out to be our main source of information, while artifacts and media played a subordinate role and were only accessed when the actors themselves referred to them. This might be related to the background of the first author in cultural anthropology, whose perspective on ethnography is centered in understanding the perspectives and interpretations of the practitioners (and not so much focuses on revealing the underlying social patterns or using ethno-methods as a means and an end of ethnography) [91]. However, this ethnographic view provided a valuable basis for the analysis of the multiple perspectives which nonetheless required the interpretation of field material with a great deal of sensitivity for the context in which it was created. This was especially related to the intricate inter-relations of practices on the backgrounds of a dynamic development approach as well as to the cultural and language-related barriers in international teams. This is especially the case when dealing with the virtuality of the inter-team cooperation, where the geographical deparation of the researchers from the field may result in a lack of common and mutual perception. It is well known that voice connections can be bad and communication is usally subjected to language problems and to a general lack of common ground. Sometimes, researchers even have to use text-based communication as an alternative, which has implications for the perceived authenticity as well as for the underlying textuality, narrowing down the form of interaction to written words and a couple of emoticons [112].

In practice, the degree of access is very much dependent on the good-will and cooperation of the participating companies, which often limits the possibilities for research to be

undertaken in terms of research time and levels of access. This became apparent in our first study at the two German SMEs. Because our partners saw no benefit from their participation in our study, they limited our access to artifacts, which they considered as confidential, even when the artifacts were extremely relevant for our research. Furthermore, after allowing us to interview their employees and granting us access for a number of days of on-site observation, companies usually signaled that they regarded their role as having come to an end. In this regard, following an action research approach helped us to (re-)negotiate access, as it allowed us to offer something to the companies. Since the related research questions were recognized by the participants as being important and relevant for themselves, we were able to raise awareness about the scope of our study at the company by offering some feedback with regard to possible improvements and joint learning processes (even though the motivation of Alpha's manager to participate in our study might also been affected by his abovementioned interest in cooperation in academic research projects). At the same time, the workshop turned out to be an interesting instrument for us in order to discuss the validity of our findings with the participants in the study. As Dittrich [63] pointed out, simply spending time on-site is often not sufficient for understanding the implicit aspects of the daily communication and cooperation of software development teams. In distributed settings, this seems to be even more the case due to the particular conditions of globalized software projects [184]. In this regard, it turned out to be very valuable for us that we had already spent time at the company during the first study, and also visited the remote team in Tomsk. Being able to refer to these experiences during our Business Ethnography was very valuable for us in conducting the interviews, especially since we already knew many of the employees and also had some insights into the situation at the cooperating site in Tomsk right from the beginning of our second study.

Compared to our first study, becoming immersed deeper into the field by contributing to a shared intention as researchers certainly helped us to get a more profound understanding of what was going on in the second study. However, adjusting to the second issue, the shifting interests of the participants, turned out to be very difficult. From our perspective, the problem we had managing our role in the joint project was related to the conditions of our research. As it had been us who suggested a shared project with the company in order to improve our collaboration, it was our duty to keep the participants from company Alpha motivated and interested in our joint project. Even though the practitioners were aware of the general problems in their existing cooperation arrangement, solving them was not a priority because of the daily problems they had to deal with in their development work. As we participated in the project as researchers,

and not as software developers, it turned out that the rationale of our research focus collided with the company's rationale—the development of software. As a result, we had to adapt to this shift in interest when the German manager suggested a new topic for the joint workshop. In this regard, it would probably have been beneficial to at least follow a much more iterative research approach, in order to allow us adapting quicker to this change of focus.

The third issue we had to deal with was perhaps the most challenging, as it was connected to the particular conditions of the company where we did our field study: the micro-political conflicts between the sites. Since we were regarded as colleagues to some extent (at least by the German developers), and not so much as outside observers, we lost our neutral status and we had to take a position with regard to operational and strategic questions of the offshore-onshore cooperation. While our role as scientists, as well as the perspective of the joint workshop helped us to channel the expectations toward the time after the analysis, we were no longer in the position of outside observers. While we made no claim about becoming yet other members of the German team and rather wanted to act as learning mediators, it turned out to be difficult for us to make this role obvious to other project members— especially to the Russian team. As the project participants based in Russia were hesitant to talk openly about problems, and as we had difficulties innegotiating our aims with them, our access to their views and expectations was clearly limited. A possibility would have been to establish a steering committee, as Rönkkö [184] suggests, for making our role more transparent as well as to include Russian team members in the related decision processes. However, the described conditions made this approach very difficult and limited our access to the Russian team.

In general, it has been very advantageous for us to employ Business Ethnography as a research concept, especially for making our role explicit in the context of the research activities and for negotiating access, by having something to offer to the participating companies [245]. This is a benefit most action research approaches have in common. Although applying our approach to a distributed software team went along with additional challenges, the relatively deeper immersion helped us obtain a much more detailed understanding of the articulation work practices we were interested in. Hence, Business Ethnography allowed us to leverage many of the known challenges of studying distributed teams [5]. Even though Business Ethnography as a method is not very specific in how research methods actually have to be applied, the theoretic lens it offers helped us in dealing with the conflicts we found in the field, as it does not only require the researcher to remain aware of his political role in the field, but actively and explicitly considers

this role as a subject matter for the research itself. Hence, even though we were forced to concentrate on the perspective of the German side of the cooperation, we think that the awareness and transparency implied enabled us to understand the related conflicts between the sites in a much more detailed way, compared to our first study. On the other hand, it has to be stressed that we also benefited substantially from being able to refer to our experiences as well as to the trust we had gained by conducting our first study; benefits, which can not be attributed to the Business Ethnography approach.

4.7. Conclusion

The geographical distribution of distributed software development projects imposes new challenges to researchers in the field of GSD. In this paper, we have introduced Business Ethnography as an action research approach and discussed our experiences with applying this method in practice in the context of software offshoring in a German SME. Even though we did not fully implement the concept in our project, we have identified several impact factors that influenced the challenges we had to deal with during our research. A possible solution that we found is based on addressing the role of the researchers in the field, in relation to the experiences of the actors that take part in the research. While this kind of studies may not be well suited for measuring success (an aspect which is mostly interesting for many companies from our experience), the broad picture they provide can offer opportunities for unveiling possible improvements and chances for inter-organizational learning. Thus engaging in a shared process with the practitioners may offer several advantages for both parties and allows to leverage many of the known challenges of studying distributed software teams. The additional chances of such approaches include being able to learn about strategies for fostering flexibility in offshoring projects, as well using the findings for further technology development. We also found interesting ideas about situated learning processes coming from the practitioners that may have been triggered by our research.

Acting as learning mediators resulted in both benefits and challenges for our research. On one hand, defining organizational learning as a meta-project was a challenge, and encouraged the practitioners to remain interested in the long term. As our feedback cycle apparently was too time consuming for the practitioners, it could be worth considering approaches that offer shorter and more rapid feedback loops in order to keep the practitioners interested, exploiting informal and situated learning processes for the purpose of the research as well. On the other hand, keeping a research stand allowed

us to leverage expectations and keep a more or less neutral status in the micro-political struggles between the teams, even though this required ongoing negotiation of our role especially with the Russian team.

Institutionalizing results still remains an important challenge for action research in distributed teams. As Dittrich pointed out, successful process and practice improvements are not only dependent on valid findings, but also on social relationships in the field [64]. When the researchers get involved in conflicts on-site and between the sites, transparent and legitimate forms of involvement of the researcher in conflicting issues have to be found. While we have no final answer on how this can be achieved, we believe that one advantage of Business Ethnography is that it actively conceptualizes these struggles and attempts, in order to make them transparent for the scientific community, as well as for the participants.

5. Coordination Practices in Distributed Software Development of Small Enterprises[1]

Global software development has become an important issue for small and medium enterprises. However, the distinct requirements of SME are still not so well understood. In order to contribute to the discussion we present case studies in two small German software companies that engage in offshoring of software development to Eastern Europe. By applying Strauss' articulation work framework we show to what extent SME rely upon situated coordination practices in order to warrant their agility. These practices are applied during discussions in which the actors reflexively evolve problems and solutions from their distinct perspectives and work practices. Thereby they are closely related to formal and informal communication, which takes place both locally and between the different teams. Our findings further suggest that specialized tools for the support of situated coordination practices in terms of articulation work are not so common in practice.

5.1. Introduction

Software development in distributed teams has become an important issue for software companies. A lot of companies expect to reduce their costs and get access to specialized knowledge by concentrating on their core competencies and outsourcing certain aspects of their software development to foreign service providers. It is commonly agreed that offshoring consulting is a growing market and will be of increased importance in the future [164]. At the same time, more and more small and medium enterprises (SME), which form the vast majority among the German software companies, engage in offshoring [247].

[1]This chapter has been published as a full paper in the proceedings of the *IEEE 2nd International Conference on Global Software Engineering* in Munich 2007 [29]. © 2007 IEEE. Reprinted, with permission, from Alexander Boden, Bernhard Nett, Volker Wulf, Coordination Practices in Distributed Software Development of Small Enterprises, Proceedings of the Second IEEE International Conference on Global Software Engineering (ICGSE), 2007. See appendix I.

The trend towards global software development has led to a discussion concerning the organizational means of distributed software development [2, 53, 16, 154]. However, empirical evidence seems to be still sparse, especially in case of SME, which often follow different business strategies from those of large companies and center their offshoring efforts on Eastern Europe instead of Asia or India [60]. There is still need for further empirical studies as well as theoretical approaches concerning strategies of organizing and managing globally distributed software engineering [53]. This endeavor is even more important as recent research indicated that the needs of distributed teams differ from local workgroups and results concerning the latter can not always be transferred in an unaligned way [54, 110].

We want to contribute to the discussion on global software development by presenting case studies in two small German software companies. In our research, we have focused on the role of informal coordination mechanisms by addressing articulation work in long-term offshoring partnerships as factor for warranting agility in distributed software development. After discussing the importance of articulation work for offshoring in SME (section 2 and 3) we describe our research method (section 4). The presentation of our findings (section 5) is followed by a discussion (section 6) that compares our findings to related work of the literature and leads to our conclusion (section 7).

5.2. Global Software Development in SME

5.2.1. Theoretical Perspectives on Offshoring

Decision-Making Many studies address offshoring from the perspective of the related decision process. A great deal of the literature focuses on the adoption of transaction-cost theory or resource-based theory for the deriving of decision criteria [58].

Approaches based on transaction costs concentrate on costs as the main decision criterion. According to these approaches, offshoring is of benefit if the reduction of production costs is larger than additionally incurring transaction costs [240]. What is addressed as major problem in this respect is the negotiation of a contract. Due to the fact that technological change can not be fully anticipated, insecurities regarding the contractual arrangements (principal–agent problem) arise because opportunistic behavior on part of the service provider must be taken into account [125]. This risk appears to be even more relevant to SME because of their financial possibilities that are usually lower [60]. Relying on an

unsuitable partner thus can lead to severe repercussions as backsourcing or changing the service provider will cause high transactions costs in most cases.

In contrast, the resource-based theory addresses competitive advantages of companies on the basis of unique, non-imitable resources and corporate competencies such as process knowledge and patents [49]. According to this theory, offshoring is profitable if the offshored parts are strategically unimportant, if previously neglected technology can quickly be adjusted due to the offshoring, or if resources are released for innovation in other areas [188].

Although different models and taxonomies are proposed [157], offshoring is often reduced to binary "make or buy" decisions. The focus seems to be mainly on the decision-making processes of the management and on early stages. There are still gaps in understanding the offshoring process as a whole, which often is a complex long-term relationship that is not equally suitable for all companies in all situations. Thus, offshoring has to be studied in a differentiating manner according to the specific needs and objectives of the offshoring company [212]. Since research into offshoring is often centered on large companies, providing case studies with focus on special needs of SME appears to be of importance.

5.2.2. Agility as a Core Competency

Achieving a more detailed understanding concerning the different mechanisms of distributed software development is even more important as SME may have requirements to offshoring that differ from the needs of large companies. For example, SME can often not be tied down to critical success factors frequently used in literature like the uniform quality standards of the software process (CMM etc.) [160]. While this proposed formalization is useful in redundant parts of the development process, it may not be adequate in fluid, innovative environments where necessities often evolve dynamically during the development process. If development projects need a great deal of flexibility, for example to react quickly to changing customer demands, more formalization may be no adequate solution [162] and reduce the agility of companies.

This seems to be of great importance for SME because they often are described as especially reliant on their flexibility in reacting to changing customer and market demands [72]. Working in small teams allows for more agile methods of dealing with coordination mechanisms, division of labor and hierarchies. If the resulting agility is supposed to be the core competency of many SME of the software industry, this agility has to be

warranted continuously during the offshoring relationship. Recent research has already addressed possibilities of using agile methods in globally distributed environments that differ significantly from the traditional plan-based approaches [175, 142]. However, it seems to be still unclear how those methods can be adapted efficiently in practice [168, 140].

The increasing offshoring of software development by SME leads to several questions:

- Which effects does the shift from local to distributed teams have on their ability to react quickly to changing market demands?
- Which strategies are deployed by SME in order to warrant their agility on global software development?
- Which implications do the specific demands of SME have for the development of software and the design of tools that support distributed teams in their work?

In order to add to the discussion we introduce two case studies of small software companies. In our research we address agility as subject of dynamic self-organization and adaptability by applying Anselm Strauss's articulation work framework [222] to our case studies. Articulation work is of great importance when plans have to be adjusted to unexpected occurrences or changed requirements [200]. Thus, being able to articulate work efficiently should be an important factor for successful software development and will be addressed in our research. Therefore, we expect to contribute firstly by testing the relevance of the articulation work approach on global software development in small companies and secondly by adding to the understanding of offshoring as business strategy of SME.

5.3. Articulation Work

Articulation work has been a core concept of CSCW studies since the definition of this research area by Schmidt and Bannon 1992 [200]. Strauss' framework aims at a better understanding of the interrelatedness of interdependent actions of cooperating actors [222] and has been of great benefit as a framework for ethnographic research into collaborative work environments [95, 47].

5.3.1. Articulation Work and Software Development

Work arrangements like distributed software development projects comprise a course of actions that include division of labor both in terms of actors and actions. Due to

the complex interdependencies between tasks and actors, small changes may lead to unwanted consequences for the system as a whole. Cooperative work has to be coordinated, not only in terms of manpower but also relating to task-to-task, task-to-person and person-to-person dependencies [220]. As work is constituted strongly by these mutual interdependencies, Schmidt and Simone [201] suggested inherent distribution as a core concept of cooperative work. From this point of view, it is not necessarily just space or time zones which matter, but the ways in which the different actors act semi-autonomously within their respect situations, relying on individual strategies, heuristics, perspectives, goals and motives [200]. It is because of this mutual interdependence that actors need to engage in articulation processes which Strauss described as articulation work. "This is accomplished by means of the interactional processes of working out and carrying through work-related arrangements. Articulation varies in degree and duration, depending upon the degree to which arrangements are in place and operative" [222].

Articulation work, above all, regulates the distribution of tasks: who does what, when, where, how, with which quality, until when etc. Yet, articulation work is more than just coordination by means of the distribution of resources [76]: it is rather some kind of detailed supra-work that mediates cooperative work arrangements. This comprises, among others, the continuous and situated renegotiation of different actions regarding their allocation, assignment, schedule, reconcilement and interdependency but also concerning individual interpretations, perspectives and accepted meanings of work [201], in short: all necessary endeavors to manage the distributed nature of cooperative work.

5.3.2. Articulation Work in Distributed Work Environments

As tasks are increasingly distributed in time and space, as structures become more complicated, as specialization grows and as market dynamic increases, articulation work is likely to become more and more complex [200]. In small collocated teams, articulation work can often be accomplished through everyday social interactions. This articulation sometimes works quite efficiently and adds to the covert nature of articulation work that remains "invisible" and is often not even approved as part of the work itself [211]. However, when complexity of the projects increases in distributed work environments, usual social interactions may no longer be sufficient. Thus, the need for efficient articulation evolves as additional challenge for the actors involved and companies are confronted with a new and sometimes unexpected complexity.

In this regard, formal organization structures, plans and work processes can be perceived as mechanisms of interaction that are proposed in order to reduce the complexity of artic-

ulation work in an objective and deterministic manner [201] and supplement other forms of social interaction such as e-mail or chat communication. However, one important characteristic of articulation work is that it seems to elude formalization. Necessities regarding coordination of software development projects can be volatile and complex, crossing the borders of established work units. As work environments can be perceived as being dynamic, it is not possible to anticipate every eventuality and coincidence adequately. Thus, formal descriptions can not be complete and their use and interpretation in practice always require articulation processes themselves [89]. As a result, a stronger focus on formalization is no solution for the arising problems of articulation work.

In order to be able to articulate the interdependent tasks of distributed project work, actors rather need access to appropriate means of communication. Being able to engage in informal communication in this respect thereby seems to be a key success factor of distributed software development [108]. With geographic distance, vital informal communication becomes less frequent and poses obstacles for efficient articulation work. Grinter et al. have shown how geographic distance may lead to ambiguity and misunderstandings which slow down the development process [106]. Proposed solutions usually center on two different strategies: reducing the need for frequent informal communication or easing the informal communication by technical means, for example by supporting concepts like awareness [208], or by a targeted combination of various communication channels [54]. Usually this is to be accomplished by using specialized tools like groupware applications, which are proposed for the support of articulation work in distributed work environments [95, 47]. Following these considerations, it is important for our research to obtain more detailed insights into the meaning of articulation processes for global software development of SME and into how these processes are taken into account during offshoring projects.

5.4. Research Method

To address articulation work as informal and situated practice [226], organizational patterns have to be addressed in two directions: espoused theories are those that actors claim to follow, while the theories-in-use often have to be inferred from actual work practices [9]. Our research is strongly oriented on ethnographic methods, which offer many advantages for the analysis of differentiated relations in complex environments [231, 227].

5.4.1. Interviews

The first stage of our research was an exhaustive analysis of the literature on offshoring, covering discourses of various communities of practitioners and scientists. Based on our findings, an interview guide was created. The guide aimed at articulation work in form of coordination and communication in SME as well as at general assumptions concerning software development and offshoring. As a second stage we conducted twelve interviews with managers, project leaders and developers of small and medium software companies, who were actually involved in offshoring projects or had experience with offshoring in the past. The interviews were held at the respective companies, and recorded. After transcription of the material, a first analysis of the findings followed, which resulted in a comparison of espoused differences of perceptions and strategies concerning offshoring and software development in general. We identified two companies in our sample which seemed to differ strongly in their software development approach and in their organization of the offshoring relationship to their partner firms. Their espoused different perceptions of successful offshoring organization as well as of formalization made them interesting cases concerning articulation work practices in SME related to formal and informal organization structures.

5.4.2. Participant Observation

The third stage consisted of two field studies in form of participant observations that were conducted at the two companies. In order to come to a better understanding of the multiple perspectives and theories- in-use in complex work environments, a third participant observation was conducted at the Russian partner company of one of our sample companies. The respective companies were visited over a period of five to seven working days each. We had the opportunity to observe local and distributed articulation processes during meetings, individual work situations and cooperative tasks. Informal interviews were conducted and we were allowed to analyze artifacts such as e-mails, chat protocols, internal work papers and white board sketches. The findings were documented by means of field notes and photos which were taken during the research. Apart from expected differences, we also found many similarities between the ways both companies organized their distributed software development.

5.4.3. Grounded Theory Analysis

For the analysis of the collected data, we drew on Glaser's and Strauss' Grounded Theory [223]. After each research step, the transcripts of the material, both field notes and interview data, were studied. This procedure was repeated after each participant observation. Our aim was to identify articulation processes in context of the work trajectories and to recognize their meaning for the actors. Relating to the Grounded Theory, we wanted to "let the material speak for itself" as much as possible. Data was coded during a process that consisted of several stages. At first, we composed categories based on the findings in the collected data. Then these categories were related to each other and developed during the further research. Our findings then were related to the literature with a focus on articulation processes during software development. Thus we concentrated on relations between formal work organization and the actual work processes, which were not considered as mechanical "performance" of formal specifications but as creative reaction to situated work contexts [226]. This focus on informal and unplanned aspects of software development in distributed teams led to significant results concerning the immanent logic of development processes, the hidden nature of articulation work and discrepancies between the formal organization of software development projects and the actual work practices, which will be presented in our paper.

5.4.4. The Cases

Alpha

Alpha is a company providing data processing products and services in the field of statistic and documentation. Most of the approximately 20 employees of the company are software developers who work in several teams on different projects. The products comprise databases, documentation and presentation systems used by cultural establishments like archives or museums, the services are offered around the use and adaptation of these products. Since the mid-1990s, the company has been employing four software developers in Tomsk, Siberia. The basis for this decision was an internship of a competent Russian developer, who still works for the company. Based on this positive experience, the decision to engage in offshoring was taken and the offshore team was expanded. In formal terms, the cooperation is carried out in form of a cost center operated by another German company which is formally the employer of the Russian team members and responsible for the communication with the authorities. However, the deadline control as well as the distribution of tasks and requirements is completely carried out by Alpha. Project

leaders are situated in Germany and assign tasks to the Russian colleagues. In case of one project, the team leader of the Russian developers is also the responsible project leader, who communicates directly with the German manager. Since the products of the company are often specialized versions dependent on a common code base there is also much communication to a German project leader who is responsible for an adjunctive product. During the interview in the first phase of our study, the German manager underlined the reliance on flat hierarchies and flexible self-dependent work. Formalization and the use of development models will be considered if the customer wants this but from the perspective of the company this is not necessary:

> "Development models are fashions, and subdued to personal initiatives. (...) For example the federal state North Rhine-Westphalia wanted [us to use] the V-Model. But in the end all what was left in practice were just some acceptance checklists, which could have been provided by any other product management system."

Beta

Beta is a company that offers a standard software solution for process modeling and services in the field of process management. The management is situated in Bonn, while the software development is carried out in an office in Berlin by seven employees. The company also engages approximately 140 freelancers who work as consultants in customer companies and offer assistance in the field of process management. In 2002, a branch office in Saint Petersburg was founded in order to reduce the developing costs. According to the manager the decision for a partner was based on personal relationships to several foreign developers but was taken admittedly mainly on the basis of the wage level. Approximately eleven software developers in two project teams now work on product development of the regular versions in Saint Petersburg, Russia. However, there is also some software development done in Berlin. The project management of the software development is based in Germany, there are five employees concerned for the greater part of their working with managing the software developers in Russia and providing support for customers. In contrast to Alpha, the manager of Beta perceived successful offshoring of software development as closely connected to consequent formalization of development processes.

> "After all, everything works fine, but only because we have documented our processes with our tool. Our development and service process is close to CMM, with templates for documentation. If we did not have this, it would

be much harder, also to incorporate new developers. We track every bug and every feature, there are alarms if deadlines are not fulfilled and so on."

5.5. Results

Despite the differing perceptions concerning the organizational needs for successful offshoring of software development, in practice similarities became apparent. Both companies in our field reported that their project leaders are usually situated in Germany, while the foreign teams act as extended team members and are directed and controlled by the German company. Thus, German project leaders were said to serve as a connection to the German customers, translating their needs into specifications. Then, these specifications should be classified and assigned to certain developers, who can be situated in Germany or abroad. In case of the latter, a team leader at the foreign company is responsible for the assignment of tasks to his developers and the timely delivery of the results. The results then are incorporated and tested in Germany. The only exception was a project of company Alpha that is managed by a Russian project leader, who answers directly to the German manager. Concerning to the German developers this constellation is only possible because the respect project is a standard software solution and thus requires much less communication with customers.

5.5.1. Bug fixing

Other differences apply mainly regarding the formal handling of documentation and specifications. The formal organization of task distribution is accomplished with several tools. Bug tracking systems serve in both companies for the cataloguing and assignment of bugs. Alpha uses the open source system Mantis, while Beta relies on the tool SQA. In contrast to Alpha the documentation of bugs is clearly formalized in company Beta. If a developer finds a bug, he has to follow the same procedure as a customer and report this bug to the hotline of the quality assurance. The QA then will try to reproduce the bug and write a standardized bug description. The company does not want the developers to write bug descriptions themselves, and it is not allowed to simply fix the bug when it occurs. Interestingly, in both companies the German team members had learned to write proper bug descriptions from the Russians. One of the German developers of company Beta pointed out: "Prior to the offshoring, all work happened locally and there would have been no need for ample descriptions because one would have been able to simply

ask a colleague" (Field notes, 26th January 2007). Since this is not possible anymore, the developers had to reduce ambiguities and write much more precise documentation.

5.5.2. Specification of Features

Specifications for new features are handled in different ways: Beta relies on Lotus Notes and maintains a central product database and a development database where features and templates are stored. The documentation language is English. Company Alpha handles the storage of specifications with several word documents, which are in the responsibility of the respect project leaders. A first version is written by the German project leader and acts as a basis for the negotiation with the customer as rough project plan. The specifications are translated into English and supplemented with notes for the Russian developers, who will then in turn do the programming. Beta follows the same formal division of labor with the Germans writing specifications, the Russians implementing them and the Germans controlling the outcome. But as the participant observation showed, this process does not always work in practice because the five German team members of Beta do not manage to write specifications quickly enough to keep eleven (or in the past even more) developers in Russia busy. Thus, despite the formal organization of Beta, the Russian team members sometimes have to write their own specifications for features, which will be controlled by the German project leader, and the Russian team leader may delegate tasks to German developers, too. This was also acknowledged by the manager of Beta:

> "In special cases there will be verbal communication with the Russians, who in turn write their own requirements in English. These requirements will then be compared in Bonn with the requirements of the customer."

Beta uses SQA to track the progress of the work in the partner company. The German project leader can see which developer is working on which bug and how far the work has progressed. However, the databases are not always up to date because the Russian developers have to track them for themselves and sometimes forget to change the status of their tasks. Then, informal communication takes place in form of chat requests, for example when the status of important bugs is not changed for a longer period of time. This is also the case when the project leader assigns a critical bug to a certain developer. He will communicate the importance of the task via chat and inform the Russian developer personally and in addition to the assignment in SQA. This practice is also usual when time estimates are made. Often, the project leader asks the assigned

developers how long the fixing will take. Then he documents the time estimates in SQA. Both companies rely on personal face-to-face meetings for the planning of new releases or new products when possible. We observed one of these occasions during the participant observation in company Alpha, Tomsk. Background of the visit by the German project leader was the kick-off of a new project. The company had a new customer who needed a customized version of a similar product of Alpha. The German project leader had communicated with the customer and wanted to specify the new product together with the Russian developer, who was to become responsible for the implementation, and the Russian team leader. In several meetings, the project leader explained the customers need based on a word document he had prepared in German language, and which had been discussed with the customer. He used a white board for detailed sketches of data models and interfaces.

> "The project leader stands at a whiteboard and draws a data model. The Russian team leader and the assigned developer listen to his explanations. The developer sits on a chair and writes down notes into a diary. The team leader stands and listens without taking notes. The project leader explains the differences between the basis software and the new version to be developed, which at this point mainly concern the data model." (Field notes, 29th January 2007).

After a while the Russian developer and team leader began to interrupt the explanations with questions, and the developer copied the white board sketches into a diary and took notes concerning the meaning. One of the meetings was even recorded with a digital voice recorder. During a next step, the Russian team leader took the notes of the developer (or in one case the recording) and wrote a detailed documentation of the discussed specifications, time estimates and whiteboard sketches, using Microsoft Word. The project leader then read this documentation and corrected it, wrote comments or additions. In the meantime, the project leader used this information to write a detailed project plan with Microsoft Excel for the customer, merely as a representation because he had already told the customer his time estimation. Now he felt obliged to fulfill his estimates by tuning the project plan to the promised deadline. In doing so, he sometimes asked the accounted developer about technical details in order to get a better idea of how much time would be needed for the implementation of new functionality.

5.5.3. Communication

Everyday communication is handled mostly by chatting with instant messenger tools. Alpha reported: "These chats often go on for one hour and are centered on problems which occur during the processing. Personnel management is usually handled by telephone." In case of Beta, communication takes place mainly in form of chats, too. Beta relies on Sametime, integrated in Lotus Notes, offering communication functionality of an instant messenger with some special features like desktop sharing. Despite the focus on chats, the project leader of Beta perceives his relationship to the Russian colleagues as a very personal one. In the beginning, there were much more telephone calls, but chats are preferred as a flexible form of communication with quick reaction times and the possibility to chat with several persons at the same time. In doing so, the instant messengers are usually used asynchronously, with shorter periods of intense communication. As answers are often not needed instantly, it is usual to send a question to somebody and continue working until an answer arrives. This may take some time. Thus, as written communication is preferred, articulation processes may take longer than for example telephone calls, especially when complex matters have to be discussed. However, developers reported it would be easier for them to communicate by instant messengers. One reason is that many of the Russian developers are not good in English. It is easier for them to use chat because they have more time to think about formulations this way. One of the German developers explained:

> "During a desktop-sharing session I observed some of the Russians using a translating tool while chatting. They wrote the answer in Russian, used the tool to transform their answer into English, and then pasted it into the chat and checked for errors. This is why sometimes chatting takes quite some amount of time" (Field notes 26th January 2007).

If time is critical, this is stated in the initial question. However, as we noticed, this strategy does not always work instantly. The German project leader of company Beta had an urgent request to a Russian developer concerning a recently discovered bug that threatened a timely release.

> "As the Russian developer does not answer the project leader sends a request to another Russian colleague, asking whether the developer in question is at his place. The colleague writes that the developer would now be back at his desk, and shortly after he responds himself" (Field notes, 25th January 2007).

Obviously, the problem in this case was that the tool indicated the online status but not whether the colleague was actually at his workplace. According to our interviews, the next step of the project leader would have been to make phone calls or use a Sametime feature that allows initiating an instant desktop sharing connection. Desktop sharing is used mainly to show functionality directly in the developed tool. Thus, the project leader can take control of the mouse of the Russian developer and show him what is to be implemented, describing the expected behavior in the chat. This practice is also usual when the German project leader wants a status report concerning the implementation of new features. The German project leader needs two or three hours a day to communicate with the Russian developers. His strategy is to start with simple tasks and communicate them via chat, while keeping the complicated things in mind. To address the complicated tasks, he uses regular personal visits to Saint Petersburg, for example, when beginning work on a new version. A similar practice was observed in company Alpha, when a new version of the tool was planned, which is developed by the Russian project leader. In order to accomplish the planning, he visited Germany together with one of his Russian developers. There he held several meetings with some of his German colleagues in a similar fashion as during the visit of the German project leader to Tomsk. However, in this case the manager of Alpha had defined the strategy and gave some broad terms of reference, which the developers then discussed during their meetings. After two days of planning, the results were informally presented to the German manager who in turn commented the results and gave further directions.

5.6. Discussion

The case studies illustrate articulation work as continuous efforts of renegotiating the allocation of tasks on distributed software development projects of SME. In this respect, Articulation work thereby can make certain scopes for remote developers necessary like in case of company Beta:

> "Many people make the mistake to think, if I have a great specification, 500 pages of paper, and I can give this to someone, I shall have a product in the end. But this is not possible."

He stated further:

> "Initially, we expected to be able to educate them [cooperation partners the foreign team] in such a way that, after one year, we could tell them

> which feature we want and they would implement it. We have strayed off this utopia. This is not possible."

The companies in our sample had to adapt to these necessities of the development process by changing their processes during the offshoring relationship. This involves practices of documenting bugs as well as the delegation of writing specifications to Russian team members. This result is also compliant with findings of the literature concerning the tensions between flexibility and discipline [140].

Articulation work clearly becomes more time consuming, as specifications and bug descriptions need to be defined more precisely and as written communication in a foreign language is a common part of everyday work. This orientation towards more formal methods may have positive effects on the local development processes in each case, as ambiguity is reduced and development is documented more precisely [175]. However, the participant observation revealed some interesting practices concerning the relation of formal and informal communication processes. The use of a central bug database in company Beta for example turned out to be as much dependent on informal articulation processes as in case of company Alpha, as deadlines, estimates and classifications of bugs and feature specifications are subject to informal communication with involved developers among both teams. This is also the case for the work on shared databases, which should provide awareness but are not always up to date.

Thus, plans do not accurately describe the real work practices [226]. The processes may be documented in form of a cascading model like in company Beta. However, during the observation it became clear that not every step of the workflow will be followed strictly as the description implies: thus a proposed "meeting" may simply be a three-minute talk between project leader and a developer. The project leader of Beta acknowledged that this would be normal for SME and processes in small teams could be handled more flexible. There would be no need for fixed meetings because, in small teams, one would simply know what the others are doing. However, even larger companies that rely strongly on formal development procedures may run into similar problems in adapting to global software development if they do not consider the necessity for informal articulation work.

According to the expected relevance of informal communication [108] developers reported during interviews that they would prefer to work in local teams if possible. This is supported by the observation that both companies rely on face-to-face communication in case of complex negotiation processes like kicking off a new project. The need for personal visits can delay development processes and is likely to reduce agility [106] in terms

of delaying important articulation processes but offers many advantages for the actual articulation of remote project work. The project leader of Alpha stated that his strategy of coordinating a new project would be very comfortable. This way he does not have to specify everything in advance but can evolve the specifications together with his developers who can assist him in making time estimates and point to certain technical details he would sometimes not have considered beforehand. General ideas of the project together with the project leader's knowledge of the customers needs, the domain knowledge about the customers sphere (in this case archives) and the technical detailed knowledge of the Russian developers are thus evolved collaboratively and iteratively into a more and more detailed project specification.

According to the theory, tools would be useful that raise awareness among the actors. However, the companies in our sample did not use specialized CSCW tools for supporting articulation work during their everyday activities. For everyday communication, tools are preferred that are flexible and easy to use. Different communication channels like chats or telephone calls are chosen and switched, as actors consider it to be appropriate. Complex tasks that require a great deal of articulation work are often delayed and solved during face-to-face meetings. Although Sametime offers additional functionality, this seems to play no significant role in practice. Developers of Alpha stressed their preference of the Instant Messenger ICQ because they like to communicate with friends and family members while they are working. Even though desktop sharing and videoconferencing are available to all developers of both companies, these communication channels play no significant role during development processes. The project leader of Beta was the only person in our sample who used desktop-sharing on a regular basis but still relies on face-to-face meetings for the negotiation of complex tasks.

5.7. Conclusion

The focus on articulation work led to important insights into the needs of SME that engage in globally distributed software development and the effects of offshoring. Our findings suggest that the evolving complexity of articulation work in the context of distributed projects was partly unexpected by the actors beforehand. Articulation work obviously plays an important role for the management of the described offshoring projects: articulation is applied during discussions in which the actors reflexively evolve problems and solutions from their distinct perspectives and work practices. These are then collabo-

ratively distilled into specifications and filed into certain databases or shared documents, the application of which in turn is a matter of articulation work.

The flexibility of SME thus seems to rely intensively upon these articulation practices that are embedded in informal communication, both locally and between the different teams. Thereby, personal meetings in form of regular visits play an important role, as they provide a personal relationship among the actors as well as a shared understanding that may ease written communication processes in later stages of the project. This should be taken into account if companies plan to offshore software development. However, as personal meetings are not always possible and may reduce agility by delaying complex articulation processes, there appears to be still need for a better technical support of articulation work.

Our findings suggest that specialized tools for the support of articulation work are not so common in practice. In our case studies, informal communication through flexible communication channels was used to articulate arising questions like time estimates or important bugs and complementing information stored in formal documents of databases or simple word documents. Our interviews suggest that specialized tools are often perceived as too complex for everyday use. Probably a good strategy for further design would be to build upon the actor's usual tools, integrating plug-ins with enhanced functionality that predefine work practices as little as possible. The growing importance of plug-in based Integrated Development Environments like Eclipse could make this approach feasible.

As expected, actual work practices sometimes differ from the perceptions that managers have about software development strategies in their companies. The ethnographic research of articulation work offers a complementing perspective and shows how the immanent logic of software development processes requires certain adjustments. If this is not considered by researchers, results concerning offshoring can be strongly dominated by a "how things ought to be"-approach instead of centering on actual work practices. The question remains as to how articulation work could be supported more appropriately. Thus, achieving a better understanding of the needs of developers by further field studies seems necessary.

6. Operational and Strategic Learning in Global Software Development[1]

Small to medium enterprises (SMEs) increasingly participate in offshore software development. Key competitive SME abilities include detecting market niches and deploying highly flexible software development approaches. Therefore, learning how offshoring affects such capabilities, which are closely related to organizational learning, is crucial. This article presents case studies from two German companies that engage in offshoring of software development. The authors highlight the different structures these companies have chosen for their development work and discuss how they enact those structures. They also discuss how related practices affect strategic and operational aspects of single- and double-loop learning. The case studies show that organizational learning may be a challenge for SMEs engaged in offshoring software development. Moreover, the inability to perform double-loop learning can even lead to failures during organizational restructuring.

6.1. Introduction

With increasing globalization, distributed software teams have become fairly common. Usually, companies that offshore their software development expect a reduction of costs and access to new markets. However, distributed teams often face problems related to globally distributed work's spatial, temporal, and cultural barriers.

For a long time, studies have mainly treated offshoring as a make-or-buy decision of large companies, offering recommendations and discussing best practices. Although offshoring today is increasingly understood as a dynamic process, few studies address its related long-term implications for key organizational capabilities, especially regarding small to medium enterprises (SMEs), which comprise 98 percent of the German software industry

[1]This chapter has been published as an article in the Journal *IEEE Software* 2010 [32]. © 2010 IEEE. Reprinted, with permission, from Alexander Boden, Bernhard Nett, Volker Wulf, Operational and Strategic Learning in Global Software Development – Implications from two Offshoring Case Studies in Small Enterprises, IEEE Software 27(6), 2010. See appendix I.

[23]. SMEs consider offering highly customized software solutions and adapting quickly to changing customer demands to be among their most important competitive capabilities [78]. Hence, offshoring must be organized in a way that lets SMEs flexibly adapt to changing demands, requiring ongoing operational and strategic learning.

In this article, we contribute to the understanding of learning in the context of offshoring by presenting an ethnographic field study of two small German software enterprises engaging in software offshoring in Russia. This comparison illustrates some of the challenges in organizational learning that SMEs engaged in offshore software development may face.

6.2. Single- and Double-Loop Learning

In this article, we explore how SMEs of the German software industry enact organizational learning. Our work complements that of Wanda Orlikowski on studying "useful practices" of self-organization for organizational development in the context of distributed teams (see the "Related Work on Organizational Learning" box) [166].

In order to conceptualize organizational development, we refer to Chris Argyris and his colleagues' framework [9], which states that it's possible to identify learning when comparing the consequences of actions with the expectations that guided the planning of those actions. In their view, learning entails two different layers that must be covered. Single-loop learning refers to the operational level (Are we doing things right?). Double-loop learning comprises learning at the strategic level (Are we doing the right things?). Thus, enterprises can derive learning from decisions dedicated to operational or strategic aspects of cooperative work.

In the case of offshore partners, reflecting on operational and strategic decisions can be particularly challenging. Although learning in local teams often occurs implicitly, distributed actors might have to adopt explicit strategies to organize their knowledge exchange. This is especially important for SMEs for which flexibility is essential. Because SMEs often highly depend on agile development methods, exhaustive communication and flexible interaction, implementing organizational learning can be difficult for them in the context of offshoring [175].

In this context, Pamela Hinds and Cathleen McGrath have highlighted the role of emerging informal hierarchies for smooth coordination of distributed teams [110]. However, offshoring projects do not per se lead to efficient informal hierarchies and smoothly coordinated cooperation. So, the question remains as to what offshore partners can actively

Related Work on Organizational Learning

Recent studies have increasingly focused on operational aspects of distributed cooperation [144], but there are still few studies that address long-term consequences on key organizational capabilities [126]—not to mention the learning necessary to secure them, for example, in small to medium enterprises (SMEs).

Software engineers have discussed learning as a major issue for software development in the context of knowledge management. In the early years of these discussions, researchers believed technology-intensive systems could accumulate and automatically provide knowledge on demand. Learning was reduced to input/output processing, and knowledge was conceptualized merely in its explicit form [22].

However, although it's possible to represent knowledge in the form of explicit content, such content can actually become knowledge only when it's contextualized. Furthermore, in practice, knowledge must be framed to contribute to the expertise needed [226]. Learning, therefore, can't be reduced to data storing in the brain, but requires understanding in a much broader sense—for example, opportunities to develop practical competences and expertise [172]. Hence, learners shouldn't be considered as mere consumers, but as decisive actors who develop cooperative activities of their own [56]—a transition that some researchers have called the "second wave of knowledge management" [120].

Based on this new understanding of learning, researchers have elaborated the paradigm of self-organization as a means and an end in computer-mediated education, and also in relation to organizational development of software companies. In this context, Wanda Orlikowski has hinted at "knowing in practice" as an important element for organizational operation [166]. She illustrates how a distributed organization enacted and (re-)constituted knowledge through several practices (such as sharing identity, interacting face to face, aligning efforts, learning by doing, and supporting participation). She then argues that, rather than hypothetically constructing formal, decontextualized "best practice" models, research should empirically identify "useful practices." Our work complements Orlikowski's work by focusing on how SMEs in the German software industry enact organizational learning.

Articulation Work

Sociologist Anselm Strauss introduced the concept of articulation work to analyze the interdependent actions of cooperating actors. Articulation work is necessary for regulating the division of labor: who does what, when, where, how, with which quality, and so forth. Yet, articulation work is a broader, more holistic concept than coordination. Whereas the latter governs only the planned distribution of labor (in the sense of distributing responsibilities), articulation work also manages unexpected preconditions and consquences that emerge because of not fully controllable circumstances.

Hence, articulation work comprises important aspects of self-organization and its integration into the formal distribution of work, thus enabling a broad understanding of cooperative work in complex environments such as offshoring projects.

For further reading on articulation work, see the following:

K. Schmidt and L. Bannon, "Taking CSCW Seriously: Supporting Articulation Work," Computer Supported Cooperative Work (CSCW): An Int'l J., vol. 1, no. 1, 1992, pp. 7–40.

A.L. Strauss, "The Articulation of Project Work: An Organizational Process," The Sociological Quarterly, vol. 29, no. 2, 1988, pp. 163–178.

do to secure their software development's agility, and if there is any opportunity to use organizational learning for this purpose. As a related conceptual framework, we can use the prescriptive model of Argyris and his colleagues as a descriptive one, trying to identify opportunities for related "useful practices" (as suggested by Orlikowski [166]).

For doing so, we adopted Anselm Strauss's conception of articulation work (see the "Articulation Work" box) [219]. Such work can contribute to learning at the operational level because it underpins formal divisions of labor by informal, flexible adjustments. However, it can also contribute to learning at the strategic level when collaborative actors reflect on their articulation work, relating it to shared experiences and discussing possible solutions [29].

Hence, by analyzing articulation work's role within offshore software development of SMEs, we can better understand the impact of offshoring on these companies' learning opportunities and practices. Our goal is to understand the opportunities that offshoring presents for different types of learning, including organizational learning, which not only requires learning but also affords the possibility of institutionalizing its results.

6.3. Research Methods

We've conducted our study in several phases since 2006. After an initial literature study, we conducted semistructured interviews with 13 managers and developers of German SMEs: two interviews with representatives of an IT industry association and a large German company, and four interviews with Eastern-European offshoring vendors. We used the interviews' preliminary results to identify offshoring challenges and the strategies that German SMEs use to deal with those challenges. From this sample, we chose two companies for further analysis, which we will call Alpha and Beta.

For the next data collection phase, we drew on a triangulation of ethnographic research methods comprising interviews, on-site observation, and artifact analysis. We conducted the on-site observation by visiting each German SME for a period of 12 business days. In addition, we visited the Russian partner company Alpha for one week. To understand the perspective of Beta's partner company, we also conducted an interview with the Russian manager in Saint Petersburg. Since the end of 2008, we've continued our study through an action research approach at Alpha.

Our data analysis was based on Anselm Strauss' Grounded Theory [223]. After each step, we scrutinized and coded the material's transcripts. At first, we composed categories

(such as knowledge exchange, informal coordination, and formal workflow) on the basis of the collected data. Then, we related those categories to one another, and evolved them during our further research.

6.4. The Case Studies

We chose two companies that expressed very different perceptions concerning strategies for successful offshore software development. Both companies had several years of experience with offshore development in Russia.

6.4.1. Company Alpha

Alpha provides data-processing products and services in the field of statistics and documentation. Most of the company's approximately 20 employees are software developers. The product line comprises databases, documentation, and presentation systems used by archives and museums.

Since the late 1990s, the company has employed four software developers in Tomsk, Siberia. The business relationship began with an internship of a Russian developer who still works for the company. Because of the positive experiences with him, the German manager decided to expand this business relationship. The first project aimed at reengineering an existing standard software product. Despite unexpected delays in development, offshoring was expanded to several smaller customer-specific projects, which involved closer cooperation between German project managers and offshore developers. During the interview in the first phase of our study, the German owner underlined the company's reliance on flat hierarchies and flexible self-organized work. The company would consider formal work processes and development models if the customer insisted on them, but, from the company's perspective, they aren't necessary. Instead, Alpha emphasizes informal and flexible work practices, allowing project managers to run their projects with great levels of autonomy. For handling specifications, the company relies on plain-text documents, which it sends to developers by email or, in some cases, through a defect-tracking system.

6.4.2. Company Beta

Beta offers a standard software solution (developed in two different branches) for process modeling and related services in the field of process management. The management

is located in Bonn, Germany. Four offshore developers in Saint Petersburg carry out the software development under the supervision of a German project manager in Berlin. Another seven employees at the Berlin office provide testing and support.

According to the German manager, the main reason for the offshoring decision was development cost reduction. Through personal contacts with a Russian developer, the company founded a branch office in Saint Petersburg in 2002. The kick-off took place in Germany, where the new Russian employees stayed for a few months. After their return, they took over software development. Since then, the Russian team has grown to 15 developers, which has required certain adjustments in the formal division of labor.

In contrast to Alpha, Beta's CEO perceived successful offshoring of software development as closely connected to a high maturity of the company's development processes on the basis of the Capability Maturity Model (CMM). Hence, Beta relies on clearly defined business processes with explicit responsibilities and standardized development routines. This was also reflected in Beta's use of a central-development database with standardized descriptions of features to be developed, which Beta updated regularly during the development process. Recently, Beta terminated its business relationship with the Russian company because of increasing development costs and ongoing problems with this relationship.

6.5. Different Work-Organization Models

The four models in Figures 6.1 and 6.2 provide an overview over the different approaches chosen by Alpha and Beta to organize their offshore software development. The models in Figures 6.1a and 6.1b describe the patterns of cooperation implemented by Alpha. The models in Figures 6.2a and 6.2b represent the adaptation Beta had to conduct to address the emerging necessities of their offshoring project.

The models represent different divisions of labor, resulting workflows, and interrelationships between the cooperating teams. They involve the exchange of artifacts such as plans, specification documents, bug descriptions, and source code or prototypes. The arrows in the models indicate the artifacts' transfer direction and are usually embedded in articulation work. The two-pointed arrows indicate exchange processes in close cooperation; the dashed lines represent the barriers between the local and remote teams. Regarding our analysis, the models can serve as hints to the necessities of learning in practice. On the one hand, these practices relate to the different tasks the teams must

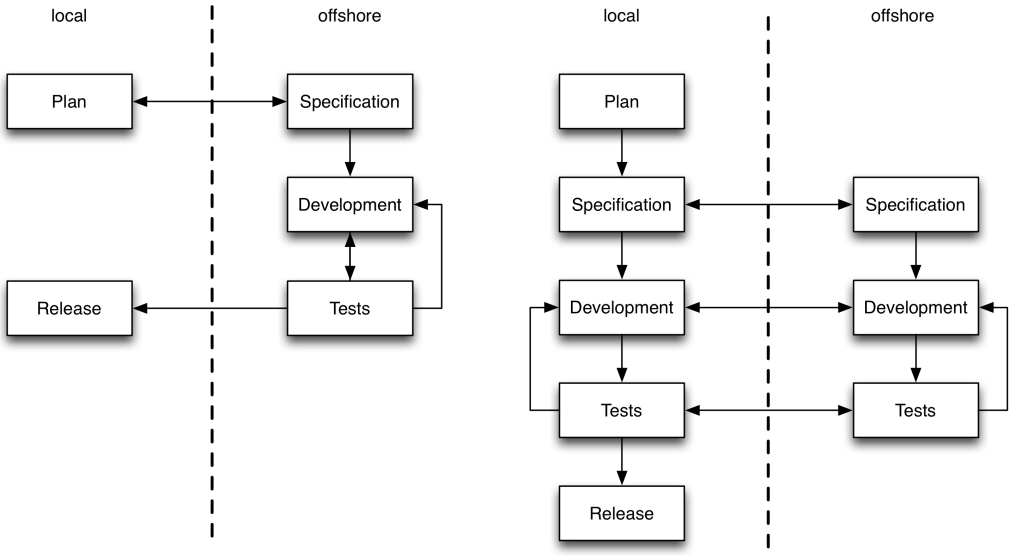

Figure 6.1.: Offshoring model representing the division of labor for company Alpha's (a) standard software solution and (b) customer-specific projects. Both models attempt to keep the core of the software development work integrated, allowing for agile, iterative proceedings on each site.

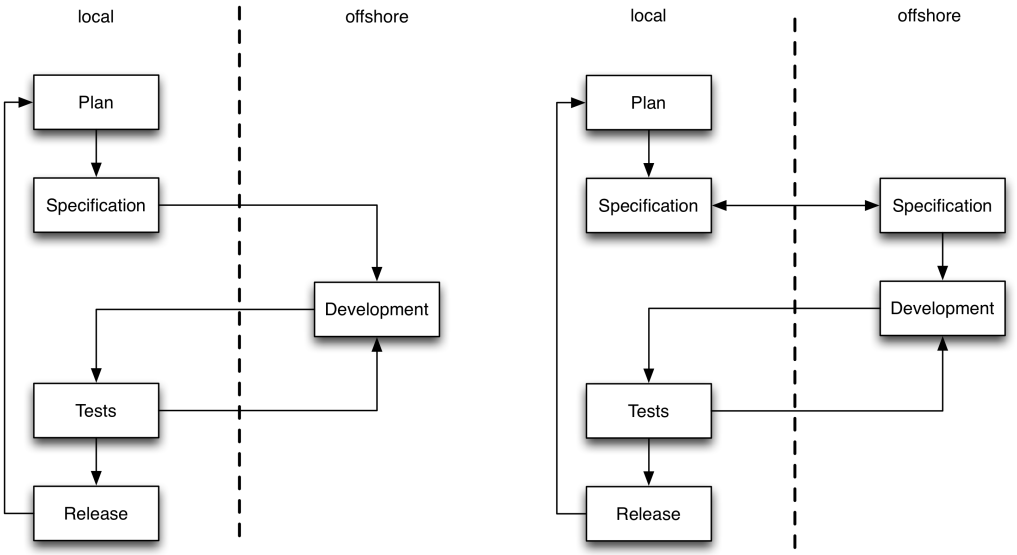

Figure 6.2.: Offshoring model representing company Beta's (a) initial division of labor and (b) division of labor after its reorganization. Both models are based on a high specialization of the cooperating teams, dividing the activities of planning/controlling and the actual development between the sites.

accomplish during their development work—for example, in learning about a new product that is to be developed in operational terms (single-loop learning) or about how experiences with product development can be implemented in the organization of the development work in strategic terms (double-loop learning). On the other hand, we were especially interested in offshoring-specific learning challenges. Hence, arrows crossing the dashed lines in Figures 6.1 and 6.2 are the focus of our analysis.

6.5.1. Model 1: Division of Labor for Alpha's Standard Software Solution

Alpha's offshoring began as a reengineering of an outdated legacy product. The corresponding model (Figure 6.1a) is rather simple, entailing intersite connections mainly concerning the project plan being transferred to the Russian team, which in turn was to deliver the reengineered product back to Germany.

Regarding coordination and learning, the model shows a clear distinction between the teams, almost resembling a customer-vendor relationship. One of the main challenges of software projects—developing and understanding the specifications—was rather easy to handle in this case because the existing product was only to be reengineered. This could easily occur fully offshore, avoiding most of the need for intersite cooperation. However, this also required communication between the German company's customers and the Russian developers in cases of bug reports or feature requests. Because the Russian team had ample opportunities for self-organization (as long as it kept up with deadlines and requirements), it was able to bring in their own expertise to make operational decisions—for example, concerning the choice of development tools, the documentation of the development, and the distribution of tasks.

Although there was related space for single-loop learning, the company's strategic project planning and formal coordination (double-loop learning) relied on regular personal visits of the German manager at the offshore site in Tomsk, as well as visits of the Russian team to Germany for strategic workshops (centering on topics like which technology to choose for the next version, developing a rough project roadmap, and so on).

6.5.2. Model 2: Division of Labor for Alpha's Customer-Specific Projects

In light of the positive experiences, the business relationship was expanded to several smaller customer-specific projects. These were mainly led by German project managers, who directly cooperated with Russian developers, with the help of the Russian senior

developer. Thus, the corresponding model in Figure 6.1b contains many interrelationships between the sites, relating to the cooperative handling of specifications and code development. Regarding articulation work and learning, these projects had a rather informal structure. The project work's initial articulation mainly took place in the course of personal meetings between the staff of the different sites, during which the developers conjointly developed specifications and discussed the project plans for the development work in both operational and strategic terms. It was apparent that the German team members valued the Russian team members' technical knowledge, and they involved them in the specification of products. Because of the direct contact between the Russian teams and the customers (see Figure 6.1a), the Russian team learned about the German user domains. This eased the handling of complex model projects (see Figure 6.1b) involving project planning, development, and testing being performed in close cooperation between the geographically dispersed sites. At the same time, the flat hierarchies allowed the Russian team to influence the project work's trajectories and introduce its own ideas.

Alpha's Offshoring Approach

Based on the different models of work organization in Alpha, it's apparent that both models in Figure 6.1 attempt to keep the core of software production integrated, and thus not separate the responsibilities according to phases such as specification, development, and testing. On the contrary, these three elements are held together, allowing for agile, iterative proceedings.

Alpha's strategy of offshoring was thus aimed at a replication of its own organizational structure (adhocracy) at the offshore site. Hence, in-house activities of everyday software development, such as features specification, code development, and testing, involved close cooperation between the sites.

Ongoing articulation work played a pivotal role for the accomplishment of everyday work because the division of tasks was always negotiated in an ad hoc way between the teams. Alpha exploited the potential benefits of specialization, but at a very low rate. Learning depended on considerable articulation work, which mainly remained focused on limited, situated problems at single- and double-loop learning. Therefore, Alpha couldn't institutionalize organizational learning because the company didn't even consider making structural changes to the established adhocracy.

6.5.3. Model 3: Initial Division of Labor for Beta

The development of the standard software solution of Beta followed a fixed release cycle of six months. Initially, Beta introduced the formal offshoring model shown in Figure 6.2a, which aimed at offshoring the development work to Russia, while keeping all other tasks (such as the definition of new features and the description and classification of bugs) in Germany. Hence, the interdependency between the two teams merely involved the exchange of specifications to be written by the German team and the software code to be implemented by the Russian team, which in turn was to be tested in Germany.

Regarding coordination and control, Beta tried to apply a far more single-sided approach. The key for this practice was the preparation of exhaustive specification documents for the Russian team, which included ample information concerning the interrelations with other software modules, the interface design, expected behavior, and so on. The Russian team, in turn, documented its progress in monthly reports and reviewed its code on a regular basis for quality assurance. In addition, the German project manager visited the offshore team on a regular basis, usually shortly before new releases. During these visits, the German project manager mainly helped in the handling of bugs (usually discovered at the last minute) and—if time allowed—discussed the features of the following release with the Russian team. Strategic questions (related to double-loop learning) were mainly discussed in Germany.

From the German team's perspective, the main challenge involved writing the exhaustive specification documents for the Russian developers. As the Russian team grew, this task became harder and harder because, according to the German team manager, "one day of development required one day of writing specifications." As it became increasingly more difficult to specify new features quickly enough to keep the growing offshore team busy, German management decided to change the formal division of labor (see Figure 6.2b).

6.5.4. Model 4: Division of Labor for Beta after Reorganization

After the restructuring, the interconnections between the teams became far more complex than initially intended. Because the Russian developers now had to write most of the specifications themselves, the German team was able to reduce its work overhead significantly. On the other hand, writing specifications demanded the exchange of necessary context knowledge and thus more articulation work between the teams.

In practice, the articulation work turned out to be difficult. Because the Russian developers lacked most of the necessary context information about practical usage (for instance,

the customer demands) and the product's technical background (such as interdependencies with certain modules), they found writing proper specifications difficult. This led to frustration on both sides. The German team members were discontented with the Russian specification documents' quality and had to assist and correct their work, requiring considerable time for articulation work. Meanwhile, the Russian team members felt overstrained and fulfilled their new tasks only reluctantly.

In an attempt to improve the specifications and reduce the need for ample articulation work, Beta introduced an even higher standardization level. By providing standardized examples and checklists, which were intended to help the Russian developers with their tasks, the company expected to reduce articulation work (visible in the amount of communication) and ensure the produced documentation's quality. However, because the underlying problem of lacking knowledge couldn't be solved easily, most of the problems prevailed and led to increasing difficulties with the Russian developers, who started to neglect inconvenient tasks (such as writing specifications) whenever possible. The ongoing problems contributed to the decision to terminate the business relationship in 2008 (although it must be stressed that several reasons contributed to this decision, including rapidly rising wages in Saint Petersburg).

Beta's Offshoring Approach

Based on Beta's work organization models, it's apparent that it followed a fundamentally different approach than Alpha. Rather than replicating its adhocracy at the offshore site, the company aimed at an ambitious division of labor that resembled specialization between the two sites. Hence, the German site concentrated on planning and controlling activities, while the offshore site, with its lower wages, exclusively performed the actual development work.

Because the mode of operation turned out to be problematic due to the high amount of necessary articulation work (exceeding the benefits of specialization), the company decided to introduce another (in a way, even higher) formalization level to reduce the demands of articulation work. In contrast to Alpha, Beta accepted neither more frequent meetings nor more intense communication between the sites.

Thus, Beta didn't relinquish its demand to exploit specialization's benefits. Instead, it tried to reorganize specialization in a formal, top-down manner. When the amount of necessary articulation work exceeded its ambitious expectations, management reacted with structural changes of its specialization model. Because innovative product finding

wasn't among the measures taken into account in this regard, the changes would resemble single-loop learning—if there would have been any organizational learning at all. In fact, the company valued only the manager's learning; feedback from the Russian team wasn't considered.

6.6. Discussion

Our case studies show that articulation work was demanding for both companies. The practices we found in the field were similar to the ones Orlikowski described in her study [166]. For example, intense face-to-face contacts, broad participation in meetings, and alignment of efforts were all important factors for the two teams (see Table 6.3). However, there were also differences related to the companies' attempts to deal with those practices and to the different types of products the companies were developing.

Whereas Alpha accepted its small customer-oriented projects' related articulation work by intensifying the personal visits and introducing workshops between the sites, Beta tried to reduce this work by increasing formalization and specialization of these tasks in the development of its standardized product. Beta's strategy turned out to be problematic because it didn't account for the necessary mutual learning regarding important context information and domain knowledge or the necessary contact with the customer.

In the case of Alpha, it became apparent that the discussions covered aspects such as the formulation of specifications (What does the customer need?), possible technical solutions (How can we build that?), and some strategic questions (Can we reuse something we already have?). In general, the focus was on operational aspects of cooperation, but basic questions such as the organization of work were barely covered, if at all. Because the company didn't change its adhocracy for dealing with the offshore situation, but rather replicated it, there was no need to broach the issue of the formal structure. In so far as double-loop learning remained limited, it didn't allow for restructuring—a major domain for organizational learning [9].

Beta, on the other hand, did engage in restructuring, but it was reduced to a single-sided top-down decision by the German management, and it included neither the expertise of the Russian side nor that of the developers. Both companies' inability to implement structural changes may be related to the particular work practices of small software enterprises [78]. The practices of articulation work, as well as the models of cooperation we observed, seem to be highly specific for small software companies. SMEs often work

Model	Operational decisions	Strategic decisions	Example of articulation work between sites
Figure 6.1a	Taken by the offshore developers (task assignment, development tools, and so on).	Taken by the Russian team manager in cooperation with the German manager (system framework, deadlines, and so on).	Russian developers visit Germany to discuss strategic questions (which technology to choose, rough project roadmap, and so on) with German developers under the supervision of the German manager. Offshore team mainly operates on its own afterward.
Figure 6.1b	Negotiated between the German project managers and the offshore developers (task assignment, development tools, and so on).	Taken by the German project managers, in consultation with the offshore developers (system framework, deadlines, and so on).	German project manager visits the team in Tomsk, Siberia, to explain his vision for a new project. Requirements and project plan are specified cooperatively during several meetings. Ongoing chats, and sometimes even prolonged visits, continue when necessary to coordinate subsequent development.
Figure 6.2a	Taken by the Russian team manager (for instance, task assignment) and by the German developers (for instance, bug assignment).	Taken by the German project manager (deadlines, specifications, and so on)	German project manager writes specifications. Russian team manager assigns them as tasks to the Russian developers, and German team tests the results. German project manager visits regularly before new versions are finished.
Figure 6.2b	Taken by the Russian team manager (for instance, task assignment) and by the German developers (for instance, bug assignment).	Taken by the German project manager, but partially worked out by the Russian developers under supervision of the German team (specifications).	German project manager explains development aims to Russian developers during his personal visits. Offshore developers must write specifications, which the German project manager in turn checks. This requires much supervision because Russian developers lack the knowledge to write proper specifications. Forms are introduced for this purpose.

Table 6.3.: Different kinds of learning and articulation work for the models shown in Figures 6.1 and 6.2.

in flexible ways and usually can't afford to engage in excessive specialization or institutionalized self-reflection [29]—with the possible consequences we found in our study.

6.7. Conclusion

According to the framework of Argyris and his colleagues, double-loop learning should be a pivotal competency for organizations, especially in volatile and dynamic environments like the software market. However, our case studies show the difficulties that small enterprises may face in developing their organizational structure when engaging in offshoring. The comparison of these two case studies also makes it apparent that restructuring the offshore relation in an inappropriate way (as in case of Beta) can be worse than sticking to pure adhocracy. Under the given circumstances, Beta's attempt to reach a high-level specialization on the basis of process maturity turned out to be less successful compared to the less ambitious approach of Alpha, which didn't even try to change its structure. From our perspective, this is an interesting finding, because researchers often discuss offshoring strategies in relation to the benefits of restructuring an organization.

Our case studies show that organizational learning can be challenging for offshore business relationships. As with Alpha, decisions can be based on distributed, situated experiences such as in organizational learning when sticking to adhocracy and thus avoiding structural change—a major potential of organizational learning. When, in contrast, a company (such as Beta) makes decisions without fully taking into account the expertise of the offshore partners, there is a high risk of failure. In both cases, offshoring can endanger a company's agility. This shows that further research on opportunities for organizational learning that fit the demands of SMEs remains an important task.

7. Trust and Social Capital: Revisiting an Offshoring Failure Story of a Small German Software Company[1]

While work organization and social capital are known to be important factors for offshoring success, there is little empirical evidence on how these aspects evolve in the course of offshoring projects. In the literature, trust has been discussed as a personal disposition to abstain from control in a given situation, and was found to remain surprisingly stable in some cases. By analyzing the relation between control and trust in the course of a failed offshoring project, we want to add to the discussion on social capital as a factor for successful offshoring. The results of our long-term ethnographic study are somewhat paradox: in our case, ongoing conflicts motivated attempts to strengthen control, although personal trust and social capital remained strong. Despite the fact that the confidence of the partners in their offshoring project was weakened over time, the trust among the partners prevailed. However, social capital was not only unable to save the offshoring project—it also seemed to hinder the conflict resolution in some regards. Therefore, we argue that while social capital is an important factor, it should not be regarded as a context-free asset, but rather (in Bourdieu's perspective) as a risky investment.

7.1. Introduction

With ongoing globalization, offshore software development has become quite common. For instance, consulting agencies promote Global Software Engineering (GSE) as a means to reduce costs and as a driver for process improvements in case activities are reengineered

[1]This chapter has been published as full paper in the proceedings of the *Eleventh Europaean Conference on Computer Supported Cooperative Work* in Vienna 2010 [31]. Proceedings of the Eleventh Europaean Conference on Computer Supported Cooperative Work (ECSCW), 2009, pp. 123-142, Trust and Social Capital: Revisiting an Offshoring Failure Story of a Small German Software Company, Alexander Boden, Bernhard Nett, Volker Wulf, Springer London 2009, with kind permission of Springer Science and Business Media. See appendix I.

and streamlined as part of the move. However, while wage differences may offer options to reduce costs, the spatial, temporal and cultural issues in globally distributed cooperative work are still challenges and need to be better understood [44, 99, 107].

Tackling these issues, it is often argued that GSE needs formalization of processes and a high level of social capital to be successful [144]. Features such as formalization and social capital accumulation may—to a certain degree—be influenced when establishing the offshore cooperation. However, there has been little empirical evidence on this topic for later stages of offshore cooperation [126]. This is even more astonishing as these factors are likely to affect flexibility which is regarded as a major demand for software development in general (and especially for small enterprises). Therefore, we need to learn more about how the relationship among clients and vendors evolves within offshoring projects and which factors contribute to or oppose efficient cooperation [75].

CSCW has a long tradition of researching problems of distributed cooperation. For example, CSCW studies have expounded the importance of awareness, tool appropriation, self-organization, behavior, interaction and communication in different kinds of work groups by means of ethnographic studies. However, there are very few in-depth studies which look at the particularities of cooperative work in off-shored software projects—specifically when small companies are involved.

In order to add to the understanding of offshoring, we conducted a long-term and in-depth ethnographic case study in a small German company between 2006 and 2008. During this time, the company developed software in an international team of German and Russian developers. In the end, the cooperation was terminated due to ongoing problems. By revisiting this failure case and the related conflicts over a longer period of time, we offer a complementary view compared to studies on best-practices. We investigate whether trust and social capital changed or remained stable over time in the offshoring project, and how these factors affected the offshoring relationship between the involved teams. In order to provide a detailed analysis, we investigated articulation work [221] conducted in the offshoring project.

The paper starts with a discussion of offshoring literature (section 7.2) before it describes the research method applied in the ethnographic study (section 7.3). The description of the case (section 7.4) is followed by a discussion under the perspective of articulation work (7.5). It turns out that trust and social capital indeed share some similarities, but are no guarantee for successful offshoring. Related findings are explained in the final chapter (section 7.6).

7.2. Offshore Cooperation in the Literature

Apart from challenges which are typical for any software development, GSE projects have to be conducted under particular organizational, cultural, spatial, temporal and legal conditions which can pose complex obstacles [108, 175]. For example, temporal differences can lead to bottlenecks in regard to time for collaboration and coordination, while cultural differences can lead to mutual misunderstandings. As spatial distribution can harden or even constrain possibilities for control considerably, it often affects the necessary level of loyalty and trust among collaborators.

Hence, trust and social capital have been pointed out to be key factors for tackling challenges of distributed team cooperation [110, 144, 186]. Trust has been characterized as a complex, multi-layered concept, which is—amongst others—related to expectations, experiences, and knowledge (e.g. is the trustee competent? Is his behavior predictable? Is he good-willing? Is he opportunistic?) [122]. For our case, trust can be interpreted as a psychological state which allows for greater levels of self-organization, and for an abandoning of (available) control mechanisms [253]. In a similar fashion, social capital refers to network ties of "goodwill, mutual support, shared language, shared norms, social trust and a sense of mutual obligation that people can derive value from" [120]. As "social glue" holding together communities, social capital is expected to promote cooperative behavior in communities and organizations [174, 45].

According to this optimist view, organizations (or teams/communities within organizations) with high levels of social capital will have a higher motivation to cooperate [119]. However, with regard to offshoring, social capital can be difficult to be fostered [110, 48]: as teams are distributed spatially, face-to-face contacts are usually reduced to few limited timeframes. At the same time, relying on ICT for cooperation implies a higher risk of misunderstandings [165, 21], especially in cross-cultural teams. For the same reasons, conflicts can be very difficult to handle, if they occur [111].

As a consequence, recent studies have stressed the importance of initial perceptions of trustworthiness for long-term relationships of international teams [141]. Inter-personal trust, once established, was found to remain more or less stable in the course of distributed projects, at least in case of cross-functional teams [253]. However, it is still not clear if the same rule applies to homogeneous fields with uni-functional dyads, such as in software-offshoring projects, where developers usually should be able to assess the development work done by the other team more easily as compared to cross-functional teams.

By revisiting the failure case story of a small German software company, we want to analyze if social capital shares the detected self-preserving effects of inter-personal trust relation, or if it may at least benefit from them. As trust can only be understood within a particular context [253], it is necessary to take a situated perspective in analyzing the actual work practices, the distribution of labor as well as the related formal regulations and conflicts in offshore relations. In this context, the case of a failure story can be interesting if it allows for the differentiation between continuities and discontinuities.

However, continuities within formal regulations neither guarantee their factual continuity, nor does changing regulations guarantee factual discontinuity, as in reality established patterns may prevail under new labels, or formal regulations may fail. Hence, it is the practical organization of collaboration, not merely the mental models of its organizers, which has to be taken into account. As we were interested in long-term dynamics of offshoring software development, a methodological focus was needed that covers all above-mentioned aspects. This made us adopt the concept of articulation work, as we will point out in the following section.

7.3. Methodology

The concept of articulation work was introduced by the sociologist Anselm Strauss for the analysis of interdependent actions of cooperating actors [221]. Articulation work, similar to coordination, is needed to regulate the division of labor: it centers on decision-making regarding who is supposed to do what, when, where, how, with which quality, etc.? To a certain degree, everybody involved in collaborative work has to reflect not only about his/her work, but also about its organization. In this regard, articulation work is also related to trust and social capital, as it entails issues of trading formal control versus flexible self-organization [30].

Generally, coordination is seen as the organization of collaborative work. However, not everything which is necessary for collaboration is explicitly discussed and regulated as coordination, and often the organization of work is more complex than perceived by many actors. For instance, collaboration may need meetings or discussions between developers, the management, and the customers, but it may also include the administration of a program for a certain task (setting up a related infrastructure), fixing a broken server, or implementing a communication infrastructure for collective work organization—aspects which are seldom interpreted as coordination.

In this regard, the concept of articulation work aims at including all necessary (meta-)work to make work work. Hence, it offers a more holistic understanding of cooperative work than concepts of coordination: while the latter usually govern the distribution of tasks and responsibilities, articulation work includes formal and informal coordination mechanisms [201] as well as related meta-work, which the actors themselves are sometimes not even aware of [211].

As a consequence, it might not always be clear what should be regarded as meta-work or coordination when it comes to particular efforts. What one actor sees as necessary in this regard does not need to be the same as the perception of another. The same is true for scientific observers, who are influenced by their perception of the case. In this regard, coordination may be understood as the explicit model resulting from self-organization, and meta-work as the related practical conclusions, both of them being dependent on cognition and practical interpretation. In contrast, articulation work is the amount of all related contributions, strategies and conflicts; it is the distributed agency of collaboration, not its result.

Articulation work takes the individual perceptions about coordination neither as per se correct descriptions of the distribution of labor, nor as pure illusions; instead, they are understood as necessary points of departures for related analyses. Explicit (coordination) and practical premises (meta-work) of collaboration are regarded as important challenges for the individual positioning of actors within an anticipated field of opportunities. By contrasting conceptions of collaborators with each other and by analyzing empirical evidence on collaboration practices and their outcomes, articulation work studies attempt to take the interests of the collaborators seriously by discussing them retrospectively against the background of all accessible knowledge about the collaboration and its impacts.

This kind of analysis allows the address of the differences between the lived (factual) and the planned, explicit organization [9]. The latter is generally more "logical" (at least at first glance) than the lived organization, which in contrast generally responds to situated particularities in a more complex way. This duality of formalized and informal organization has been discussed within the CSCW community [226] for a long time and led to a much broader understanding of cooperative work in this community when compared to hierarchical models of coordination [200, 71].

In order to study articulation work, we did not only have to look at efforts of coordination and meta-work, but also had to analyze them by contrasting the anticipated logic of the process with what we observed as the factual one. Revisiting the history of a cooperative project over several years with our theoretic stances is difficult, and requires

careful examination. Unfortunately, our access to the company was limited to particular timeframes, and we had to reconstruct (and interpret) parts of the case study by relying to narrative interviews with the involved actors. However, we tried to overcome the limitations of our approach by a triangulation of several ethnographic research methods, comprising semi-structured and narrative interviews, participant observation as well as artifact analysis.

In order to understand the logic of offshoring strategies, we started by collecting related conceptions in the literature. Furthermore, we conducted a semi-structured interview with the German manager of the company in 2006 that centered on his general perception of offshoring, as well as on his particular offshoring strategy.

For the investigation of articulation work practices, we drew on participant observations which were conducted by visiting the German SME several times during 2007 and early 2008. The first two observations lasted one week each and focused on local and distributed software development practices in individual and collaborative work situations and tasks. We were also allowed to analyze artifacts such as emails, chat protocols, internal work papers and whiteboard sketches, and we conducted many informal interviews with developers and the German project manager during our stay. The findings were documented by means of field notes and photos, which were taken during the research.

Our analysis of the collected data was based on Strauss' and Corbin's Grounded Theory [223]. After each step, the transcripts of the material, both field notes as well as interview data, were scrutinized. Data was coded during a process that consisted of several stages. At first, we composed categories based on the findings in the collected data. Then these categories were related to each other and evolved during the further research. These categories were analyzed under the presented articulation-work perspective. First, we attempted to differentiate between formal work organization (taken from the interviews) and the factual work practices we had observed. Then, we tried to identify converging and different perceptions of the offshoring project, as well as reconstructing related interests on the basis of a careful examination of our data.

As a further step, we refined the results of our analysis by conducting extensive narrative interviews with the German project manager during a third on-site visit, as well as a Skype interview with the Russian team manager.

7.4. The Case Study

The offshore software development project we researched was conducted by a German SME. The company offers a standard software solution for process modeling as well as services in the field of business process management. Being part of a holding, the management and sales of the company were handled by an office in Bonn with seven employees. The holding had several other offices, for example in Hamburg (data processing) and Düsseldorf (holding-management). Additionally, about 200 business consultants worked as freelancers in close cooperation with the company.

The software development we studied was carried out by an office in Berlin with seven employees. Apart from the development, the team in Berlin was accountable for the customer support as well as the management of the offshore cooperation with the Russian partner company. This cooperation had been started in 2002, when the German software office (at that time not yet integrated into the holding) had decided to found an offshore branch office in Saint Petersburg in order to reduce development costs.

The decision to locate the branch to Russia was based on a personal friendship of the German entrepreneur with a Russian developer who, according to the German manager, was trusted to be a competent and loyal team manager. This developer had been employed and ordered to hire three further developers in Saint Petersburg. The whole team was invited to Germany in order to become acquainted with the code base of the company. The team was able to take on the leading role of software development after a couple of months.

The German team manager described how the development of a formal model of work distribution marked the beginning of the cooperation. This model defined different roles and tasks for the teams. It included the role of the (German) project manager, the (Russian) team manager, the (Russian) software developers and the (German) testers. Thus, the German project manager wanted to oversee the development of the offshore team directly. In disciplinary or legal matters, the local team manager could be involved.

The German team concentrated on quality assurance, which involved the helpdesk for customers, the testing of the developed code and the strategic planning of upcoming versions. Thus, the definition of new features in terms of specifications and the description and classification of newly discovered bugs were under the responsibility of the German project manager and his team, while the offshore branch was responsible for the execution of the development. In the daily work, the results of the tests, descriptions of new features customers had asked for or bugs that were encountered by the helpdesk team and similar

Tool	Type	Used for
SourceSafe	Version Control System	Managing source-code
Product database	Lotus Notes database	Administrating specifications and releases; tracking progress of work
Development database	Lotus Notes database	Sharing templates for specifications, bug reports and formal work conventions
Sametime	Lotus Notes plug-in	Communicating via Instant Messages; sharing screens
SQA	Bug tracking system	Administrating and tracking bugs
Borland / Eclipse	IDE	Working on code

Table 7.1.: Tools provided for cooperation.

information would be communicated to the offshore branch for investigation. This usually involved personal visits of the German project manager to the offshore site shortly before new releases. During these visits, the German project manager helped handle the bugs (usually discovered in the last minute) and discussed the features of the following release with the Russians. The Russians in turn were to document their progresses in terms of monthly reports and review their code on a regular basis for quality assurance.

In the cooperation, members of both teams relied on several tools, which included a shared code repository (SourceSafe, situated in Germany) and IDEs/compilers (for C/C++, Java and Visual Basic), a bug tracking system as well as a product and development database based on Lotus Notes. For daily communication, a Lotus Notes plug-in called "Sametime" provided instant-messaging and screen-sharing functionality. Sametime allowed for the integration into the Lotus Notes environment and for encrypted communication and recording of screen-sharing sessions for later reviews (see table 7.1).

7.4.1. Changes to the Division of Labor

According to the German team manager, the quick growth of the offshore team soon required certain adjustments of the formal division of labor. As he explained, it had become increasingly difficult to specify new features quickly enough to keep the growing offshore team busy—especially, when the number of Russian developers had exceeded the size of the German team. As the German project manager put it: "One day of development required one day of writing specifications" (Field notes, March 11, 2008).

As it became harder and harder for the German team to keep up with their work, the decision was made to change the formal division of labor. The Russian developers were now to write the specifications themselves, which were then in turn checked by the German team. According to a German team member, this decision was also based on the high competency of the Russian developers, who were trusted to have a deep understanding of the technical feasibility since they were in charge of the development. This change allowed the German team to reduce its work significantly and to enable the further growth of the offshore team which soon reached a size of up to 15 developers, as the German project manager reported.

However, delegating the requirement-engineering tasks to the offshore team led to significant problems, as the Russians lacked the necessary context knowledge: "The required information is very detailed: what does the user interface look like, what conflicts may prevail with other features and what are special cases etc." (Field notes, March 11, 2008). As the project manager pointed out:

> "Knowledge concerning the practical usage and the technical background has to be combined in a creative way in order to find a solution. There is a difference between requirements specifications [considering the context-of-use] and design specifications, [being limited to the technical background]. The Russians tended to produce the latter" (Field notes, March 12, 2008).

According to the project manager, the problem was exacerbated by the Russian team's poor English skills. While only rudimentary English skills would be needed for the coordination and control of already defined tasks, the definition of new features or the transfer of context information would be much more complex, thus sometimes exceeding the skills of the Russian colleagues. The German project manager explained: "The chats took much time and it was very difficult to transfer the related knowledge. It is easy to assign tasks or take over results, but it is hard to explain what needs to be done" (Field notes, March 11, 2008).

On the other hand, the Russians also reported problems concerning this way of cooperation:

> "People [from the German team] (...) had no time to review them [the specifications], so the developers started to work without acceptance of specifications. (...) [So the] specifications did not follow the real implementation, or it took too much time for writing specifications" (Interview, May 28, 2008).

7.4.2. Attempts of Standardization

Faced with severe problems of communication and knowledge transfer, the company introduced a higher level of standardization to their documentation. Thus, standardized forms for documents, conventions for bug descriptions, source code comments and specific languages were developed. By providing examples and checklists, seen as help for the Russian developers with their tasks, the company expected to reduce the amount of communication and to ensure the quality of the produced documentation. The related documents were stored in the development database.

However, the complexity of writing specifications in combination with the missing background knowledge still made the tasks difficult and inconvenient for the Russian team, as the German project manager explained: "[The Russians] lacked the understanding of the program and the context of its use and the work is very unattractive, as it is very challenging and not well supported by tools" (Field notes, March 12, 2008). In addition, the German team reported increasing difficulties with the offshore developers, who started to ignore tasks that were recognized by both teams as being unpleasant and annoying. This mainly included the writing of documentation and specifications as well as the tracking of the work with log-files.

> "The Germans introduced forms to the product database in which the Russians should have entered their tasks with the expected beginning and end. They did this, but only at the beginning of the planning stage. As everything is very complex and unexpected dependencies occur, it is impossible to anticipate everything. Thus data needs to be updated regularly, but the Russians did not do so" (Field notes, April 12, 2008).

The following excerpt of a conversation illustrates this problem. The dialogue was taken from the chat-log of an online meeting between the German project manager and one of the Russian developers. The initiator of the online-meeting was the project manager who wanted an overview of the developer's tasks. Using Sametime, the project manager was able to take control of the mouse and screen of the Russian developer and test the newly implemented features in this way. The inspection was accompanied by a chat discussion and took nearly three hours. The subjects of the discussion were the tasks (mainly feature specifications) in the product database, which were worked off feature by feature:

> "Project manager: I know we spoke MANY times about it... (...) it is impossible for me to follow progress if you don't write comments! so please don't let me repeat it again :-((...)
>
> Developer: I don't understand what should I write here, the implementation is fulfilled in 100 %
>
> Project manager: let me show you how i do it in my tasks.
>
> On the shared screen, he [the German project manager] shows Dmitry some comments he has written. He opens one of his tasks, where he has already noted his progress like in a diary. The comments hint at problems he encountered, and at discussions with the developers. Then, he opens the product database and starts to comment on another task:
>
> - started implementation
>
> - bss [abbrev. name of another developer] send me new idea, so i stopped implementation. see info above.
>
> Project manager: you can decide the details in the comments. you should however add info that may be useful to you and to other people. this may help you keep notes on tasks (instead of using paper :-)) or for example when you stop a task or need to restart it after some time.... you can use this to remember what you have to do. in any case whenever you update STATUS, PROGRESS or DATES.... then you should add a comment regarding the reasons of the update.
>
> Developer: o.k." (Field notes, July 11, 2007).[2]

Similar discussions concerned the conduction of internal code reviews of the offshore team as well as other examples of missing documentation.

7.4.3. Selling the Offshore Organization

According to the project manager, the problems with motivation were exacerbated when the decision was taken to sell the branch office to the Russian team manager in 2004. This decision was related to ongoing problems with the cooperation, as the Russian team manager reported:

> "When we started [we were] four people (...). [In] 2004, all of these four developers left the company, because they were not satisfied with the situation.

[2]Spelling and accentuation are taken from the original chatlog.

> And from my side I wasn't able to do anything, to keep them (...). Because I had always to discuss any small question with Berlin" (Interview, May 29, 2008).

Furthermore, the decision was related to the challenges of handling the complex legal and organizational requirements of running an offshore branch. The communication with the local authorities turned out to be a serious and permanent challenge for the small German company, having no previous experience with Russian law. Hence, by changing the status of the Russian partner to that of an independent company, the German entrepreneur hoped to avoid many of the legal problems of managing an international company.

Thus, in 2004 the decision was made to continue the work by means of a contract between the SME and a now legally independent Russian company. As the Russian team manager explained, the Russians were quite happy with this change: "After we started to work as an independent company it got much easier for me to take decisions (...). And before, it took long discussions with Germany about why it was required" (Interview, May 29, 2008).

However, according to the German project manager, this change had dramatic consequences for the international cooperation. The Russian team manager, now being the proprietor instead of the employee, started to expand his company and look for new customers in order to reduce his dependence on the German SME.

In the interview, the German team manager described this strategy as expectable and even understandable. However, there were also unforeseen consequences, as it became much more difficult to continue the cooperation when the Russian team manager increasingly reduced his commitment to the cooperation. As finding new business partners became the main goal of the Russian partner, the German developers again were unable to control the Russian developers, who (from the perspective of the Germans) lacked discipline.

Even worse for them, according to the German project manager, the Russian team manager had been the most experienced and trusted team member abroad (especially since so many others had left the company), and his change of interest led to severe problems, as the other Russian developers were unable to perform his duties with the same professional standard:

> "[The German project manager] was unhappy that [the Russian team manager] was not available as a developer anymore from one day to another. As he was the manager instead of the developer now, the relationship had changed:

> instead of giving orders, everything was subject to negotiation" (Field notes, April 12, 2008).

In this regard, the dependency on the Russian developers made it difficult for the German team to enforce a reasonable accomplishment of tasks (especially of inconvenient ones) by the Russian team. "[The Russian developer] agreed to change his behavior, but he did not do it. And the Germans apparently were unable to convince him" (Field notes, April 12, 2008).

7.4.4. Salaries and Infrastructure

The problems with the offshore developers hit the company at a disadvantageous point of time. In 2006, the German company had been taken over by a holding. At the same time, according to the German project manager, the development costs had almost tripled compared to the situation in 2002. As both sides reported, the level of the salaries was an ongoing field of conflicts between the sides. As the Russian team manager explained:

> "Finally they realized that they paid much more than they expected. (...) Salaries grew up too much in Saint Petersburg, and (...) I think, currently it makes not big sense to outsource from Germany to (...) Saint Petersburg. Because prices are comparable. (...) [And I told them] I was not ready to continue our contract on these terms" (Interview, May 28, 2008).

Because of the poor performance in combination with the rising development costs, the holding decided to reduce the size of the offshore team to eight—a decision, which further increased the frustration of the offshore team, as the German project manager reported.

According to him, the reduction in the number of employees belonging to the offshore team made it easier to coordinate the shared development, but the financial problems prevailed. He explained that this was due to the growing importance of Saint Petersburg as a software region. Western companies were in search for offshore developers, and the job market was growing rapidly. The lack of social security (sometimes seen as an argument for the attractiveness of a country) made income the only security for employees, and thus contributed to increased salaries. Policies of the German SME to keep salaries low were a constant field of conflict in the offshoring cooperation. At the same time, the small team size made the company especially vulnerable to fluctuation of team members, while the low level of specialization required extensive training of new developers.

In this context, the German team reported that the Russians tried to use their influence on the development. Conflicts started about the distribution of (inconvenient) tasks and the technical infrastructure. For example, instead of using Sametime for their communication, the Russians started using Google Talk, and instead of using the company's Lotus Notes Database for shared documents, the Russians switched to Google Docs for their daily work.

Furthermore, the Russian team decided to stop using the shared bug-database SQA in favor of a self-developed database in 2007:

> "The management of the company wants to get monthly reports concerning the ratio of feature development against the fixing of bugs. The tools [SQA and the product database] distinguish between both kinds, but it is not possible to (...) create an automated report, which the Russians find annoying. Therefore, they plan to administrate features and bugs in a shared database and have begun to develop their own, web based solution" (Field notes, July 10, 2007).

As a result, the teams had to track bugs in two parallel systems, because the German company was reluctant to change their established infrastructure. On the other hand, the German team manager did not want to antagonize the Russian team:

> "Basically, [the German project manager] likes the idea [of a shared system], but the report feature is not necessary because they only need rough estimates for the taxes. But to avoid decreasing the motivation of the Russians they let them do as they like, as long it does not involve more work for the company" (Field notes, July 10, 2007).

Therefore, he did not intervene, but his acceptance was based on the condition that the Russians took on the necessary overhead work of maintaining two systems. In addition, the Russians planned to develop an import/export filter for the automatic synchronization of the two databases.

The other changes of development tools, i.e. using Google Talk instead of Sametime and Google Docs instead of Lotus Notes, were justified mainly with the available resources of the developers' computers. Since Sametime, according to the developers, needed much processor time and memory, it was annoying for the Russians to do their everyday work. Using the web-based Google Talk would be much more convenient for them. From the perspective of the German project manager, the decision had another reason. According

to him, the Russians wanted to keep up to date with the tools they used. Thus Sametime and Lotus Notes would not be as trendy as the newer Google tools.

7.4.5. The Termination of the Cooperation

In 2007, the size of the offshore team was further reduced to four. Finally, in early 2008 the German holding decided to stop the cooperation completely, first by reducing the team size to two, and then by suddenly stopping the offshoring by the end of the month. The decision itself had neither been unexpected nor was it unwelcome by the German partners:

> "All in all, everyone was unsatisfied with the state of affairs. The Russians, because the holding paid unpunctually, the developers in Berlin, because bad work was delivered, and the holding, because everything was considered as being too expensive, and the prices were increasing further" (Field notes, March 12, 2008).

Accordingly, both teams had considered the possibility of terminating the cooperation, and the German project manager had made up a plan together with the Russian team manager which was meant to arrange this termination to be as smooth and easy as possible for both teams. According to the project manager, this was not only due to his own team's interests, but also due to the personal friendship with the Russian team manager. In this regard, both teams said they would have liked to continue the cooperation under different circumstances, and they blamed the holding management as being the one responsible for the failing of the project.

Hence, in the end, only the abruptness of the decision caught both teams by surprise. As the Russian team manager explained:

> "In the middle of December, [the German holding] said, o.k., please keep these four developers until end of May (...). So we will have five months to move the development from Saint Petersburg to Germany. (...) But [then] they said that they had changed their decision and needed only two people until the end of February. This was unexpected (...) and I had to pay salaries for them and even (...) fire one developer" (Interview, May 29, 2008).

7.5. Analysis of Articulation Work and Social Capital

While the last chapter recapitulated the course of events from the perspective of the practitioners, we will now revisit the offshoring story from an articulation work and social capital perspective.

As we were told, the initial phase of the cooperation was supported by a high level of trust between the teams, which was based on the friendship between the German entrepreneur and the Russian team manager. Furthermore, the visit of the whole Russian team to Germany had helped to form social ties between the developers, too. However, despite this high level of social capital, the German team wanted to stay in control of the development as much as possible, as software development was still deemed as the core competency of the company. The Russians, on the other hand, accepted this distribution of tasks, as it allowed them to concentrate on the technical side of the development only.

However, in order to do so they were dependent on exhaustive specifications of features which the Germans found increasingly difficult to afford. As the initial distribution of labor turned out to be problematic, the German company had to learn that writing complete specifications (even for the standard software product) can be as time-consuming as the development itself (or even more). Instead of being self-explanatory and efficient, the disjunction of requirements-engineering and coding led to severe coordination problems which were caused by the necessary knowledge transfer and articulation work between the teams.

As the workload of the German team increased, the decision was taken to change the distribution of work while ensuring that control remained with the German team. The Russians accepted this change unwillingly. Despite their good technical knowledge, the Russians had difficulties with the task of writing specifications. As they lacked the necessary context knowledge, the effort of writing adequate documentation was very high, even more as they could not draw upon shared business experience with the customer. As a result, the Russians felt overstrained, and the amount of necessary requests, clarifications, and corrections increased—classical aspects of articulation work. At the same time, the dependency on ICT for articulation work created bottlenecks, which were further aggravated due to language issues between the teams. While the trust between the teams was still high, the German team attempted to improve the documentation by introducing standardized forms for specifications and bug descriptions. However, this attempt to support the Russian team in writing specifications did not work, as the necessary knowledge exchange was still insufficient. In contrast: the efforts to formalize the

development turned out to be only new forms for informal articulation work, and for related uncertainties in the development process.

Despite the related increase of informal communication—which has been found to support knowledge exchange and even conflict resolution in distributed teams [111]—the company could not benefit from the change, because the Russians lacked the necessary context information which was paramount to successfully accomplish the task of writing proper specifications. After all, communication needs to support the underlying work structures of a team [110]. Instead of supporting the necessary communication work, the management of the holding—bound by the necessity to coordinate two organizations—reacted by intensifying control and formalization. This in turn was seen as an escalation and systematization of attempts to blame the Russians for the prevailing problems. The social capital which had formed the basis of the commitment of the Russian colleagues started to become eroded—despite the initial high level of trust between the sites, which rested on the personal relationship between the German project manager and the Russian team manager.

The Russians, unable to meet the expectations of the German team, began to neglect certain tasks which were regarded as being annoying and unnecessary, like tracking the progress of their work. The German company had to realize that it was dependent on the commitment of the Russian team and that formal methods of control cannot guarantee personal obligations—or even damage them [122]. Even worse, the German team was unable to solve this problem. In this phase of the cooperation, the still high level of social capital apparently hindered an open argument between the teams. The German management avoided blaming the Russians outrightly for not fulfilling their tasks, while the Russians avoided arguments with the German side by simply ignoring inconvenient tasks. In this regard, social capital apparently became a trap: the German manager understood the anger of the Russian team, but regarded the current division of labor as necessary. The Russians, on the other hand, accepted the decisions of the Germans, but felt unable to work under these conditions.

As more and more of the Russian developers left the company, the decision was taken to sell the offshore organization to the Russian team manager. While this decision was approved by both sides, it became the origin of further emerging conflicts, as the cooperation with the now legally independent Russian enterprise made it impossible for the Germans to use hierarchy in order to maintain their idea to substitute informal demands of articulation work by means of intensified formalization. The loss of competent developers was a significant drawback for the company, not only in terms of knowledge, but also

in terms of social capital. While the initial cooperation had rested on the personal ties which were formed during the extended personal visit of the Russian team to Germany, the newly hired developers could not benefit from such relations. Instead, the social ties between the teams were mainly focused on the Russian team manager, who shifted his focus to acquire new customers instead of concentrating on the existing cooperation. As a result, the problems with the motivation of the Russian developers aggravated, as social capital as a means for motivating cooperation between the German team and the new Russian developers was weak.

The German team—discontent with the development of the cooperation with Russia—felt trapped: since none of the teams was able to work efficiently without the other team, but every team had the possibility to jam shared projects (by ignoring or by misinterpreting cooperation demands), successful cooperation became unlikely. Collaborative demands on articulation work—considered to be substitutable by formalization and control by the holding management—emerged again on each level of conflict resolution and turned out additional strategic options for constraining the cooperation afterwards.

In this regard, the company also suffered from hard-to-anticipate indirect effects, as, for example, the rising salaries in the region of Saint Petersburg, which were partly connected to the boom of investments in the area and contributed to cost the inefficiency of the cooperation and its termination. Moreover, the change of infrastructure on behalf of the Russians was hard to foresee. The German team accepted these changes within certain limits because they feared to further discourage the offshore team members.

Apparently, the management of the holding overestimated the possibilities of formal control, and neglected conflict dynamics and social capital issues. As a result, conflicts manifested when coordination necessities emerged on the basis of inter-dependencies in the work constellation which could not be settled by controlling and formalizing the software development. When people tried to solve the problems by means of formalizing articulation work, the situation did not improve, but deteriorated—and was further aggravated by the structural circumstances like rising costs, decreasing social capital and the organizational consequences of the divestment of the branch office.

7.6. Conclusion

Our case study illustrates the endeavors of a small German enterprise to keep its offshoring project running. Looking at the related failure story from a long-term perspective, complex and inter-related conflicts within the field of articulation work become

visible. The German management, for instance, tried to take advantage of the relatively low wage levels of the Russian partners whom they sought to control by determining their work (by means of the division of labor and tool usages).

The Russian partners turned this claim around by showing that, if control was that important, there was a lack of it in the whole collaboration. How could the Russians work well if the requirement delivered to them were not controlled (for instance, if they complied with international standards)? The counter-reaction of the German management was, again, a turn-around: if the requirements were that difficult to handle, the Russians should write them themselves.

This shows that decisions were taken to shift responsibilities between the offshoring partners. Therefore, articulation work obviously was not only a contingent dimension of decision making, but (in the given case) even attributed to a history of its own in regard of work regulations. The related ping-pong effect of control-based arguments shows that both partners shared related convictions or, at least, did not want to question them. Hence, the changes to the work arrangements were partly the result of continued shared convictions about the necessities of control and formalization. But those were not the only continuities.

While it became apparent that any new regulation led to new areas of conflict, it has to be noted that this did not diminish the mutual appreciation among the actors. Their mutual trust remained through all these conflicts. But what about social capital? It was defined before as "network ties of goodwill, mutual support, shared language, shared norms, social trust, and a sense of mutual obligation that people can derive value from" [120]. Have these ties declined throughout the diverse conflicts? The astonishing fact is that partners from both sides still would have liked to collaborate even after the termination of the offshoring project which was seen as a salvation on both sides.

Obviously the actors differentiated between the personality of their partners and the offshoring situation as a whole [122]. This implies that the management partly understood the strategies of the Russians to take advantage of higher salaries, or at least did not see it as personally insulting. In contrast, the Russians obviously understood the role of the holding as a limiting factor for the German project manager. Insofar, the trust—with regard to mutual goodwill—among the actors prevailed even through disappointing collaborative experiences regarding opportunistic and sometimes unpredictable behavior. The same was true for social capital in the mentioned sense as an accumulative value.

However, the social capital, which contributed to motivation at the start of the offshoring project, also turned out to be a hindrance at its end, as the assumed knowledge about

the personalities of the partners made it easier to detect structural limitations of the situation, and as it apparently hindered an open argument about the prevailing conflicts. This means that social capital can really be an asset in the sense that collaboration would not be possible without it. Nevertheless, it can become dysfunctional, a mis-investment in terms of the capital metaphor. Social capital is not only about cognition, inter-action, and shared perceptions: it also relates to fallible investment of efforts.

This fallible characteristic of social capital is not covered in the necessary detail by Putnam's tradition of social capital as "goodwill, mutual support, shared language, shared norms, social trust, and a sense of mutual obligation". Hence, it seems to be fruitful to expand the given understanding of social capital by referring to Bourdieu's [36] notion of social capital as a means to reconstruct the risky decisions of individuals when attempting to establish profitable value chains, which can explain why social capital apparently can change from an asset to a hindrance.

In relation to offshoring, it was found that there seems to be something like a tendency of trust to prevail [253]. We came to similar results for social capital, but our results also question the concept of social capital as a merely positive factor for global software engineering. Like for trust, it seems we need a much more differentiated understanding of social capital in the context of GSE. Without social capital, GSE as a complex form of distributed collaboration will hardly be possible. On the other hand, formalization and social capital are no guarantee for successful performance. As we have seen, impacts of the international environment and contingencies of articulation work make it very likely in GSE that a given arrangement changes quickly. Therefore, it seems to be a major challenge for GSE to develop forms of making articulation work reflexive and operative, for example, through globally distributed organizational learning.

8. Knowledge sharing practices and the impact of cultural factors: reflections on two case studies of offshoring in SME[1]

The impact of culture on knowledge management in international teams is an important topic which is still not well understood. We contribute to the discussion by presenting two case studies of small software teams involved in distributed software development. In doing so, we illustrate how cultural and social issues influence the way knowledge exchange is performed by analyzing four knowledge sharing practices: status meetings and maintaining awareness, the collaborative use of shared artifacts and repositories, spending time at the other site and human "bridges" that mediate between people and cultures. Our findings suggest that organizational culture is permanently re-negotiated and adjusted to fit the distributed collaboration, as the teams learn how to deal with each other. Socialization plays a significant role in this learning process, and people are more likely to draw on national stereotypes when breakdowns occur. The influences of national culture and site-specific organizational culture are subtle and not easy to separate from other factors. Based on our experience, we argue that in order to achieve an accurate understanding of knowledge sharing practices in globally distributed software teams, these need to be studied in context, longitudinally, and from both the onshore and offshore perspectives.

[1]This chapter has been published as an article in the Journal *Software Maintenance and Evolution: Research and Practice* in 2010 [26]. Reprinted, with permission, from Alexander Boden, Gabriela Avram, Liam Bannon and Volker Wulf, Knowledge Sharing Practices and the Impact of Cultural Factors: Lessons from Two Case Studies of Offshoring in SME, Software Maintenance and Evolution: Research and Practice, 2010, Copyright © 2010 John Wiley & Sons, Ltd. See appendix I.

8.1. Introduction

The ongoing economic incentives and pressures of globalization have led to the growth of distributed international organizations that attempt to take advantage of time differences, varia- tions in labor availability and cost in order to improve competitiveness. The software industry is one of a number of fields that has been influenced by this increase in globalization, leading to a variety of business strategies that attempt to harness the possibilities of, for example, offshoring certain activities to lower-cost sites, developing globally distributed software teams that can work on projects 24/7, etc. As software and source code can be transferred easily between globally distributed sites, offshoring of software development has been widely seen as a means for cost reduction and efficiency gains. However, international teams have to cope with a multiplicity of organizational, temporal, spatial, legal, national and cultural barriers, which can affect the devel- opment pace and the quality of the software. A growing literature base exists on the delineation of these problems, and attempts to resolve them, in the context of the emerging sub-field of software engineering known as global software engineering (GSE). Problems of communication, coordina- tion and collaboration in the software domain are also studied in such fields as information systems (IS), computer supported cooperative work (CSCW) and knowledge management (KM). However, there are still many unresolved issues concerning the management of such globally distributed development efforts.

In this paper, we provide a contribution to this research field through documenting and analyzing some of the knowledge-sharing practices and communications of distributed international teams of software developers. We make a number of observations concerning the problems encountered. Our approach is one that focuses on the work practices of the different software teams, and the many and varied ways in which they manage to accomplish their work, despite the difficulties of communicating across languages, cultures, time and distance. Along the way, we critique certain GSE approaches to the study of these distributed groups that overly rely on what we believe to be limited conceptions of such key concepts as culture and knowledge.

With regard to knowledge, there has been a focus on KM approaches for understanding the organization and behavior of distributed teams by referring to "canonical" concepts of knowledge as a product—suggesting that most knowledge can be de-contextualized and shared explicitly amongst teams relying on databases and ICT [22]. However, this "knowledge as a product" view is questionable. Practice-based approaches and theories of social learning (cf. [232]) suggest that while ICT may be well suited for dealing with explicit knowledge, implicit knowledge cannot be shared out of context. Hence, these

alternative approaches focus on understanding how knowledge is embedded in social relationships and how actors actually share and put their knowledge into practice [1, 163].

One important issue with international teams is the impact of culture—under multiple aspects: national, organizational and professional. While the topic of "culture" is one that has captured the interest of the SE community for some time [118] most of this work has tended to focus on attempts to apply, for instance, Hofstede's [117] work on dimensions of national cultures in what we believe to be problematic ways. There seem to be very few studies dealing with the issue of culture in regard to knowledge sharing work practices in GSE. We contribute to the discussion by presenting two case studies of small-size software teams involved in distributed software development in the context of offshoring. In doing so, we want to illustrate how companies deal with knowledge exchange in practice, and how cultural influences (in a broad sense) affect KM practices, in the particular case of small enterprises.

The paper is organized as follows: after a discussion of the related literature (Section 8.2) we introduce our cases (Section 8.3) as well as our methodology (Section 7.3). Then, we present our findings (Section 8.5) and discuss the data in relation to our research question, as well as the existing literature on this topic (Section 8.6) before concluding (Section 7.6).

8.2. Related Work

In our study, we focus on work practices—with an emphasis on knowledge sharing activities—and we attempt to provide a better understanding of the impact of the social and cultural factors on these activities. As a basis for our discussion, the following sections are meant to provide an overview on our perspective of these concepts.

8.2.1. Knowledge in (Global) Software Engineering

The research literature offers different conceptualizations of the role that knowledge plays in software engineering. While studies in the field of GSE are adopting both technocratic and behavioral approaches, technocratic approaches are clearly dominating the scene [22]. This focus on rather traditional KM concepts is problematic, as it supports a view that considers knowledge as being an object that can be de-contextualized, captured and disseminated "on demand", without any loss of meaning, through information systems.

In contrast, behavioral approaches stress that although information can be represented in the form of explicit content, it needs to be contextualized in order to become knowledge again. Furthermore, knowledge needs to be framed in order to contribute to the expertise needed in practice [226]. Knowledge sharing, therefore, should not be considered as mere consumption of information, but as a complex and reflexive practice of cooperating actors [1]. This is reflected in a broad set of theories claiming that action is situated [226] and deeply connected to tacit knowledge [119], which cannot be made entirely explicit. Huysman and de Wit have labeled this shift of focus toward tacit and emergent aspects of knowledge as the "second wave" of KM [119]. In this socio-technical understanding of KM, the focus moved from setting up canonic knowledge databases to supporting informal knowledge sharing in communities with tools grounded in the practices of the particular fields [120]. Hence, knowledge is rather thought of as being socially embedded, and appropriate strategies take into consideration the practice-related aspects of KM. Orlikowski [166] has hinted at knowing-in-practice as an important element for organizational operation by illustrating how knowledge was enacted and (re-) constituted through several practices in a distributed organization (such as sharing identity, interacting face-to-face, aligning efforts, learning by doing and supporting participation).

While these analytical concepts are concentrated on the various ways of handling knowledge in practice, other perspectives focus on the social structure supporting knowledge sharing. These approaches are interested in how social relationships shape the way in which knowledge is shared in practice. For example, Granovetter [93] has emphasized the role of social connections for the functioning of organizations. Also Marczak et al. [149] have shown the importance of social networks in fostering relationships, trust and KM. Building on these concepts we want to analyze the role of culture for KM practices in international teams.

8.2.2. Cross-Cultural Aspects of Global Software Engineering

Cultural compatibility is often described as an important factor (among others) in determining the success of collaboration in international software development teams [42]. The impact of culture on software development—be it national, organizational or professional culture—is a topic with a long tradition in IS research. The recent spread of global development teams has spurred interest in this topic and led to a broad variety of studies investigating the impact of cultural issues on ICT adoption, use and development [83, 92]—and also to a discussion on the impact of culture on KM practices in GSE [57].

The cultural terms used in the GSE literature often focus on the national aspects of intercultural work [146, 244]. These approaches usually treat culture as being equivalent to national identity, referring to Hofstede's framework of cultural dimensions [117]. Thus, one sees survey studies done on the differences in the communication style between North American engineers and Indian engineers for example, which seem to assume that one can work with such generic categories based on the geographic location or national identity. Usually, such studies attempt to compare national cultures by operationalizing variables such as "power distance", "individualism" or "uncertainty avoidance" which are expected to represent characteristic attitudes shared amongst the citizens of a nation (and/or members of an organization). Within the organizational studies field, Hofstede's formulations have been the subject of extensive conceptual and methodological critique (see, for example [151]). Criticisms of this approach include: (1) Culture is seen as a never changing, monolithic concept; (2) cultural groups are seen as homogeneous, whereas the possibility of diverging subcultures is ignored; and (3) actors are allocated to one culture at a time, whereas different cultures are seen as being mutually exclusive. Without wishing to be drawn into this controversy here, we do urge caution in the use of such self-report survey instruments to investigate globally distributed software teams. The wholesale adoption of this approach by certain software engineering researchers probably has more to do with the relatively straightforward way in which these concepts can be operationalized and data "captured" using easy-to-apply survey instruments, than to any real engagement with the underlying organizational "theory".

Other researchers have developed more nuanced interpretations of the culture concept itself— moving from a focus on the concept as denoting a set of pre-programmed stereotypical behavioral responses, to an understanding of the dynamics of interaction within and across professional, organizational and national boundaries [133, 163, 131]. In accord with such interpretivist approaches, we propose a much broader understanding of culture: we see culture as a shared web of meanings that shapes roles and interpretations, and is dynamically (re)negotiated by the actors in the course of their daily work. Hence, we are more interested in the actors" interpretations and related processes of sense making, than in the definition of cultural particularities [86].

This broad understanding of culture entails many different layers referring to national, professional or religious aspects, which are seen as being intertwined in a complex, non-hierarchic way, and which can hardly be studied in isolation [83]. It also includes many invisible aspects which cannot be studied directly, such as values, beliefs and attitudes. However, it is possible to study culture by referring to its manifestations in the form of

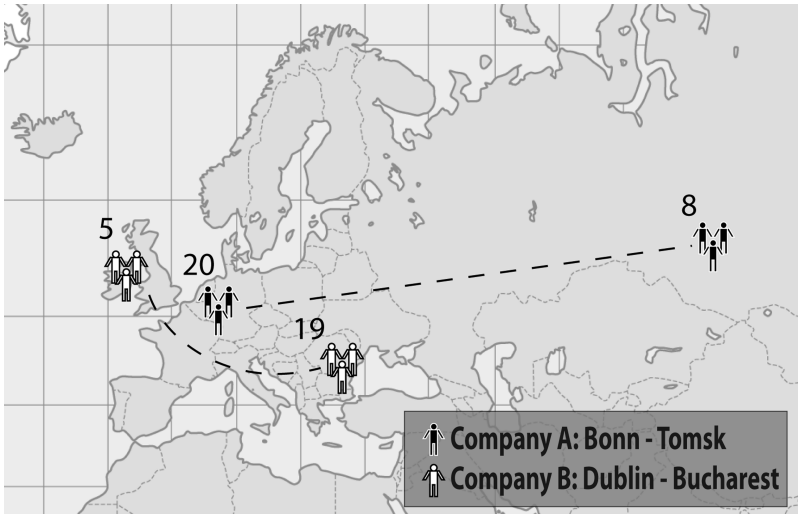

Figure 8.1.: The locations of the teams. Numbers indicate the team sizes. (worldmap by I.K. Sankakukei, http://english.freemap.jp, distributed under a CC-BY 3.0 license).

artifacts, practices and routines (as well as the related interpretations and connotations), which will be at the center of our attention in this paper.

In the following, we adopt this more nuanced interpretive approach to examine two case studies of globally distributed software development. Our aim is to look at what we can learn from practice, with a view to developing our understanding of what kinds of software tools and organizational support mechanisms might be required.

8.3. Cases

In this section we will provide a short overview of the two case studies we have conducted (see figure 8.1).

8.3.1. Company A Overview: Germany (Bonn)—Russia (Tomsk)

Company A is a small German software enterprise engaged in the field of statistics and documen- tation located in Bonn. The customers are mainly German archives and museums. The company was established in 1980 and had approximately 20 employees in 2009. In the mid-1990s, the company found it increasingly difficult to hire German developers, as wages had increased considerably and the labor market had shrunken. Hence, based

on a positive experience with a very talented Russian developer who did an internship with the company, the owner of Company A decided to expand his company to Russia and founded a branch in Tomsk, Siberia. Since then, a number of four to eight employees have been working for Company A in Tomsk, The first project aimed at reengineering an existing product, which had to be rebuilt in C++. Hence, despite considerable delays in the development, offshoring enabled the company to redesign their existing products shifting toward a modern architecture. This created a competitive advantage for the company, which would have been impossible to acquire otherwise, as competent C++ programmers were far too expensive in Germany at that time. As a result, the cooperation with the offshore site was expanded to several small size projects, employing Russian developers—working in close cooperation with a German project manager—for customizing software products to the specific needs of particular customers. The long lasting cooperation (more than 10 years) made Company A an interesting case for our study.

8.3.2. Company B Overview: Ireland (Dublin)—Romania (Bucharest)

Company B was established in January 2006 in Dublin, Ireland. The two owners had worked together before in a company providing software applications for telecoms and media companies for four years. During that period, one of them had been a project manager and the other (originally from Romania) had been working on his team as a senior developer. In January 2006, following changes in the management of their employer, they decided to leave their employment and set up their own company. They hired four developers in Ireland to work on their first project, and they took on project management positions. In an attempt to acquire other customers and expand the company, they tried to recruit more developers in Ireland, but failed due to the harsh competition. Therefore, the Romanian project manager identified a small company with five employees in Bucharest, Romania, which they consequently acquired. The Romanian company is legally independent and incorporated in Romania, but the same two managers (Irish and Romanian) have equal shares in it. In December 2007, there were 19 people working in the company's offices in Romania, and another project manager (besides the Irish founder) in Dublin, with the Romanian manager traveling between Dublin and Bucharest frequently. In January 2009, the number of employees had grown to 26, of which 7 were based in Dublin (including four Romanian developers). Besides managing specific projects, the two managers are actively involved in acquiring new projects internationally. Being an Irish-based company makes them attractive on the international arena: in doing busi-

ness, Irish companies have the reputation of being stable and reliable, and the fact that they have their development division in Romania is a signal for potential customers that the company can offer quality work at a lower price than other competitors.

8.4. Methodology

The two case studies presented in this paper were investigated using similar approaches, relying on ethnographically informed methods and based on an interpretivist paradigm.

8.4.1. Case Study A: Research Methods

The first case study (Company A) started in 2006 and has been conducted in several phases. It aimed at understanding coordination and organizational learning in small-sized, distributed software teams [29, 32]. The contact with the company was initiated during a first phase when interviews with 13 managers and developers of German SMEs, as well as four interviews with people from Eastern-European offshore companies were held. The interviews lasted roughly 1 hour each and were used for identifying the challenges of offshoring for German SMEs, as well as some general strategies used by these companies to deal with them.

The second phase of data collection was performed using ethnographic research methods, comprising interviews, on-site observations and artifact analysis at Company A. The on-site observation involved spending two and a half weeks at the company's headquarters in Germany. In addition, we spent 1 week on the site of the Russian offshore partner. We also conducted 13 extensive interviews with members of the German and Russian teams, including developers, as well as people from the management. The interviews lasted between 1 and 2 h each and were used to analyze the different perspectives on the work practices we identified during our observations. In order to check our findings, we discussed the results of our research with the practitioners during a workshop.

8.4.2. Case Study B: Research Methods

The second case study (Company B) was based on the findings of an exploratory study conducted in 2006 [11]. The 2006 study surveyed six small Romanian software development companies and three freelancers who were involved in outsourcing relationships

as vendors. A number of categories were identified after data coding, and these categories guided our next study. One of the conclusions regarding methodology was that an outsourcing relationship needed to be studied from both ends, in order to get a more objective picture.

A new study was conducted in 2007, after identifying an Irish company with a development unit in Romania-Company B in the current study. This case study focused on the challenges encountered by SMEs involved in outsourcing, with an emphasis on the role of cultural mediators in distributed software development [179]. The methods employed were ethnographically-informed: two visits and observations on both the onshore and the offshore sites, four interviews (with each one of the two managers and two of the Romanian developers), examination of artifacts. A new round of interviews was conducted in January 2009, for getting an update on the company's situation and practices, as well as for validating the analysis and interpretation of the data collected in 2007 from a new angle.

8.4.3. Data Analysis for the Current Study

The frame for this study was a result of a discussion between the authors on their ongoing research at the time, revealing the similarities and differences between the two cases. The data were then analyzed jointly, by taking different aspects of KM from the literature and discussing their manifestation in each of the cases. Based on the discussions, we identified several practices of KM which were relevant to our cases. They will be presented in the following section, followed by a discussion of the cultural aspects entwined with KM practices in these distributed software teams. For describing each practice, we relied on our own observations, on what our informants said during the interviews and on the analysis of artifacts that were made available to us (meeting minutes, chat archives, email exchanges, requirements documentation).

8.5. Research Findings

In this section, we will connect our findings from the two cases, presenting a number of work practices used by the two companies for facilitating knowledge exchange. In line with our own understanding of culture, we will focus on three relevant aspects:

- Practices: How do practitioners share knowledge?
- Interpretations: What is their own perspective on how they are doing it?

- Artifacts: What is the role of the related documents, tools and media in this regard?

Our analysis of the settings is organized around a number of topics which have been shown to be of crucial importance in helping the distributed teams in our two cases to coordinate their work and ensure that the work was accomplished. The challenges introduced by having to work across cultures are also addressed.

8.5.1. Status Meetings and Maintaining Awareness

An important practice we encountered in both cases was that of regular meetings (co-located or distributed) and status information exchanges between sites.

Company A had regular weekly meetings at its German headquarters, in order to give people an overview on what was going on in the company, discuss current developments and problems and share information on new technologies and tools that might be useful for the team as a whole. The offshore team in Tomsk was holding a similar weekly meeting. Starting with 2008, the team members in Tomsk were required to write brief minutes of their weekly meetings (in English) and share them with the German team. As both developers and project managers reported, information shared during the meetings (and in form of the minutes) was not sufficient for keeping up-to-date: "(...) if all I know (...) is that a developer has worked on this or that... this is somehow sparse information". Hence, the developers and project managers explained that they would rather stay aware of what was going on by going around and talking to people—a practice mainly possible in collocated environments. Furthermore, the short references to what was going on in Tomsk were sometimes used as props for direct requests and communication by some of the German developers, but the minutes were not considered as a medium for exchanging knowledge directly.

Company B had a different approach: the two managers started every day with a status review of all their ongoing projects over Skype. They coordinated their activities for the day and divided the tasks. During the workday, they permanently maintained an open communication channel. This practice was probably a result of their long-term collaboration. Mirroring this practice, the Romanian developers working jointly with customer development team members also maintained open channels with their remote counterparts throughout the day. It was also customary for the managers to have regular conversations via instant messenger with each developer in order to get updates on the status of specific tasks.

Status meetings are common practice in GSE; in these two particular cases, the particular organizational cultures and power structures have led to different practices that best suit the two companies.

8.5.2. Collaborative Use of Shared Artifacts and Repositories

For software development, be it local or distributed, knowledge related to the application domain and the aims of the development play a very important role for successful collaboration of devel- opers. Usually, companies use artifacts such as project plans and specification documents as boundary objects for supporting the necessary knowledge exchange and coordination between the customer, the manager and the (in this case distributed) developers [29].

In the case of Company A, specifications were handled in the form of text files, compiled by the German project managers based on the contract with the customer. Before they were sent to the remote site, the assignments in the contract were translated into English and commented on by the German project manager. However, these documents were only used as rough project guidelines during the later development, and were not always maintained up-to-date. Generally, we found very little documentation available about the technologies deployed in both companies. As a Russian developer of Company A explained: "(...) some specifications of features exist in the documentation (...). But documentation—for obvious reasons—never goes into details on how things are implemented. The internal architecture is not documented yet (...)."

Keeping documentation to a minimum is also one of the strategies of Company B, and like in Company A there is a strong reliance on informal communication and direct requests in case of problems. After reaching an initial agreement, the project manager and the developer assigned to the task discuss the requirements with the customer (usually via call conferences) and the developer is asked to write brief specifications in English (double checked afterwards by the project manager and the customer) to be attached to the contract.

In general, our informants reported that the information is fragmented across various databases, e-mails or chat-logs and can be hard to find: "(...) one notices again and again that information is there, but is distributed in a way that makes collating it cumbersome..." (Developer, Company A). Hence, in many circumstances, rather than looking for information in the documentation, people prefer to simply ask local or remote colleagues. The prevailing reliance on communication in the absence of documentation can

lead to breakdown situations when one team needed information from the other site and could not get a prompt answer—as several interviewees from Company A told us. However, the reasons for the coordination breakdowns were assessed quite differently by the different teams.

According to German project managers and developers, the Tomsk developers simply did not like to write documentation. Instead, they preferred to write code considered by them "self explanatory", and not linger with documentation which "would be anyway outdated most of the time". This focus on programming as opposed to other aspects of software engineering work was accompanied by the temptation to redesign existing technical frameworks instead of focusing on the requested features. As a German project manager put it: "All developers are architects-to-be, too. (...) You want to have a car door repainted, and instead you get a new vehicle." This situation seems to be more the result of specialization and division of tasks between project managers, architects and developers than an effect of the geographical distance.

The Russian developers we interviewed had a different perspective on the documentation issue. The Tomsk team leader found that his developers wrote much more documentation than their counterparts in Bonn, who often ignored these tasks. The case of a German project manager who "forgot" to update the specifications with change requests from the customer, making the Russian team work for several weeks on features that had been dropped, was mentioned. Again, this appears to be due to the lack of coordination between the two professional groups rather than due to differences in the national cultures.

8.5.3. Spending Time at the Other Site

Interviews in both companies have revealed that personal face-to-face contact plays a very important role in knowledge exchange, confirming what the previous research on GSE had already shown [165]. Besides building trust and getting a realistic perspective on the skills of remote team members, personal meetings have a significant role in learning how to approach a person from the other site. The face-to-face meetings constitute an important basis for building social ties that can be reinforced later by exchanging informally personal information online (about family events, kids going to college, health issues, etc.—as mentioned by the Irish manager of Company B).

In order to deal with the prevailing communication issues between the teams, Company A supported regular visits of their staff to the Tomsk site and also tried to invite Russian

developers over to Germany for longer periods of time. The motivation for this practice was three-fold: to support mutual enculturation, to support knowledge exchange between sites and to serve as an incentive in preventing Russian developers from leaving the company.

In Company B, brief visits of Romanian developers to customer sites, perceived as direct contact opportunities and marking important phases in the project were systematically organized. During these visits, developers got the chance to gain a better understanding of the environment that their counterparts are working in, to see them at work and learn from their practice. The Irish manager paid regular visits to the Romanian site, while project managers and developers from the customer side also spent time in Bucharest working with the Romanian developers. Several informants told us that once back to their desks from visiting customer sites, they were pro-actively sharing what they had seen and learned with their colleagues.

Generally, these mutual visits were highly appreciated by the developers on both sides. One of the German developers of Company A explained: "I myself have realized that the contact became much better after (some of the Russian developers) have been on site, I would say. Often, especially with regard to technical details or to the design of a user interface, the communication over the Internet was rather slow. And then, when we sat together face to face, and I made a few gestures, and showed what I wanted, the understanding came much quicker (...)". Furthermore, the visits endorse personal contacts between teams, as formal work visits are usually complemented by social activities. For example, one of the German project managers reported that he followed the personal invitation of one of the Russian developers to spend a weekend skiing when he was in Tomsk. Furthermore, during the on-site observation in Tomsk, the German guests were invited to a bowling center during the lunch break by a Russian developer celebrating his birthday. These events played a significant role in socialization, as both teams liked to show their guests around during their stays.

8.5.4. Human "Bridges": Mediating between People and Cultures

both companies, we found people bridging the two cultures who also have notable technical and domain knowledge. They were very important for managing and mediating the communication between the teams, by working with both sides and keeping in constant touch with the other team.

In Company A, several Russian developers living in Germany acted as mediators between the sites. "I am frequently getting requests from (the German manager) or from (the Russian team manager) to improve communication. So, then, what am I doing? I am moving around, asking people about the status of different things, the difficulties in communication, the points where people feel dissatisfied with the other party's work. And then I try to create a kind of neutral technical description of the situation. It worked so far". In regard to his role in the company, he further explained: "I think I became part of the German team—for sure, because my normal working routine involves working here in Bonn with my German colleagues. I have a cultural connection and some psychological connections with the Russian team, of course. It saves a lot of time, effort and emotions that I understand the language, that I can hear their complaints (laughter)". Knowledge exchange between the teams not only required the translation of documents, but also being aware of sensitive issues and personal preferences—requiring excellent social skills. This is a very important task, as a Russian developer (living in Germany at that time for a year already) explained. As an example, he reported about an incident where the Russian team had set up a homepage in a rush, following the request of a potential Russian customer, without consulting the German team. The new homepage included, besides internal technical information (for example IP addresses of internal services like the CVS), copyrighted images "borrowed" from the Internet. The German team demanded the immediate removal of the homepage, but the Russian team seemed to lack any understanding of the legal problems their action could entail. They reacted by asking: "Why are you starting a war on this?" According to the Russian developer living in Germany, this had to do with the prevailing "culture of blame" in Company A (as labeled by our informant), which affected the interpretations of the event in a negative way—and he had many difficulties in moderating the conflicts arising from this incident.

In Company B, the Romanian manager played a key role in running the company; her 7 years spent in Ireland working closely with her Irish counterpart gave her the chance to acquire valuable domain knowledge and business skills, and also have been the basis of the shared understanding they developed. Whether spending time on the Romanian site or traveling to acquire new customers, she had permanently an open channel with the other manager and with the Romanian developers. During the interview, the Irish manager spoke about how collaboration with Romania would have probably been a totally alien idea to him 10 years ago, but having the Romanian manager on his team for 4 years before starting the current company had given him confidence in her skills and commitment, and consequently, in the people she recruited in Romania.

8.6. Discussion

In the following, we will examine the role played by culture in the KM practices described in the previous section. For each, we will examine the impact of culture as reflected in practices, artifacts and routines. As mentioned previously, culture is an elusive concept that can only be studied by referring to its manifestations.

Regarding the first category of practices, *Status meetings and maintaining awareness*, our observations reveal how onshore organizational culture can be extended to the offshore location. In Company A, both sites have their own weekly status meetings, but only the Tomsk site has to write minutes in English and send them to Bonn, the team in Bonn not having to reciprocate. In Company B, the joint practice of the two managers of having status meetings every morning is complemented by the maintenance of open Skype channels throughout the work day between managers, managers and developers, developers and customers. Regarding knowledge exchange, the primary role of meetings and supporting artifacts (meeting minutes) is maintaining awareness.

However, these meetings are also an opportunity for people to get to know each other, as they are used as props for requests and communication between the teams. Thus, they also allow differences to become apparent.

Collaborative use of shared artifacts and repositories: The role of artifacts and repositories is to support collaboration. But team members from onshore and offshore might have conflicting understandings of the role of specific artifacts, the importance to keep them up-to-date, who should contribute, who benefits. The daily exchanges of technical knowledge between onshore and offshore observed were more or less unstructured, highly situated and bound to the emerging work trajectories, for example when unexpected problems occurred, or if changes in one part of the code base affected other modules. The onshore and offshore teams used shared repositories and communication channels, but as Krishna et al. showed [133], this cannot guarantee the success of collaboration in any way.

Regarding specification documents, the practices are quite different in the two companies, reflecting the differences in the organizational cultures regarding customer relationships management. While in Company A, the relationships with the customers are handled exclusively by the German employees, in Company B this responsibility is delegated to the Romanian developers themselves. One interesting detail about this is that the communication between the German and Russian site happens in English—none of the teams actually using English for intra-team communication, whereas in the case of Company

B, the usage of English is prevailing, with developers having to work directly with their international customers. As far as Company A is concerned, the practice has clear consequences for the teams, who have to deal with translations and/or communicate in a foreign language in large parts of their daily work. This also had impact on the organizational culture, concerning the routines of handling specification documents and the mediation of the contact between the Russian developers and the German customers.

Although the case of the Russian team working on obsolete features is an extreme example, it illustrates how particular organizational practices look from each side's perspective. When problems occur, the differences tend to be attributed to the national culture ("Germans are like this, Romanians are like that!"), although many might be due to the specific division of tasks (e.g., business analysts vs developers). As the Russian developers of Company A had no direct contact with the customer, they needed clear and detailed instructions. Their German counterparts, on the other hand, worked under totally different circumstances, and needed to keep a close connection with the customer. Hence, they preferred to work in an agile way, with requirements being subject to ongoing negotiation and change. Existing studies have shown that most developers prefer to write code and not documentation [162, 143], as was also confirmed by the project manager of Company A In a similar fashion, the tendency toward innovation and adding state-of-the-art details as opposed to working on the features stipulated in their contract is the object of another well-documented dispute between developers and project managers, irrespective of nationality [179].

Referring to the practice of *Spending time at the other site*, in Company A, where the visits of Russian developers in Bonn are relatively long term, the purpose is clearly defined: enculturation, knowledge exchange, transfer of particular technical skills to German team members. In Company B, the visits are relatively short, and are meant to give employees the opportunity to see the environment and the people they are collaborating with, getting a direct experience of what it is like to work from there. On the other side, these visits are also useful for perceiving and understanding the differences between sites. The existence of social ties has been shown to improve knowledge transfer and communication in general [131].

The practices of creating and maintaining social ties were closely related to the visits to the other site, but also to private initiatives of developers who befriended colleagues from the other team. These personal ties helped in bridging the distance and resolving problems attributed by our interviewees to the cultural differences.

In connection with the practice we named *Mediating between people and cultures*, we found that social connections and human "bridges" between the teams played a very important role for knowledge exchange (our findings being pretty similar to those of Milewski et al. [153], Cataldo et al. [43], Marczak et al. [149], Krishna et al. [133]). It became apparent that companies in both our cases rely heavily on some key people (called "liaisons", "bridgeheads" "straddlers", "cultural mediators" by different sources), who act naturally as information brokers and conflict medi- ators. The "bridging" roles were mostly informal and taken on voluntarily, with these people understanding the different perspectives of both teams and doing their best to fill in the gaps. Hence, they contributed to the creation of shared understandings. These "bridges" were also very important for the continuous reorganization of the shared practices, and they contributed to the mediation of "team knowledge", found to be very important for efficient teamwork in distributed software development by other studies [70]. More specifically, the knowledge exchange between teams was heavily reliant on the relationships between particular developers who possessed excel- lent social networks in both teams. In order to develop software in distributed teams, compa- nies need to establish shared practices and ways of dealing with the everyday work or what Krishna et al. [133] name a "compromise work culture", that is permanently re-negotiated and adapted.

The practices we described were based mostly on direct or mediated communication between people, and less on tools, repositories and artifacts. Instead of pursuing standardized and formal KM approaches, knowledge exchange amongst practitioners was informally performed, based on triggers found in weekly meeting minutes or development databases—which also served as props for initiating context-related coffee/lunch discussions. This also applied to the knowledge exchange between the teams, where our field studies showed an almost complete absence of traditional KM strategies, and a high reliance on informal communication via instant messenger or face-to-face meetings.

In this regard, it was interesting to see that the practitioners in our study mostly referred to the national differences in breakdown situations, and that the use of cultural stereotypes interpretations was often connected to negative interpretations and mutual criticism. Our observations coincide with the findings of Damian and Zowghi [52], who found that national culture was sometimes used as a "scapegoat" to explain ineffective management practices at various levels. More specifically, in our case studies, national interpretations were often used in terms of (over-simplified) stereotypes from an external point-of-view, when the behavior of the other team was considered to be inappropriate or unprofessional. The actors used stereotypical references to national culture in situa-

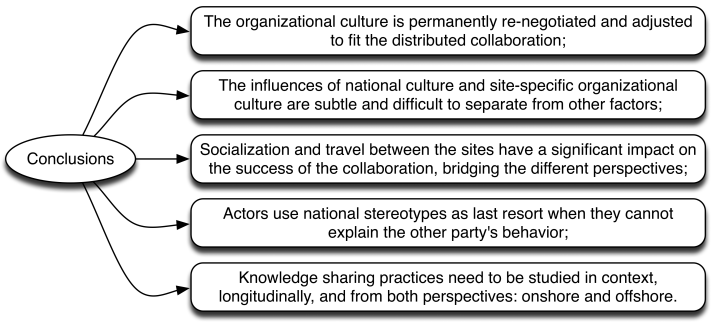

Figure 8.2.: Lessons learned.

tions where professional or organizational explanations did not seem to work anymore, i.e., in conflict situations, emphasizing the "us vs them" delineation between teams. In this regard, cultural interpretations were used to label the others when the collaborating teams lacked shared understandings, generating affirmations such as: "This is how the Russians (or Germans) are like!"

8.7. Conclusion

Our stated aim for this study was to examine what we can learn from these two cases, considered by our informants as successful and long lasting collaborations, despite the occasional misunder- standings and frustrations (see figure 8.2 for an overview).

From what we have seen in our two cases, the organizational culture is permanently re-negotiated and adjusted to fit the distributed collaboration; the influences of national culture and site-specific organizational culture are subtle and not easy to separate from other factors. Socialization and travel between sites, although they come at a cost, have led to good results and the two companies in our study believed that they were important elements of their collaboration. One important conclusion is that like any other collaboration, distributed collaboration is a continuous learning process: teams and people learn, becoming aware of the cultural differences and adapt.

The cases presented in this paper cannot be, by any means, considered representative, and our aim was rather to illustrate the intricacies of knowledge work in a distributed software development setting than to attempt any large-scale generalizations. However, we do wish to point out how a recourse to explanations based on national cultural differences are often, on closer examination, shown to have other rationales which do not draw on such a simple conceptual framework. We have shown how incidents usually attributed to

national culture are mostly due to the incapacity to explain the other party's behavior within one's own reasoning frameworks (Nicholson and Sahay [163] have reported similar findings with regard to knowledge embeddedness).

From a methodological point of view, we insist on the importance of looking at the practices in context, paying attention to the correlated and embedded motivations, maintaining a dynamic perspective. Taking into account both the onshore and offshore perspective is vital for gaining a real understanding of complex situations (as we have seen that the perspectives can significantly differ between the cooperating teams). Our argument here can be stated as follows: in studying distributed software development practices, the concept of culture needs to be addressed, not simply in terms of national stereotypes, but through the examination of diverse aspects of professional and organizational, as well as local cultures. Investigating such issues is a difficult process, and requires an assessment of the strengths and weaknesses of different methodological approaches. What we have attempted to illustrate here is one way of getting into the field to study such issues. No single approach will provide all the answers. However, we believe that the predominance of GSE studies that apply survey-type instruments utilizing forms of Hofstede's dimensions of cultural variation is problematic for the field, as it tends to over shadow the value of other, more interpretive approaches to understanding the myriad ways in which communication and coordination across globally distributed software teams is affected by local, organizational and professional "cultures". It is also worth emphasizing that the application of specific research methods and techniques is not separable from an understanding of the conceptual frameworks within which such instruments have been constructed. One cannot simply take instruments developed within a particular framework and apply them without a solid grounding in that particular approach. There is a danger, within the software engineering field of applying various forms of social science methods and analyzing the "results" without such a grounding (this holds true across-the-board, and not only for studies using Hofstede's framework). This issue requires more extensive deliberation and discussion within the SE research community itself, but it obviously also affects the SE practice, where the interpretation of many research studies in this area needs to be treated with a certain amount of caution, in terms of the generalizability of the findings, and their applicability to particular settings.

Part III.

Conclusions

9. Analysis

Part II has offered a broad view on the role of articulation work in the specific context of distributed software development in two small- to medium-sized German companies. This included the related articulation work practices in the field (chapter 5), the inter-relation of articulation work and learning (chapter 6), and the socio-cultural embedding of articulation work in terms of its relation to soft aspects like social capital (chapter 7) or culture (chapter 8).

This chapter will discuss these different aspects from a unified perspective on the roles and forms of articulation in the specific context of software offshoring in SMEs. The discussion in the first section is divided into three parts (based on our perspective outlined in chapter 3):

- Articulation work in software offshoring of SMEs;
- Articulation work and organizational learning; and
- Articulation work and its socio-cultural embedding.

In the second section, we will then discuss our findings with regard to their :

- Methodological implications for research;
- Theoretical implications for the scientific discussion;
- Practical implications for companies; and their
- Implications for the design of supportive software tools for distributed software development in SMEs.

The chapter closes with a section on open questions and topics for further research.

9.1. Summary of Findings

This dissertation aimed at understanding management practices in distributed software development of two small German companies. In order to understand the management

practices of software development in the context of the daily cooperative work between the distributed software developers, we relied on interviews with the team members that were involved in the inter-site cooperation in both cases (even team members that were only indirectly involved, like secretaries), as well as on on-site observations in the companies. We also tried to involve the perspectives of the offshore developers into our study by conducting interviews with the Russian colleagues and on-site observations in case of Alpha in order to overcome the local perspective which we necessarily had to take during our research (see chapter 4 for a detailed discussion of the challenges we had to tackle in this regard).

9.1.1. Articulation Work in Software Offshoring of SMEs

In general, our research supported the view that software development is a highly situated, creative, and knowledge-intensive activity [225]. During our stay in the companies, there were ample opportunities for us to observe articulation work practices which were necessary for the constant (re-)negotiation of the distributed software development activities in the field [29]. This included personal meetings for the cooperative planning of new projects as well as ad-hoc discussions in case of breakdown situations, but also discussions with regard to recently implemented features. Although the managers of the two companies we chose for our observation—Alpha and Beta—made very different statements during our initial interviews regarding the role of formal coordination protocols (like specification databases and bug-tracking systems) for the successful organization of their offshoring cooperation, we found during our participant observations that situated, informal articulation work was of high importance for the daily conduct of the local and trans-local software development work [30]. For example, in company Alpha project plans were usually handled in form of customer contracts that were stored as Microsoft Word documents. These plans were very specific about the estimated costs and deadlines, but far less specific with regard to the technical details about the implementation. Technical details were usually worked out in the course of personal meetings between the project manager and the involved developers, during which the actors tried to bring their different perspectives together ("What does the customer need?"—"How can we build that?") in order to reflexively discuss aims and problems and evolve related solutions. The results then were written down in form of annotations to the contract document or as personal notes, but they were not stored in any centralized, "official" manner. Afterwards, the created documents were used by the developers as guidelines for their work (in terms of what to develop and when). Reference to them during the

later development by the project leaders was also possible in order to assess the state and status of the project work, or to remind the developers of overdue features. Hence, the documents more or less represented a rough outline of the agreed upon basis for the projects, but their specifics usually were so vague that their fullfillment required regular discussions and re-negotiations during the development phase, especially with regard to the technical design of the product (see chapter 5 for details).

In company Beta, specification documents played a much more important role, as the company tried to maintain exhaustive documentation and provided very elaborated specifications to their Russian colleagues during the first years of the cooperation. While this practice was reported to work well for the developers, the project managers soon found it hard to write detailed specifications quickly enough to keep the growing offshore team busy. Given the time and effort that was necessary for working out (at least almost) complete specifications with all the needed information like possible cross-dependencies, GUI design and so on, the project manager soon recognized that in principle he could also develop the software himself—especially as the Russians still had constant requests on aspects that were ambiguous in the specifications, and articulation work was still needed for example in case of conflicts, customer requests, or technical breakdowns (see chapter 7 for details).

In general, it became apparent that informal articulation work played a very important role for the practices we investigated, as the formal specifications, development models and plans that the actors had worked out to guide the development always needed to be interpreted in order to put them into practice [84]. While this did not mean that everything was negotiated in an ad-hoc fashion, it forced the actors to remain in constant contact and re-negotiate plans in case of apparent gaps or emerging problems (see above). In the specific cases of software offshoring we observed, the different contexts the cooperating teams were working in could lead to problems in this regard as the distribution sometimes caused misunderstandings and deviating interpretations of the tasks (and plans) ahead. These misunderstanding often were difficult to re-adjust or even identify due to the limited inter-team awareness as well as the socio-cultural and practice-related barriers between the teams. For example, in case of company Beta, the Russian colleagues lacked the necessary background knowledge to anticipate the needs of the users. While it was possible to communicate the function of the software by the written documentation, the forms of interaction with the software were much harder to specify. As the Russians lacked the necessary knowledge about how the software was used by the customers, they found it very hard for example to design usable interfaces.

Hence, it turned out to be problematic in both cases to come to a shared understanding regarding priorities and tangible implementations of development tasks.

With regard to the business models of the companies we investigated, their high service orientation further aggravated this effect because it made the software development even more case-specific and dynamic, as frequent change requests and highly specific user domains called for constant re-adjustments and re-negotiations of aims and even of basic conditions of software development work [32] (see chapter 6). As the company aimed to establish long-term relationships with their customers, developers had to be very open towards change and feature requests. These were often handled in the context of service contracts, which also included support for appropriation and tailoring of the software products the company developed. This demand for flexibility was further aggravated as the company followed a kind of unstructured approach towards software development, shifting developers between projects and changing priorities in a very flexible manner. Hence, project work had to be constantly re-articulated as plans and responsibilities were shifting all the time, leading to changes and conflicts (see chapter 5 for details and a comparison of the different approaches the companies followed).

In the case of Alpha, for example, the specific domain of the company demanded a high level of domain knowledge in order to develop solutions that fitted the work practices of their customers: archivists and curators of museums (which both practice specific, highly specialized and historically grown working cultures in Germany). It was very interesting to see that despite the cooperation was performed in English during the personal meetings, the Russian developers used archivist's terms like "Verzeichnungseinheit" or "Bestand" during the discussions with the German project mananger (and also in their notes). As they used these terms without further explanation, it can be concluded that they understood what these term were referrring to, and that the Russians had obtained a general understanding on how a German archive works.

In order to develop innovative and specialized software, the cooperating teams in Germany and Russia had to bridge the distance between the two companies, bringing together their different competencies and perspectives. In this regard, the distribution of responsibilities (German "project managers" vs. Russian "expert technicians") also resulted in a distribution of competencies which required a constant and highly complex knowledge exchange between the teams with regard to finding a match between the usage context of the software to be developed (the domain of the German side) and possible technical solutions (the domain of the Russian side). In order to accomplish a shared understanding, both companies preferred personal visits [29]. For example, in case of

company Alpha the German project managers visited the Russian company two times during our study, while the Russians also visited the German company twice. Furthermore, the German manager visited the offshore site at least once a year, much like in company Beta where the German project manager had a similar schedule (always before a new version of their product was to be released). However, these occasions still were constrained by financial and legal limitations as well as by conflicts with other duties (fixing bugs in the last minute) and personal aspects like the unwillingness to travel on part of some employees.

The daily articulation work of software development that was necessary to re-adjust project plans and to deal with the necessary coordination of tasks and unforeseen contingencies like new found bugs was mainly performed in a written manner, usually relying on Instant Messengers or emails (the latter of which were mainly used for more formal matters and when time differences did not allow for synchronous communication). In this regard, the actors reported a preference for written communication over phone calls because writing in a foreign language was easier for them than speaking, as they said. Being able to use dictionaries and translation tools for looking up unknown terms was reported as a major advantage of written communication. One developer of company Beta even reported that he had once observed a Russian developer writing Instant Messages in Russian, then copying his text into a translation tool, and copying the English result back into the Instant Messenger, finally checking for errors before he sent his answer—a fairly complicated and cumbersome practice in his eyes, but it allowed the Russian who apparently felt not very secure in English communication to discuss his work with the German project manager. In this context, practitioners said they liked that Instant Messages were rather informal compared to emails, and that they granted enough time to come up with formulations and to correct errors before sending the message, thus avoiding unintentional and awkward communication breakdowns. Further advantages were the possibility to archive discussions in personal chatlogs and email repositories, as well as allowing to share links, code snippets, and specifications (just to name a few examples) easily by copy and paste. Last but not least, the developers liked the low level of intrusiveness of Instant Messengers and emails, which would allow the recipient to finish a task at hand before answering the message; this was especially preferred by developers who reported a high level of immersion when working on code or thinking about complex development problems [25] (see chapter 8).

For the organization of the development work, centralized repositories and databases also played a significant role in both cases. For example, company Alpha used the centralized

version control system Subversion (SVN) for managing their codebase, as well as the bug tracker Mantis for managing bugs and feature requests. As the code development of the larger projects was mainly carried out by the Russian colleagues, the SVN was operated in Tomsk for performance reasons. The SVN was commonly used by all projects of the company that involved code development. The use of Mantis, on the other hand, differed between the various projects of company Alpha, depending on the project managers who sometimes preferred more lightweight solutions in case of smaller, local projects (see chapter 5). We also observed different assessments of tool usage in case of company Beta, where the Russian developers wanted to deploy other tools than the Germans, even though the company had clear guidelines about the tool infrastructure (see chapter 7 for details on this argument between the teams).

During our on-site visits, we observed again and again that the use of formal coordination tools like bug tracking or version control systems was highly embedded in informal communication that preceded and followed the interaction with these coordination mechanisms. For example, project managers and developers used to discuss questions like "What is the status of bug number X", "What should I do first: this or that task", or "Can you help me with this task please, I need help!". The information stored in the formal databases was thus used by the actors to refer to current tasks and breakdowns (in case of bugs), but the context of the information needed to be discussed before the related work could actually be performed.

The reason was that unexpected contingencies like new found bugs made it necessary to re-articulate projects and plans in a fashion very similar to the one described above. As project managers (responsible for the communication with the customer and the planning of the project) often were not able to assess the severity and technical background of a breakdown situation, they needed feedback from the Russian developers in order to decide how to deal with it. The Russians, on the other hand, often lacked knowledge about the business background (such as priorities, context of the customer relations etc.) in order to assess which task had the highest priority, why a technically interesting idea had sometimes to be dropped in favor of a far less stable one, and how much time and resources would be available for them to finish the project (an aspect which could change all the time due to conflicts with other projects etc.). Information like this ususally was not stored in the coordination tools because it was regarded as the (necessary) context of the task but not as part of the task itself. Nevertheless, it was of utmost importance to know about these issues in order to articulate the development work.

Furthermore, our study showed that the information stored in the systems often was out of date because the developers did not update the status of their tasks frequently (especially in critical situations like when a project was close to finish and a lot of bugs, often found in the last minute, had to be fixed). This was especially the case in company Beta, which saw the coordinative tools not only as a manner for coordination, but also as a tool for controlling the progress of the development work. However, as this aspect was only important for the German project management, Russian developers were reluctant to maintain this control function by constantly updating their tasks in the systems, thus making it necessary for the German project manager to constantly request their status as he could not trust the state that was represented by the task overview in the bug tracking system [31] (see chapter 7, see also [84]). We will come back to these socio-cultural aspects in section 9.1.3.

The formal systems thus often served as a means to document the result of informal articulation work, like for example the outcome of the discussion of a new bug (including deadline for fixing and results of the attempts to reproduce it) or the request for a status update. Practitioners often simply copied and pasted sections of their chatlogs to the systems, thus storing the personal logs for later retrieval by themselves or other team members. In general, individual tasks were rather articulated in an informal, direct manner, while the formal systems were mainly used to document agreements, trace the (rough) trajectory of the project, or represent the current state. While these coordination mechanisms still had a very important role in serving as maps or scripts for team members [197], they were not self-sufficient and highly related to the informal coordination work that was going on "invisibly" between individual team members in order to negotiate the context specific information that had to be discussed in order to take a decision or to come up with a plan. In this regard, the necessary articulation work for making use of the formal coordination mechanisms was related to many aspects of learning and knowledge exchange between the teams, as the organization of their work required knowledge of the particular background of a situation in order to find a practicable solution for a problem. Related discussions circled on questions about the current situation of the company, available competencies of team members, the needs and expectations of the customer, the potentials of the technical frameworks that were used, possible conflicts with other projects, available funds, and legal backgrounds (just to name a few). For instance, in company Alpha a project leader said during the interviews that it would sometimes be hard to keep the Russians on track. Due to their technical focus, they would often like to rebuild existing software in order to create a technically sound solution, but without taking into consideration whether the company could afford this extra work or

the customer would be willing to pay for that. This often caused discussions about what was possible in the context of a project, and what was not—mediating between technical and business aspects of the cooperation (as it would sometimes make sense to re-engineer a piece of software even though the company would not earn money by it directly, for example in case that it would benefit possible other projects that relied on the same code base, but often made no sense from the short-term perspective of the daily operation of the company). Articulation work thus stood in a dialectic relationship to learning and knowledge exchange, which on the one hand was *necessary* for the articulation of software development tasks (for example for planning a project) but on the other hand was *stimulated* by the articulation work going on in the project (for example with regard to possible synergies or new project ideas).

9.1.2. Articulation Work and Organizational Learning

With regard to the relationship between articulation work and learning discussed above, our analysis of the two cases showed that articulation work was not only important for the operational aspects of single-loop learning ("Are we doing things right?"), but also for the strategic level of double-loop learning ("Are we doing the right things?") [9]. For the specific context of software offshoring in SMEs, our research showed that the modularization of software development work in terms of dividing responsibilities between German project managers and Russian programmers could lead to problems in practice. As planning and coding were handled locally as far as possible, this form of distribution required less inter-team articulation work, thus limiting the awareness between the teams (and thus the indirect learning possibilities at the inter-team level in terms of over-the-shoulder learning, see [234]). Furthermore, as informal articulation work was mainly performed on a personal level which often was "invisible" even for the local team, the situated forms of knowledge exchange between the teams were reduced further [32] (see chapter 6).

Generally, both cases showed that situated, spontaneous learning often was initiated in a proactive fashion by developers who showed an interest in what was going on in the company, or in the course of overhearing something that was discussed in the coffee kitchen or at an adjacent desk. As both companies only consisted of two or three office rooms for three to five developers as well as smaller rooms the doors of which remained open most of the time, these incidents were common in the local offices. While explicitly organized occasions for learning like weekly meetings or project related workshops were reported as being useful for getting an overview, practitioners reported that their benefit for learn-

ing was limited as they were often too constrained for useful discussions or focused on details that were not relevant for all participants [26] (see chapter 8). Especially the German developers of Alpha who were not involved in project management criticised the meetings as more or less irrelevant for their work. This was reported also with regard to the product and development databases that company Beta had introduced (see chapter 7). According to the practitioners, information provided there often was too detailed or too generic to be useful with regard to specific problems or challenges the individual team members had to face in their daily work. This also related to the necessity to get an overview of the overall situation in the companies, which was hard to get by relying simply on the formal occasions and tools, as was discussed earlier [24]. Hence, instead of relying on the fragmented and often outdated information in the information systems, practitioners usually asked colleagues directly or walked around the company to get an overview about what was going on, often meeting in the coffee kitchen which turned out to be an important place for knowledge exchange (and was also equipped with a whiteboard in case of company Alpha). Information picked up during the meetings or in the information systems was used for discovering new topics in the company or finding out who could be asked regarding certain problems, but the meetings did not serve as opportunity for the knowledge exchange itself (see chapter 8)—instead, knowledge exchange was organized in a fashion similar to the formal and informal aspects of articulation work discussed above, with practitioners asking colleagues directly for advice in a situated fashion.

As these practices mainly worked locally, knowledge exchange and learning between the cooperating teams was limited to the aforementioned chats as well as the personal meetings between the sites [25]. The latter turned out to be highly important for the inter-team learning both at the operational and at the strategic level. Apart from planning new projects, creating a shared understanding (which also made the later interaction via chats easier) and establishing a technical basis for the cooperation, we were also able to observe incidental knowledge exchange with regard to the appropriation of software tools, technologies and development practices [27]. For example, in case of company Alpha we were able to attend a two day meeting between German and Russian developers. The aim of this meeting was to discuss the benefits and possible risks of migrating a product developed by the company from C++ to Java—a transition which had been suggested by the German manager as a strategic direction, because the customers often requested platform independent tools. As the transition had strong implications not only with regard to the possibilities of the two programming languages, but also for the deployed development tools (switching to the IDE Eclipse), the developers spent several hours

discussing possible strategies, advantages and disadvantages, as well as possible problems which were mainly related to the high efforts of re-engineering the whole platform. This meeting was assessed as being very valuable and important by the participants, as it allowed them to spent much time in discussions and exchange knowledge that was highly relevant for their current (and future) work. At the same time, such occasions also were reported to be important for the practitioners who saw them as a good opportuinity to learn from each other, benefiting from the experience of senior team members or getting an overview regarding the state of other projects in the company (see chapter 8 for a detailed discussion of this practice; see also [27]).

However, such meetings were relatively sparse as travelling is expensive and usually limited to one or two developers or project managers. This limited the possibilities of knowledge exchange between the teams considerably and turned out to be problematic especially with regard to the strategic learning capabilities of the companies. As outlined above, both companies felt that they needed to exchange more business related knowledge with the Russian colleagues, but had problems to achieve this in their daily practice. At the same time, in technical issues both companies felt dependent on their Russian partners to some extent, as the Russians were responsible for the main part of the code development. This was further aggravated by the flexible work contracts in Russia, which reduced the stability of the Russian team. Hence, in company Alpha the surprising resignation of one of the Russian developers (who had been offered a more lucrative job by a large Russian corporation) was perceived as a great loss of competency, although this developer had only worked for the company for one year. In order to prevent a slow but steady loss of competencies (also in view of a possible debacle like the loss of the Russian senior developer which was perceived as a major threat to the company), the manager of Alpha established the practice of inviting capable Russian developers together with their families to Germany for longer periods of time, in this case for two years. On the one hand, the possibility to be invited was meant as an incentive for the Russian developers to stay with the company and work hard; on the other hand, the manager hoped to be able to initiate an exchange of knowledge from the Russian technology experts to the local team members, who often had not worked with the code in their function as project managers for long periods or were new employees that needed an introduction to the frameworks and code base of the company [26] (see chapter 8). During our research, the company also thought about sending a German project manager to the offshore site for a similar period of time in order to exchange business-related knowledge with the Russian colleagues; due to the financial crisis in 2010 as well as personal reasons, this exchange has not been initiated so far.

Apart from such attempts to intensify personal contacts between the teams, we were not able to observe any systematic form of organizational learning in the companies [32] (see chapter 6). The dependency on written communication constrained the possibilities for learning between the teams considerably, limiting it to concrete operational problems and basic articulation of individual tasks that remained more or less invisible at a team level. While the practitioners were aware that their inter-team cooperation was lacking such aspects to some extent, for example with regard to limitations of their formal coordination tools, and even though they had some ideas as to what the reasons for these problems were and how they could be solved, they lacked a way for systematically working on such problems and, more importantly, implement the related solutions [28] (see chapter 4). This limited the possibilities for organizational development of the two companies in our study considerably; while company Alpha sticked to their form of adhocracy with all the related problems and the high overhead of articulation work, company Beta encountered a drawback due to their failed attempt to restructure the organization towards a closer cooperation between the sites (see chapter 7.3 for a detailed discussion of these incidents). This situation was partially related to the necessities of the daily work which offered not much room for strategic considerations as prevailing operational problems often had priority and called for pragmatic solutions; on the other hand, the socio-cultural aspects of the cooperation made it hard even to broach these issues across team borders or to implement solutions in the distributed cooperation due to the limited possibilities to deal with the necessary articulation work.

9.1.3. Articulation Work and its Socio-Cultural Embedding

During our research, we found that articulation work practices often were affected by the socio-cultural relations in the field, which turned out to be a dominating factor for the cases of inter-organizational cooperation we investigated [25]. In this regard, trust and social capital turned out to be important (and necessary) assets for the companies, especially with regard to the dependency on the expertise of the Russian developers in case Beta (see chapter 7 for a detailed discussion of the role of trust and social capital for case Beta). The practitioners considered this aspect as very important, as both cooperations where based on personal contacts to offshore developers *before* the cooperation was started. This finding is in line with the results of the initial survey amongst German SMEs, where most of the companies preferred similar strategies. Furthermore, most companies said that they would have actively looked for nearshore locations as opposed to

offshore countries like India or Asia because of the perceived greater cultural adjacency between Germany and Russia (see chapter 5 for a discussion of this aspect).

Culture also turned out to be an important factor in the two case studies which affected the cooperation in terms of how the members of the cooperating teams dealt with each other. However, related conflicts were not so much based on the assumed different "national" cultures between Germany and Russia but more on the different perspectives that were outlined above as well as the asymmetric power relationships in the field. These often led to misunderstandings and micro-political struggles that made it hard for the actors to articulate projects and learn across the borders of companies [26] (see chapter 8 for a discussion of the role of culture in case Alpha). Different assessments on what constituted "good" software development practices, the use of "appropriate" tools [27], as well as language problems which made it hard to plainly discuss delicate topics due to the limitations of written communication [165] played important roles in these struggles. For example, in case of company Alpha the Russian project manager (who was responsible for one of the projects) and the German project managers (who cooperated with Russian developers in the context of other projects) expressed very different assessments of "good" practices for software development. While the German project managers stated that the Russian developers would be hesitant to document their work because they would think that good code was self-explanatory, the Russian project manager explained that the Russian team would maintain a very detailed documentation of their work (in contrast to the Germans, who would not be interested in rigor documentation). During the observations, these different assessments could be related to the fact that documentation in both teams often was kept in form of private notes, which only sometimes were stored on a shared FTP server (usually when the German project manager requested it). Even where public information was present, actors doubted that it would be up-to-date and thus preferred to ask colleagues directly for assistance instead of relying on the documentation (as outlined above). Due to the private character of documentation, no member of the teams had an overview over the state of the documentation; at the same time, the private character made developers hesitant to publish it in some sort of central repository because it required some effort to bring it into a format that was self-explanatory or even understandable for colleagues. Similar issues were observed in case of company Beta with regard to the use of their centralized development and product databases (see chapter 7).

Regular personal visits as well as the attempts to initiate a long-term exchange of employees were very important practices for dealing with these problems, as they allowed

practitioners to obtain a better understanding of the work context and perspective of the remote team during their visits. Especially in the case of company Alpha, a Russian developer who visited the German headquarters for two years, functioned as a "bridge" between the two cooperating companies [153], not only enabling the trans-organizational knowledge exchange but also mediating micro-political conflicts that prevailed between the two teams [24]. As the senior Russian developer, he had a deep understanding of many of the projects that had been handled by the Russian colleagues. Thus he was asked to introduce German developers to the different products of the company, sharing his knowledge with his colleagues to bring back important technical knowledge to the German team (as was explained above). At the same time, he often was asked to mediate in case of conflicts with the Russian team, because the German colleagues where often hesitant to talk plainly to the Russians about such controverse issues. An example for such an incident was the conflict about the homepage of the Russian company (see chapter 8 for an analysis).

Generally, the social relations between the teams were important factors for articulation work [31]. As the Russian team members as contractors were constantly in a weaker position when it came to conflicts with their German colleagues, formal coordination mechanisms were sometimes regarded as "annoying" duties or single-sided means of control without much practical benefits for the work of the Russian developers. For example in the case of company Beta, the German project manager wanted to control the progress of the development work by checking the status of tasks in the bug tracking system, and thus requested status updates in case of delays. As discussed above, these control mechanisms were not only considered to be annoying by the Russian developers, but also turned out to be deceptive. As the Russian developers were reluctant to maintain their tasks in these systems despite regular admonishments from the German project manager, the systems often were not up-to-date, and constant informal communication was necessary to get an overview of the project status. This example illustrates a constant, underlying conflict of interests: the German companies wanted to keep an overview of the progress of the work in the remote team in order to identify possible problems and prevent the misuse of resources at an early stage. At the same time, the Germans feared to demotivate their Russian developers by enforcing duties and directions that were perceived as being annoying. As they were not able to enforce their preferred coordination mechanisms without putting the whole cooperation at risk (which depended on the motiviation of the Russian developers), the German project management needed to trust their Russian employees to some extent. The Russian developers could easily sidestep the available control mechanisms by simply ignoring them, thus, the German management had to ap-

peal to the Russian colleague's willingness to cooperate. As a consequence, the offshore cooperation stood in a field of tension between the need to control and the need to trust the remote team members, thus making social capital an important asset for the cooperation. At the same time, as we have shown in case of Beta, the social capital was not able to save the cooperation although it remained strong despite the prevailing problems (see chapter 7 for a detailed analysis of the role of social capital in the offshoring failure story of company Beta).

For similar reasons as outlined above, practitioners often avoided discussing prevailing problems in the cooperation. On the one hand, such discussions often led to very emotional arguments between the sites (due to the limited possibilities to communicate, see above). As such arguments were regarded as threats to the conduct of the daily work, the German clients tended to avoid such discussions, fearing to demotivate their Russian contractors. On the other hand, different perspectives and assessments were often related to "cultural" differences between the teams, for instance regarding different assessments about what characterizes "good" software development practice or how to deal with criticism (see chapter 8). Our analysis showed that these references to a perceived national culture often were used as a self-recursive, stereotypical pattern of explaining breakdown situations which could not easily be solved by referring to "rational" arguments [26]. "National culture" thus was used as an universal argument for explaining prevailing problems that the actors found difficult to solve, thus interpreting such struggles as external limitation of the cooperation that could be mediated, but not be eliminated—and needed to be accepted as given. For example, employees of Alpha explained that Russians would always react very sensible to straightforward criticism, while German developers would be much more open in that regard. On the other hand, Russians told us that they would get criticized all the time, and that they would hardly get any positive feedback when something went well. In combination with the asymmetric power relations and the different perspectives outlined above, this indicates that the perceived sensitivity of the Russian colleagues was not a matter of any "national" culture (as some of the German colleagues assumed), but rather a matter of the context of such situations and the weaker position of the Russian colleagues in case of conflicts (see chapter 8 for a detailed discussion of this aspect of Alpha's cooperation).

9.2. Implications

9.2.1. Methodological Implications for Research

Apart from findings regarding coordination and learning in distributed teams, our study revealed implications with regard to the methodological aspects of researching into distributed software teams. As we have pointed out above, conducting research in the context of Global Software Development can be subject to similar difficulties as the developers have, for example with regard to the limited awareness of the remote site. The highly political context of distributed development work as well as the challenge of getting access to the field in the first place are further challenges in such fields. These issues are not new for ethnographers as they apply for other fields like virtual organizations or communities in general, too [112, 245], but they have hardly been discussed in the context of software engineering research so far [173, 28].

As we have shown in our analysis of the research methodology, the role of the researcher in the field can be an important pivotal point for the whole research project. In our project, taking the role of a learning mediator allowed us to get a better access to the company, but at the same time it exposed us to the micro-political quarrels between the sites. At the same time, we received very positive feedback during the workshop in company Alpha. This reassured us that the general aim and scope of our project was valuable for the practitioners, even though the differences between the academic world and the software company made it hard to align our research to the always changing interests of the practitioners (see chapter 4 for a discussion of this aspect). In a way, the researcher can be seen as a bridge between the cooperating teams as well as between the management of the company and the developers. Due to the high impact of social issues that we have encountered in our study, we conclude that doing qualitative research in GSD is not only a methodological problem but also a socio-cultural issue between the role of the researcher and practitioners in distributed cooperations—an aspect, which is important to keep in mind when doing research in this field (as it is easily overlooked).

While the inherent problems of doing research in GSD can probably not be solved generally, our experiences hint towards several implications for entering distributed fields: first of all, it is important to spend enough time in the field to fully understand the phenomenon to be investigated. As we have seen, the practices of managing distributed software projects can be quite complex and volatile, and they include the use of various interrelated software tools and communication channels that can be hard to grasp. At the same time, as German team members concentrated on the management side of the

software development work, our case studies were affected by a clear "abundance of code", which often was referred to, but seldom touched by the actors during our presence in the field. We thus needed much time and careful analysis in order to understand what was going on in the projects, especially as the actors themselves often had problems to keep an overview (see chapter 8) and broach these issues during interviews [225].

While the time problem seems to be a trivial point that is well known in the literature (see for example [176]), distributed fields often further reduce the possibilities for research because multiple sites have to be covered [245]. While it would be possible to concentrate on one site only, such an approach would usually be biased due to the different assessments and perspectives that we have outlined in the previous sections. As we have seen with regard to how practitioners presented to stereotypical cultural explanations for seemingly unexplainable organizational problems (see chapter 8 for a discussion of this finding), it turned out that it was very important to look at these issues from different perspectives in order to fully understand them. Hence, we conclude that in order to get an analytic understanding of distributed software development even in such small teams, it is not enough to only focus on one team [26]; one should always also try to understand the perspective of the cooperating team(s). As our example showed, even though getting access to remote teams can be quite hard, there is much to be gained because many assumptions on the particularities of international cooperation turn out as far more complex when they are analyzed from multiple perspectives. Furthermore, practices need to be studied in context, and should not be reduced to oversimplified "best practice" examples [166]. At the same time, complex concepts like "culture", "trust" or even "coordination" need to be carefully applied in order to avoid falling for stereotypes and biased views on the practices unders study.

9.2.2. Theoretical Implications for the Scientific Discussion

In his article "The articulation of project work" from 1988, Anselm Strauss introduced a generic framework of project dimensions for analyzing articulation work [221]. It is not surprising with regard to this framework that the software development we found in the field needs to be interpreted as consisting of *complex* trajectories that involve many *non-routine* tasks. The specific situation of the small to medium sized companies we investigated in terms of their highly distinct market niches and service orientation made it necessary to keep up a high amount of flexibility [160, 78]. This made the projects also very complex as this demand required ample articulation work for constantly dealing with unforeseen change requests as well as overlapping responsibilities and contingencies

between different projects. Despite the fact that the involved teams only consisted of up to about 20 developers, practitioners found it hard to implement a stable division of labor and clear-cut responsibilities (even in the case of company Beta which struggled to implement a more formal way to deal with their software development, and failed in the end). Instead, as the projects at hand were constantly changing and highly volatile, informal ways of dealing with software development with a very low amount of routinized procedures and formal divisions of tasks and responsibilities were upheld by the companies [29].

It was especially the very vision of the product to be developed with its plethora of (often contradictory) interpretations, implications and dependencies that made it hard for the practitioners to treat software development as a standard routine. Even though many procedures of software development could be seen as routine tasks—such as filing a bug in the system or committing changesets to a source code repository—the higher-level project visions that were changing all the time in combination with the always prevailing contingencies (related for example to the business strategy) required a constant re-negotiation of the cooperation arrangements and made it hard to come up with standardized strategies for dealing with the daily work [7]. Hence, as the interpretation of what was sought to be the shared vision of the product was far from being stable and hardly documented, articulation work was needed for constantly (re-)creating common ground between the actors and making the whole cooperation possible. While this is in line with the literature on the general use (and related limitations) of formal coordinative tools for software development [94], our findings revealed particular aspects of the use of such systems in the context of small enterprises which have hardly been discussed in this regard: especially in company Alpha, shared specifications turned out to be less technical, but more conceptional in the given case. This means that the level of detail was adjusted more to the perspective of the customer than to the perspective of the developers that had to implement the given features at a technical level. This finding extends current findings with regard to the evolution of "inofficial" tools around the use of "formal" IS in the context of large software engineering projects [101]. While general aspects like which technology to use and what features to implement at the functional level were documented, the technical aspects of the implementation (user interface design, branching or re-use of existing code, algorithms etc.) were documented only in the context of written notes, and very roughly at the beginning of a project—while the particularities were negotiated later in the course of the project (orienting on the general agreement with the customer about the schedule and development aims). The stability of the plans was upheld only towards the customer who wanted a clear agreement on what was to be

developed—but apart from that, specifications were neither documented nor stable, and could change according to emerging events and ideas of the developers and the project leaders in the course of projects. This finding opens an interesting perspective on the design of central repositories for managing project artifacts as suggested by the literature [51]. At least such repositories should offer the possibility to add personal notes to the artifacts and take into account the different levels of detail required (see chapter 9.2.4 for a discussion of the design implications of our study).

In both cases, the specific context of offshoring introduced new challenges to the practices of articulating software projects. These were related to the communication barriers and the limited awareness between the teams [70] but also to the socio-cultural differences that we found in the field [26]. Even in company Beta, which followed a much more plan-based approach as compared to Alpha, the premise that software development could be organized by pre-defined plans turned out to be deceptive [10]. The development (in terms of programming) could not be isolated from the planning (in terms of requirements specifications), because the plans and specification documents always required interpretation with regard to particular aspects as outlined above. While this is in line with the literature on plans and situated action ([226, 199]), the specific case of offshoring introduced new challenges to this general problem: articulation was needed in this regard to mediate the different perspectives and contexts that the teams were working in (as a kind of *agency*), in order to realign interpretations of plans and deal with emerging contingencies. New rules and tools only introduced new possibilities for interpretations in this regard, as the example of company Beta showed, without solving the underlying problems of the distributed cooperation [31]. For the same reason, the standardized coordination mechanisms that the companies used were only partially suited for dealing with such problems as they mainly focused on the managing task-to-task and task-to-person dependencies but did not cover the context-related aspects of articulation work [220]: the person-to-person dependencies and the organizational setting of the interaction, the privileges and prejudices involved, possibilities of regulating conflicts of interest, as well as impacts of the function of the enterprise in the socio-economic system at large [12]. Especially for software development work, these aspects turned out to be of utmost importance as they represented the necessary "common ground" for interpreting the information stored in the coordinative artifacts in order to establish patterns of collaboration that did not require intensive explicit communication. As the literature suggests, core developers played a very important role in this regard as the drivers of productivity and as "bridges" between the teams at the same time [43]. In this regard, our findings support the view that such roles can not be established formally but that informal mechanisms

like trust and social capital played a much higher role in this regard. The implications of these findings for supporting bridges in practice will be discussed below (see chapter 9.2.3).

Our findings also provide insights into the relations between articulation work and learning in the context of software offshoring in SMEs. As we have found, formal approaches for supporting articulation work (for example bug trackers and version control) played an important role for the coordination of software development in the cases we have investigated. Hence, it can be concluded that such focused approaches (like for example ARIADNE [206]) are feasible also in case of software offshoring, as they provide important functionality for the actors in terms of providing lists of tasks and guidelines (for the software developers) as well as an overview of the current project state (for the project managers). At the same time, it became apparent that the *context* of the tasks that were to be coordinated by these systems were rather negotiated in an informal, situated manner as outlined above—indicating that such tools either need a higher level of tailorability, or a closer integration with informal communication media like Instant Messengers [96]. The information in the formal coordination tools turned out to be important points of reference for discussions about how to deal with the related tasks in practice. Emergent roles like that of knowledge brokers and bridges are important aspects in this regard, as are ad-hoc processes of situated articulation work. Especially with regard to the grown, heterogeneous tool infrastructures with the dynamic practices that we found in the companies of our study, it is unlikely that one solution can be designed that fits all needs [138]; rather, our findings suggest that it will be important to think of ways how different tools and services can be integrated with each other, and how practitioners can be supported not only in tailoring the coordinative artifacts (the plans and specifications) but also the related coordination mechanisms in terms of the practices that evolve around the use of these tools. As tools and organizations are known to interact with each other in the long term, it is important to support such interactions [95]. Hence, we conclude that management conceptions that take into account the relationship between formal and informal coordination (like continuous coordination [177]) can be very useful in the context of offshoring, but only if they are fine-grained enough to account for the flexible and complex interactions outlined above, also with regard to the different work contexts the teams are operating in (i.e. their different perspectives).

In the particular case of distributed software development, we have seen that such practices often happen in the context of personal meetings, where actors try to bring their different perspectives and needs together, conjointly creating (and altering) practices and

plans to the emerging necessities of the cooperation. This common ground then serves as a basis for the later cooperation, which is often subject to emergent changes and adaptations (because the plans, as pointed out, only provide a rough guideline, but do not prescribe every detail of the joint project). Software development of SMEs, especially in distributed teams, is not mere office work; the interpretations of the tasks that are negotiated are far from being stable and can change from project to project. At the same time, strategic aspects have to be considered that can stretch from one project to another. As a result, such fields require coordination mechanims that are more flexible and have to be understood from two sides: as the foundation of articulation work, but also as its result—not just in the sense of plans serving as maps and scripts [226, 197], but also in their dialectical relation to learning and knowledge exchange at the operational and strategic levels. Especially in the case of software offshoring in SMEs, such approaches also have to consider the specific socio-cultural embedding, in so far as articulation work and related protocols do not only serve as coordination mechanisms but also as "interaction mechanisms" which mediate the socio-cultural interaction between the teams. In such a perspective, it is important not only to support modeling coordination mechanisms, but also to support bridging the different perspectives on such models (the "social" side of rules) that are in place in complex work environments like GSD projects [198]. Hence, such aspects have to be taken into account if we want to make articulation work reflective and operative in distributed teams, covering both aspects of single- and double-loop learning (as both are closely interconnected for the practice of distributed software development) [32]. Our research offers several insights with regard to how this can be achieved in practice, as will be discussed in the following sections on organizational and technological implications.

9.2.3. Practical Implications for Companies

At a practical level, our findings stress the role of inter-organizational learning for distributed software teams in German SMEs. Our theoretically substantiated findings allow to draw several conclusions for organizing distributed software development projects:

First, software SMEs that work highly customer- and service-oriented in very specific domains benefit substantially from the direct interaction of developers across companies' borders. Hence, such strategies should be successful which orient on (knowledge) management concepts that do not include too much specialization and segmentation. Orienting on agile development methods like Scrum [228] or Extreme Programming [17] seems to be promising in this regard. Formalizing development work may seem as a

possibility for reducing the need of ample articulation work, but such strategies have the disadvantage of reducing the necessary flexibility of SMEs (see chapter 6). Hence, our findings suggest that companies should avoid over-formalizing such approaches by introducing specialized roles and formal documentation, contrary to what literature on agile offshoring often suggests [194, 14]. Instead, our findings imply that companies should rather stick to informal concepts like "bridges" to leverage the difficulties of distributed cooperation [153]. Actors serving as bridges or knowledge brokers have turned out to play an important role for knowledge exchange [147] and mediate in conflict situations [244]. As we have shown, such roles can best be fostered by initiating close personal contacts between the sites, for example by inviting colleagues from the offshore teams over to the German site for longer periods, or vice-versa. At the same time, we have shown that learning on the strategic level remains a challenge for agile software teams that so far is not well supported by existing models (agile methodologies are rather interested in operational aspects of learning, often ignoring such challenges [235]). As a result, related management practices are hardly established, especially in the case of offshoring with all the related limitations. Possible improvements in this regard could be derived from action research approaches for organizational learning like *Business Ethnography* as we have outlined above [214].

Second, on a technical level there is a high availability of tools for the formal articulation of local and distributed software development projects. Freely available and highly adaptable bug-tracking systems and version control systems are common and very useful coordination mechanisms, especially for distributed software development. The informal aspects of articulation work in software offshoring are far less supported right now, and mainly handled by sticking to emails and light-weight Instant Messaging tools. While these tools clearly have limitations, they apparently worked for the practitioners in our study, helping them to stay aware of what was going on in the cooperating team, and performing the necessary articulation work for using the formal development IS [6, 100]. However, our findings revealed also limitations that were related to the dependence on such tools [123]. First of all, the communication needed the common ground that still depended on personal meetings in large parts. While operational questions could then be handled rather easily in the later course of the projects, strategic aspects were only hardly covered and limited the strategic learning capabilities of the companies. Furthermore, such tools mainly work on a personal level, and the articulation work that is done in such chats remains more or less invisible to the team. In this regard, there should be a high potential in using micro-blogging tools as lightweight coordination tools inside companies. As such tools have features similar to the used Instant Messengers—being lightweight,

rather informal, non-intrusive—they could well complement the informal tools used for articulation work. At the same time, micro-blogging has advantages with regard to the visibility and traceability of the exchanged information, which could have a high potential for supporting learning processes within and between the teams [35, 181, 252]. However, it is important to take into account that tools can be adopted differently in different companies [145], and that appropriation support thus is a pivotal aspect of introducing new tools, especially in the field of GSD [138].

Third, with regard to the socio-cultural embedding of articulation work in offshoring projects, our results show that concepts and tools can only be successful if acceptance can be achieved on an inter-team level. Here, our findings suggest that trust and social capital can be two-edged swords and that bridges can be very successful for dealing with problems related to socio-political struggles that result from this field of tension. At the same time, there is also the danger that important topics are not broached in order to avoid offending the distributed colleagues—one of the possible negative side effects of social capital (see chapter 7). Companies should not forget the benefits and dangers of such close contacts, even when they benefit from a high level of social capital with their offshore partners, as we have pointed out above. For the practice of small software teams, we further see a high potential in adopting social media for software offshoring projects, as such informal tools could probably leverage the limitations of distributed cooperation [218, 19]. In general, our analysis showed that practitioners benefited considerably from personal contacts between the sites, which played a pivotal role for dealing with underlying conflicts and misunderstandings between the teams. As personal meetings are costly and limited, practitioners need substitutions which should also be addressed by information systems; social media can serve as informal and flexible extension to the more formal information systems of software companies [124, 230].

All in all, our findings suggest that it can be very risky for SMEs like the ones in our sample to engage in offshoring. The disadvantages and challenges are not easily to be dealt with, and even if companies succeed in implementing practices and tools that help to mediate the distribution of the cooperation, the related management overhead makes it likely that they will not achieve the same efficiency as compared to local teams. Even from an economic perspective, the companies in our sample found it hard to assess whether they had an overall benefit from the offshoring cooperation in place (see chapter 4). This also related to the often promised possibility to address new markets by enganging in offshoring [243], an aim that both companies in our sample failed to achieve. Based on the cases in our study, it seems that international cooperation in software de-

velopment is dependent on having talented and motivated developers at *both* locations. Hence, remote partners should not be seen as more or less exchangeable helpers but as partners in an inter-organizational learning process. If companies manage to learn from external partners, instead of teaching them how to do things, companies can benefit from international cooperation. This seems to be congruent with the phenomenon of Open Source Software development, where communities often evolve around a stable core of developers, be it in the context of a company or an individual developer or a group of technology experts (or hybrid forms of such arrangements) [39]. At the same time, it contradicts one of the basic premises that is often present in the discussion and literature on offshoring (even with regard to agile methodologies): the assumption that software development is a task that can be easily performed by external programmers if only the plans (architecture, specifications) are worked out well enough [225]—a premise that did not work out for the companies in our sample, as the case of company Beta illustrates. As a result, we conclude that small companies need to take into account the complex socio-cultural context of international cooperation and include the perspectives of the offshore developers instead of single-sidedly attempting to formalize their development practices [104].

9.2.4. Implications for Design

Our findings revealed complex interrelations between formal and informal aspects of articulation work with forms of organizational learning. This complexity can lead to problems in distributed teamwork, with negative consequences for team coordination and related learning processes. From a design perspective, it is thus promising to aim at supporting these aspects, taking into account the theoretical and practical implications of our work.

Existing approaches for supporting distibuted software development often are based on attempts of minimizing the need for articulation work by optimizing the division of labor or formalizing communication [109, 180], or on supporting articulation work by offering further or richer comunication media [90] (see chapter 2.2.2). Our findings hint at additional implications for design with regard to improving the visibility of articulation work with a view to the shared work context in distributed teams, for example by identifying and filtering individual articulation work events in the existing communication media and development IS and presenting them on a shared semi-public display or a similar device. An adequate visualization in form of "articulation spaces" would solve the problem that articulation is often invisible for the team as it is often performed individually.

At the same time, such an approach would improve the traceability of articulation work, which often is hard to reconstruct by referring to the formal coordination tools and artifacts (like bug trackers or project plans), and thus could support double-loop learning in distributed teams.

Conceptionally, related designs could refer to prior work from the field of Media Spaces and Public Displays, as well as to the literature on supporting awareness for seamless team coordination. Like in the conception of the awareness pipeline [80], an *articulation pipeline* could be designed to aggregate and filter information about articulation work events from the various tools, media and artifacts that are used for articulation work in distributed software development (see figure 9.1). In contrast to existing "dashboard" approaches [20] that focus more on presenting formal aspects of the cooperation for the team, such an approach would have to take into account the aspects of privacy and information overload that has been discussed in the context of awareness [132, 82, 79], but also the socio-cultural background of international software teams, for example by including social, informal media into the conception [124]. Due to the highly private context of such media, such information could probably only be accessible by offering assistants for the actors, and not by automated retrieval [218, 19]. It would be especially promising in this regard if team members who act as bridges and knowledge brokers could be motivated to use the system actively. Furthermore, such an approach would also allow to include other important technologies like appropriation support [215], community-based help systems [216], expertise recommender systems [178], or awareness mechanisms that have been developed in the context of coordinating software development work [20, 177].

For the visualization side, public displays have hardly been discussed in the context of supporting software offshoring so far. Instead, existing tools usually center on desktop workplaces as interface for the supporting Information Systems [233, 207]. While such approaches seem to be the natural choice for supporting computer-based work like software development, they mainly focus on supporting individual developers without much consideration of the related team awareness. At the same time, the desktop metaphor has some limitations. For example, information is only accessible if actors actively search for it, while notifications for example by pop-up windows are often regarded as being intrusive and annoying. The presentation of awareness information thus has to be context-sensitive [183], a requirement which has not been solved completely so far. With regard to our findings, it could be interesting to experiment with Instant Messengers as output medium at the desktop level. At the same time, public displays have several advantages

in comparison to desktop notification systems: they support *peripheral awareness* of such information that is not actively searched, with a very low level of intrusiveness [130]. At the same time, they can be used cooperatively and thus provide a shared context for the cooperative work [242]. According to our findings, a good space for setting up such a display would be the coffee kitchen where much of the knowledge exchange takes place anyways. Furthermore, such an informal space could stress the informal character of the display and make clear that the solution would not center on the formal aspects of the development work only. Other possible space would be a meeting room, or even the different office rooms where displays could be placed near the door to support peripheral awareness when leaving a room.

Due to the distributed work context of software offshoring, related approaches would have to include several displays at different companies, as well as personalized ways of access for developers working at customer sites or in their home offices [38]. The displayed information would have to take into account the different working contexts of the cooperating teams because the use of the displays as planning and learning tools can be expected to differ strongly between the cooperating teams due to the different perspectives, aims, and areas of work. The articulation spaces should thus not focus on providing as much information as possible (in the sense of the *rich media theory*, see [165]), but on putting the available information into context [81]. Hence, they should be designed as *boundary objects [210],* and not as central knowledge repositories [73]. Given the private nature of the documentation that is in place (and the hesitance of Russian developers to maintain "official" documentation in central databases, respectively), it would be probably more important to share small chunks of information that provide hints towards what is going on in the company and who could be asked for which problems than to provide fully self-sufficient and codified knowledge in form of a public repository. As pointed out above, micro-blogging could be an interesting metaphor to implement a related solution, as this technology seems to combine several advantages of Instant Messengers (informal character, not intrusive, etc.) with a better traceability and visibility [252, 35].

Based on our findings, the conceptional development of articulation spaces would require to account for the following design aspects:

- The different aspects of articulation work and relevant context information of informal, less structured media such as Instant Messengers and social media would need to be identified and aggregated with the formal development IS as well as additional sources like for example usage data (requiring the development of APIs and assistants). For doing so, it would be necessary to analyze the use of Instant Messengers

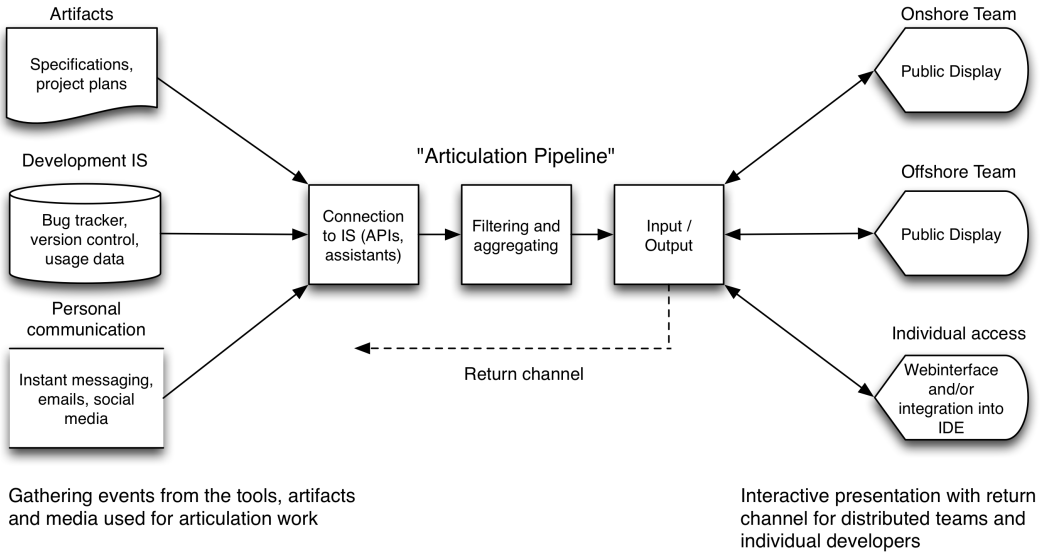

Figure 9.1.: The concept of articulation spaces.

in software development in great detail in order to understand how it relates to the use of formal systems (as we have seen, their use is closely intertwined). If such interconnections could be identified, it would be promising to connect the information that is exchanged in the several systems and make it visible for example by providing assistants in the IDE that help users in bringing important information to the attention of the team (by pushing it into the articulation pipeline).

- Filter mechanisms will be needed to secure the privacy of the developers, especially with regard to media which are regarded as informal and personal (such as Instant Messengers), and to prevent flooding the practitioners with information that they regard as being irrelevant. Further, the socio-cultural background of the cooperation will have to be taken into account. On the input side, this can be accomplished by using assistants rather than generating and publishing information automatically, as we have outlined above. Other systems like the RSS feeds of bug trackers or the version control systems could probably be added without filtering on the input side, but it would be important to control the output of the articulation pipeline in a way that allows users to subscribe to and unsubscribe from particular feeds—always taking into account that individual and team-related configurations have to be distinguished from each other. Again, the use of a micro-blogging service would offer interesting methaphors for this design rationale, as these systems allow

for fine-grained control of what is displayed (filtering by users, groups, hashtags), and offer an unfiltered public timeline at the same time.

- Structured and unstructured information has to be agreggated and put into a context, in order to support awareness of relevant events at team level (with regard to operational and strategic aspects of the cooperation in order to support single- and double-loop learning). For this it would be necessary to analyze where relevant information is exchanged, in order to combine different sources of information with each other in relation to their current context. Hence, the assistants outlined above should take into account that articulation work often spans across sites, artifacts, and media, and help to provide a shared context to these different events perhaps in form of a thread or by tagging them accordingly. At the same time, the presentation device should make these interconnections visible, for example by displaying them in different colors or visualizing the links between the artifacts.
- An interaction concept will be needed that supports the articulation work of the (distributed) team members, allowing to cooperatively reflect and modify coordination mechanisms as well as stimulating learning processes between the teams (in the sense of a "ticket to talk" approach). Hence, it needs to manage several interaction modes for displaying information and working with the display. For example, if the devices were placed in the coffee kitchens of the cooperating companies, it would be promising if the local actors could use the device to cooperatively reflect their results. As these places have turned out to be important spaces for knowledge exchange, the device should probably support different input mechanisms as well as access to the IS systems so that actors can use the display to actively discuss relevant aspects of their work with their colleagues, and feed the results back into the articulation pipeline if required (for example, screenshots of cooperatively created whiteboard sketches). The displays also need to take into account the different perspectives of the teams, as the Russians probably have other interests than German employees (given the division of labor we have encountered in the two companies). At the same time, with regard to supporting inter-organizational knowledge exchange, the display should also show what is relevant for the other team and vice versa. This could also relate to informal, everyday information like the local time, local weather, and local holidays for example.
- Different form factors would have to be tested in different places in the cooperating companies. In order to do so, the social, informal spaces [66] of the company would have to be identified and related to different designs (for example using large or

small displays, experimenting with different interaction concepts, using mobile or stationary displays etc.). While the coffee kitchen seems to be a natural place for putting up a large display, there are also other possibilites that could be explored. For example, displays could be placed in the meeting rooms (implying a more formal character), or in each office room of the company (implying a more personalized character). As the choice of the place will probably have a great impact on how the displays are used, it will be important to experiment with various possibilities and examine the practices that are formed around the different artifacts in practice.

- It has to be ensured that informal aspects of articulation work are not formalized too much by the articulation spaces, because such a change could be perceived as unnecessary bureaucratic burden by the practitioners (possibly even destroying the informal character of the coffee kitchen). At least, such effects need to be taken into account, for example by including information that comes into existence organically within the work process, and offering assistants to further enrich such information. In this regard, it would be interesting to experiment with different presentation modes that imply a "playful" character for the interaction, for example by not showing a chronologic timeline, but by showing information as sticky notes or bubbles that can be manipulated by touch gestures (for examples allowing to burst "bubbles" with irrellevant information, or enlarging important "sticky notes" to make other team members aware of the information). Furthermore, the technology should be thought of as an information offer, without guaranteed retrieval—but with good chances for discovering something interesting or new if explored.

The interactive vizualisation of distributed articulation work is a promising approach of supporting software teams, which would harmonize well with the existing approaches of supporting distributed software development outlined above. According to our findings, supporting awareness about hidden and invisible aspects of team articulation work would improve the implicit, seamless coordination of distributed software teams. At the same time, the interactive visualization would help distributed teams to realize learning potentials which otherwise would not be accessible at all, or would at least lead to high travel expenses. For example, a better awareness about what is going on in the company would probably encourage more direct interaction on part of the distributed team members, which has proven to be of enourmous value for knowledge exchange and coordination in distributed teams. By using the display for presenting the information, information overload could be counteracted because the actors desktops would not be cluttered with awareness information, thus offering interesting implications for the design of social net-

works for software development [18]. At the same time, this would require the solution to follow a "bulletin board" metaphor, as actors could not be sure that information that is put there would necessarily be noticed. While such technologies are not really new given the long history of community bulletin boards and public displays (see for example [46]), their successful application in the domain of distributed software development remains an open challenge. A careful combination of established technologies with the practical and theoretical findings outlined above would offer ample opportunities for supporting the intricate practices of informal articulation work that we have found in the field.

9.3. Outlook

Our research has provided detailed insights into the complexities of distributed software development in two cases of software offshoring of SMEs. As in every research project, there remain open questions and new questions that emerged due to the analysis of the material. Hence, further case studies into distributed software development will be needed; this section is meant to provide some suggestions for possible topics and research questions for future work.

First of all, as the focus of this dissertation intentionally was rather narrow, it would be interesting to see which parallels exist between the cases of software offshoring we found in the SMEs researched by us and other forms of distributed software development. While we have reflected on differences compared to the situation of large software companies, there are further forms like distributed development work of (small-sized) IT departments of larger corporations that would be interesting to compare to our findings. Furthermore, Open Source software development is a big topic also for companies these days, and it would be very interesting to see if the forms of articulation and learning we found in our cases are also important for more unstable cooperations like in the case of Open Source communities or related companies. It will also be important to learn more about how agile development methodologies can be used in distributed contexts, without formalizing them in a way that questions their very foundation as a set of practices that can be combined and adjusted as needed (see chapter 9.2.3).

Second, with regard to the practical implications of our research, there remains the problem that forms of organizational development suggested by us with regard to strategic learning are hard to implement in distributed teams. To our knowledge, there are only few approaches which focus on the important relations of operational and strategic learning in distributed cooperation with regard to the related socio-cultural context. At the

same time, we have seen that the asymmetric power relations of software offshoring can make it very hard to implement changes, even when the practitioners manage to identify possible improvements. Hence, it would be very interesting to investigate possibilities for reconsidering concepts of organization development like *Integrated Organization and Technology Development* (OTE), *Business Ethnography*, or *Cooperative Methods Development* in their potential for supporting learning in distributed organizations (especially with regard to how learning results can be implemented in such fields). Even though their application may be very problematic in distributed contexts, they would have a huge potential for solving many of the underlying challenges and drawbacks of international cooperation (also, for example, with regard to mitigating the socio-cultural problems that can prevail in such projects, see chapter 9.2.1).

Third, the design implications that we identified need to be tested in practice, especially with regard to the aspects that we have worked out in the previous section. Even though research on awareness support for cooperative work has been around for decades, there are still design problems in place that have not been fully solved so far: most prominently the access to awareness information (privacy issues), and its presentation (information overload). As these problems are probably not universally solvable, we need highly contextualized solutions in order to support the specific work of practitioners. As we have showed, such design projects can be even harder to deal with in distributed contexts where the interpretation of new tools has to be negotiated between teams with often very different perspectives on the cooperation. Furthermore, dealing with the soft, informal aspects of articulation work remains a challenge. On the one hand, such aspects are highly important for distributed cooperation but it is very difficult to combine formal and informal aspects of coordination without formalizing the latter. This may indicate that the dichotomy between formal and informal articulation work practices which has guided our research as an analytic lens may have to be re-thought in a design context. While the analytic distintion between "formal" and "informal" has helped us to identify and explore "invisible" aspects of software development work that have often been overlooked in the management literature on offshoring (see chapter 4.2), from a design perspective it may be more useful to center on the general genesis of these practices (for example how coordination mechanisms are cooperatively created, adapted, and applied in practice), and how they interact with changes of technology and new designs [213, 250]. For the practitioners, it is not important whether a practice is formal or informal; what counts, is if the practice works and helps to deal with the contingencies and challenges of the daily work. Hence, instead of accounting for the informal and formal sides of coordination, the aim would to support work practices in a way that makes them more useful for

the actors—regardless of whether these practices are changed towards a more formal, or more informal way. In a way, the informal aspects of articulation work seem to elude a formalization anyways (see [200]), as we have also seen in the context of how practitioners used formal development IS like bug tracking systems (see chapter 5). Thus, it is likely that new forms of (informal) coordination emerge with the adoption of new technologies, which would offer new possibilities for support, and so on.

Last but not least, the methodological challenge of researching distributed forms of cooperation remains. Even though we tried to grasp the multiple perspectives of the actors involved by visiting the remote sites, observing visits and interchanges of developers, and conducting interviews with all actors involved, our access to the field remained more or less focused on the German side of the cooperation (see chapter 9.2.1). Careful analysis was necessary in order to avoid a bias in our conclusions as far as possible, and we see a great potential for future research projects in working with multiple researchers who synchronously observe work practices where they happen, at the cooperating sites. This would probably allow for a far more detailed analysis of the different perspecties, expectations, and interpretations of cooperative work, and thus could considerably contribute to deepen our understanding of distributed software development.

Bibliography

[1] Ackerman, M. S., Pipek, V., and Wulf, V. *Sharing expertise: beyond knowledge management*. MIT Press, Cambridge, Mass., 2003.

[2] Agerfalk, P. J., and Fitzgerald, B. Flexible and distributed software processes: old petunians in new bowls? *Communications of the ACM 49*, 10 (2006), 27–34.

[3] Amberg, M., Gräf, L., and Wiener, M. *Modelle für die Outsourcing-Entscheidung von Softwareentwicklungsprojekten*, vol. 2. Bonn, 2005, pp. 248–252.

[4] Amberg, M., Schröder, M., and Wiener, M. Competence-Based IT Outsourcing - An Evaluation of Models for Identifying and Analyzing Core Competences. In *Proceeding of the Eleventh Americas Conference on Information Systems* (Omaha, 2005), pp. 2702–2712.

[5] Anderson, R. *Work, Ethnography, and System Design*. Marcel Decker, New York, 1997, pp. 159–183.

[6] Aranda, G. N., Chechich, A., Vizcaíno, A., and Piattini, M. Technology Selection to Improve Global Collaboration. In *Proceedings of the IEEE Conference on Global Software Engineering* (2006).

[7] Aranda, J., Easterbrook, S., and Wilson, G. Requirements in the wild: How small companies do it. In *Proceedings of the 15th IEEE International Requirements Engineering Conference* (Dehli, 2007), IEEE Computer Society, pp. 39–48.

[8] Aranda, J., and Venolia, G. The secret life of bugs: Going past the errors and omissions in software repositories. In *Proceedings of the IEEE 31st International Conference on Software Engineering* (Vancouver, 2009), IEEE Computer Society, pp. 298–308.

[9] Argyris, C., Putnam, R., and Smith, D. M. *Action science*. The Jossey-Bass social and behavioral science series. Jossey-Bass, San Francisco, 1985.

[10] Aspray, W., Mayadas, F., and Vardi, M. Y. Globalization and Offshoring of Software: A Report of the ACM Job Migration Task Force. Tech. rep., 2006.

[11] Avram, G. Developing Outsourcing Relationships: A Romanian Service Provider Perspective. In *TProceedings of the first Information Systems Workshop on Global Sourcing - Services, Knowledge and Innovation* (Val d'Isere, France, 2007).

[12] Bannon, L. J., and Schmidt, K. CSCW: Four characters in search of a context. In *Proceedings of the First European Conference on Computer Supported Cooperative Work (ECSCW)* (Gatwick, London, Sept. 1989), pp. 358–372.

[13] Baranano, A. M., Bommer, M., and Jalajas, D. S. Sources of innovation for high-tech SMEs: a comparison of USA, Canada, and Portugal. *International Journal of Technology Management 30*, 1-2 (2005), 205–219.

[14] Bardram, J. Activity-based Computing Support for Agile and Global Software Development. In *Proceedings of the CSCW 2008 Workshop on Supporting Distributed Team Work* (2008), pp. 1–5.

[15] Baskerville, R. L. Investigating information systems with action research. *Communications of the AIS 2*, 3 (1999).

[16] Battin, R. D., Crocker, R., Kreidler, J., and Subramanian, K. Leveraging Resources in Global Software Development. *IEEE Software 18*, 2 (2001), 70–77.

[17] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 1999.

[18] Begel, A., and DeLine, R. Codebook: Social networking over code. In *Proceedings of the IEEE 31st International Conference on Software Engineering* (Vancouver, 2009), pp. 263–266.

[19] Begel, A., DeLine, R., and Zimmermann, T. Social Media for Software Engineering. In *Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research (FoSER)* (2010).

[20] Biehl, J., Czerwinski, M., Smith, G., and Robertson, G. FASTDash: a visual dashboard for fostering awareness in software teams. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems* (2007), ACM, pp. 1313–1322.

[21] Billings, M., and Watts, L. A. A safe space to vent: Conciliation and conflict in distributed teams. In *Proceedings of the 10th European Conference on Computer Supported Cooperative Work* (Limerick, 2007), pp. 139–158.

[22] Bjornson, F. O., and Dingsoyr, T. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology 50*, 11 (Oct. 2008), 1055–1068.

[23] BMBF (Bundesministerium für Bildung und Forschung). Analyse und Evaluation der Softwareentwicklung in Deutschland: Endbericht an das Bundesministerium für Bildung und Forschung. Tech. rep., 2000.

[24] Boden, A., and Avram, G. Bridging Knowledge Distribution - The Role of Knowledge Brokers in Distributed Software Development Teams. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)* (Vancouver, Canada, 2009), pp. 8–11.

[25] Boden, A., Avram, G., Bannon, L., and Wulf, V. Knowledge Management in Distributed Software Development Teams - Does Culture Matter? In *Proceedings of the Fourth IEEE International Conference on Global Software Engineering (ICGSE 2009)* (Limerick, Ireland, 2009), pp. 18–27.

[26] Boden, A., Avram, G., Bannon, L., and Wulf, V. Knowledge Sharing Practices and the Impact of Cultural Factors: Lessons from Two Case Studies of Offshoring in SME. *Software Maintenance and Evolution: Research and Practice* (2010).

[27] Boden, A., Draxler, S., and Wulf, V. Aneignungspraktiken von Software-Entwicklern beim Offshoring. Fallstudie eines kleinen deutschen Softwareunternehmens. In *Multikonferenz Wirtschaftsinformatik (MKWI)* (Göttingen, 2010), pp. 755–766.

[28] Boden, A., Müller, C., and Nett, B. Conducting Business Ethnography in Global Software Development Projects of Small German Enterprises. *Information and Software Technology 53/9* (2011), 1012–1021.

[29] Boden, A., Nett, B., and Wulf, V. Coordination Practices in Distributed Software Development of Small Enterprises. In *Proceedings of the Second IEEE International Conference on Global Software Engineering (ICGSE 2007)* (2007), pp. 235–246.

[30] Boden, A., Nett, B., and Wulf, V. Articulation work in small-scale offshore software development projects. In *Proceedings of the 2008 ICSE workshop on Co-*

operative and human aspects of software engineering (CHASE) (Leipzig, Germany, 2008), ACM, pp. 21–24.

[31] Boden, A., Nett, B., and Wulf, V. Trust and Social Capital: Revisiting an Offshoring Failure Story of a Small German Software Company. In *Proceedings of the Eleventh Europaean Conference on Computer Supported Cooperative Work (ECSCW 2009)* (London, 2009), Springer, pp. 123–142.

[32] Boden, A., Nett, B., and Wulf, V. Operational and Strategic Learning in Global Software Development - Implications from two Offshoring Case Studies in Small Enterprises. *IEEE Software 27*, 6 (2010), 58–65.

[33] Bodker, K., Kensing, F., and Simonsen, J. *Participatory IT design: designing for business and workplace realities*. MIT Press, 2004.

[34] Boes, A., and Schwemmle, M. *Herausforderung Offshoring. Internationalisierung und Auslagerung von IT-Dienstleistungen*. Böckler Stiftung, Düsseldorf, 2004.

[35] Bougie, G., Starke, J., Storey, M.-A., and German, D. M. Towards Understanding Twitter Use in Software Engineering: Preliminary Findings, Ongoing Challenges and Future Questions. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering* (2011).

[36] Bourdieu, P. *The form of capital*. Greenwood Press, New York, 1986.

[37] Brodman, J. G., and Johnson, D. L. What small businesses and small organizations say about the CMM. In *Proceedings of the IEEE International Conference on Software Engineering* (Sorento, 1994), pp. 331–340.

[38] Broughton, M., Paay, J., Kjeldskov, J., O'Hara, K., Li, J., Phillips, M., and Rittenbruch, M. Being here: designing for distributed hands-on collaboration in blended interaction spaces. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group on Design* (New York, New York, USA, 2009), OZCHI '09, ACM Press, p. 73.

[39] Brügge, B., Harhoff, D., Picot, A., Creighton, O., Fiedler, M., and Henkel, J. *Open-Source-Software. Eine ökonomische und technische Analyse*. Springer, Berlin, Heidelberg, 2004.

[40] Calvo-Manzano Villalón, J. A., Cuevas Agustín, G., San Feliu Gilabert, T., De Amescua Seco, A., García Sánchez, L., and Pérez Cota,

M. Experiences in the Application of Software Process Improvement in SMEs. *Software Quality Journal 10*, 3 (Nov. 2002), 261–273.

[41] Carmel, E., and Abbott, P. Why 'nearshore' means that distance matters. *Communications of the ACM 50*, 10 (Oct. 2007), 40–46.

[42] Casey, V. Leveraging or Exploiting Cultural Difference? In *Proceedings of the Fourth IEEE International Conference on Global Software Engineering (ICGSE 2009)* (2009), pp. 8–17.

[43] Cataldo, M., and Herbsleb, J. D. Communication networks in geographically distributed software development. In *Proceedings of the ACM 2008 Conference on Computer Supported Cooperative Work* (San Diego, CA, USA, 2008), ACM, pp. 579–588.

[44] Cataldo, M., Wagstrom, P. A., Herbsleb, J. D., and Carley, K. M. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. In *Proceedings of the 20th ACM Conference on Computer Supported Cooperative Work* (Banff, 2006), pp. 353–362.

[45] Cohen, D., and Prusak, L. *In good company: how social capital makes organizations work*. Harvard Business School Press, Boston, 2001.

[46] Colstad, K., and Lipkin, E. Community memory: a public information network. *ACM SIGCAS Computers and Society 6*, 4 (Dec. 1975), 6–7.

[47] Crabtree, A., O'Neill, J., Tolmie, P., Castellani, S., Colombino, T., and Grasso, A. The Practical Indispensability of Articulation Work to Immediate and Remote Help-giving. In *Proceedings of the Conference on Computer Supported Cooperative Work* (Banff, Alberta, 2006), pp. 219–229.

[48] Cramton, C., and Hinds, P. Intercultural Interaction in Distributed Teams: Salience of and Adaptations to Cultural Differences. In *Academy of Management Best Paper Proceedings* (Philadelphia, 2007).

[49] Cumps, B., Viaene, S., Dedene, G., and Vandenbulcke, J. A Theoretical Exploration of the Relationship between Outsourcing and Business/ICT Alignment. In *Proceedings of the 2006 European Conference on Information Systems* (Gotenborg, Sweden, 2006).

[50] Dahlberg, T., Nyrhinen, M., and Santonen, T. The Success of Selective and Total Outsourcing of firm-wide IT-Infrastructure: An Empirical Evaluation. In

Proceedings of the 14th European Conference on Information Systems (Göteborg, 2006), pp. 1–12.

[51] Damian, D. Requirements Engineering in Distributed Projects. In *Proceedings of the IEEE Conference on Global Software Engineering* (2006).

[52] Damian, D., and Zowghi, D. An Insight into the Interplay between Culture, Conflict and Distance in Globally Distributed Requirements Negotiations. In *Proceedings of the Hawaii International Conference on System Sciences* (Hawaii, 2003), pp. 19–29.

[53] Damian, D. D., and Deependra, M. Global Software Development: How Far Have We Come? *IEEE Software 23*, 5 (2006), 17–19.

[54] Damian, D. D., Lanubile, F., and Mallardo, T. The role of asynchronous discussions in increasing the effectiveness of remote synchronous requirements negotiations. In *Proceedings of the International Conference on Software Engineering* (Shanghai, 2006), pp. 917–920.

[55] Davis, G. B., King, W. R., Ein-Dor, P., and Torkzadeth, R. Information Technology Offshoring: Prospects, Challenges, Educational Requirements, and Curriculum Implications. In *Proceedings of the Twenty-Fifth Conference on Information Systems* (2004), pp. 1027–1038.

[56] DePaula, R., Fischer, G., and Ostwald, J. Courses as Seeds: Expectations and Realities. In *Proceedings of the Second European Conference on Computer-Supported Collaborative Learning* (2001), pp. 494–501.

[57] Desouza, K. C., Awazu, Y., and Baloh, P. Managing Knowledge in Global Software Development Efforts: Issues and Practices. *IEEE Software 23*, 5 (2006), 30–37.

[58] Dibbern, J. *The Sourcing of Application Software Services. Empirical Evidence of Cultural, Industry and Functional Differences.* Physica, Univ. Diss. Bayreuth 2003, Heidelberg, 2004.

[59] Dibbern, J., Goles, T., Hirschheim, R., and Jayatilaka, B. Informations systems outsourcing: a survey and analysis of the literature. *ACM SIGMIS Database 35*, 4 (2004), 6–102.

[60] Dibbern, J., and Heinzl, A. *Selective Outsourcing of Information Systems in Small and Medium Sized Enterprises.* Springer, Berlin, Heidelberg, 2006, pp. 57–81.

[61] Dibbern, J., and Heinzl, A. Outsourcing of Information Systems Functions in Small and Medium Sized Enterprises: A Test of a Multi-Theoretical Model. *Business & Information Systems Engineering 1*, 1 (Dec. 2008), 101–110.

[62] Dibbern, J., Wynne, C. W., and Heinzl, A. The Impact of Human Asset Specificity on the Sourcing of Application Services. In *Proceedings of the European Conference on Information Systems* (2005), pp. 1–14.

[63] Dittrich, Y. *Doing Empirical Research in Software Engineering - finding a path between understanding, intervention, and method development.* MIT Press, 2002, pp. 243–262.

[64] Dittrich, Y., Rönkkö, K., Eriksson, J., Hansson, C., and Lindeberg, O. Cooperative Method Development. Combining qualitative empirical research with method, technique and process improvement. *Empirical Software Engineering 13* (2008), 231–260.

[65] Dourish, P. Implications for design. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (Montréal, Québec, Canada, 2006), ACM, pp. 541–550.

[66] Dourish, P. Re-space-ing place: "place" and "space" ten years on. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (Banff, Alberta, Canada, 2006), ACM, pp. 299–308.

[67] Dourish, P., and Bellotti, V. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work* (New York, New York, USA, 1992), ACM Press, pp. 107–114.

[68] Durissini, M., Nett, B., and Wulf, V. *Kompetenzentwicklung in kleinen Unternehmen der Softwarebranche. Zur Praxisorientierung im Software Engineering.* Oldenbourg, Munich, 2005, pp. 91–100.

[69] Eisenhardt, K. M., and Martin, J. A. Dynamic capabilities: what are they? *Strategic Management 21/10-11* (2000), 1105–1121.

[70] Espinosa, J., Slaughter, S., Kraut, R., and Herbsleb, J. Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal on Management of Information Systems 24*, 1 (2007), 135–169.

[71] Faergemann, L., Schilder-Knudson, T., and Carstensen, P. H. The Duality of Articulation Work in Large Heterogeneous Settings - a Study in Health

Care. In *Proceedings of the Ninth European Conference on Computer-Supported-Cooperative-Work* (2005), pp. 163–183.

[72] Ferneley, E., and Bell, F. Tinker, Tailor: Information Systems and Strategic Development in Knowledge-Based SMEs. In *Proceedings of the European Conference on Information Systems* (2005), pp. 1–12.

[73] Fields, B., Amaldi, P., and Tassi, A. Representing collaborative work: the airport as common information space. *Cognition, Technology & Work 7*, 2 (June 2005), 119–133.

[74] Fischer, G. Symmetry of Ignorance, Social Creativity, and Meta-Design. *Special Issue on "Creativity & Cognition" of the International Journal on Knowledge-Based Systems 13*, 7-8 (2000), 527–537.

[75] Fisher, J., Hirschheim, R., and Jacobs, R. Understanding the outsourcing learning curve: A longitudinal analysis of a large Australian company. *Information Systems Frontiers 10*, 2 (Apr. 2008), 165–178.

[76] Fjuk, A., Smordal, O., and Nurminen, M. I. Taking Articulation Work Seriously - an activity theoretical approach. Tech. rep., Turku Centre for Computer Science, 1997.

[77] Floyd, C. *Software Development as Reality Construction.* Springer, Berlin, 1992, pp. 86–100.

[78] Friedewald, M., Rombach, H. D., Stahl, P., Broy, M., Hartkopf, S., Kimpeler, S., Kohler, K., Wucher, R., and Zoche, P. Softwareentwicklung in Deutschland. Eine Bestandsaufnahme. *Informatik Spektrum 24*, 2 (2001), 81–90.

[79] Fuchs, L. *Situationsorientierte Unterstützung von Gruppenwahrnehmung in CSCW-Systemen.* Univ. Diss Essen, 1998.

[80] Fuchs, L. AREA: a cross-application notification service for groupware. In *Proceedings of the Europaean Conference on Computer Supported Cooperative Work* (Sept. 1999), pp. 61–80.

[81] Fuchs, L., Pankoke-Babatz, U., and Prinz, W. Supporting cooperative awareness with local event mechanisms: the groupdesk system. In *Proceedings of the fourth European Conference on Computer-Supported Cooperative Work* (Sept. 1995), pp. 247–262.

[82] Fuchs, L., Sohlenkamp, M., Genau, A., Kahler, H., Pfeifer, A., and Wulf, V. Transparenz in kooperativen Prozessen: Der Ereignisdienst in PO-

LITeam. In *Proceedings of the Deutsche Computer Supported Cooperative Work* (1996), pp. 3–16.

[83] Gallivan, M., and Srite, M. Information technology and culture: Identifying fragmentary and holistic perspectives of culture. *Information and Organization 15* (2005), 295–338.

[84] Garfinkel, H., and Bittner, E. *"Good" organizational reasons for "bad" clinic records.* Prentice-Hall, New Jersey, 1967, pp. 186–207.

[85] Geertz, C. *The Interpretation Of Cultures.* Basic Books, 1973.

[86] Geertz, C. *Thick Description: Towards an Interpretive Theory of Culture.* Basic Books, New York, 1973, pp. 3–30.

[87] Generaldirektion Unternehmen der Kommission der Europäischen Gemeinschaft. Internationalisierung von KMU. Tech. rep., Brussels, 2003.

[88] Gerson, E. M. *Reach, Bracket, and the Limits of Rationalized Coordination: Some Challenges for CSCW.* No. 2003. Springer, 2008, pp. 193–220.

[89] Gerson, E. M., and Star, S. L. Analyzing due process in the workplace. *ACM Transactions on Office Information Systems 4*, 3 (1986), 257–270.

[90] Geyer, W., Richter, H., Fuchs, L., Frauenhofer, T., Daijavad, S., and Poltrock, S. A team collaboration space supporting capture and access of virtual meetings. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work - GROUP '01* (New York, Sept. 2001), ACM Press, p. 188.

[91] Gobo, G. *Doing Ethnography.* Sage Publications, Los Angeles, 2008.

[92] Gotel, O., Kulkarni, V., Say, M., Scharff, C., and Sunetnanta, T. Quality Indicators on Global Software Development Projects: Does "Getting to Know You" Really Matter? In *Proceedings of the fourth IEEE International Conference on Global Software Engineering (ICGSE 2009)* (2009), pp. 3–7.

[93] Granovetter, M. Economic Action and Social Structure: The Problem of Embeddedness. *American Journal of Sociology 91*, 3 (1985), 481–510.

[94] Grinter, R. Using a configuration management tool to coordinate software development. In *Proceedings of the Conference on Organizational Computing Systems* (New York, 1995), ACM, pp. 168–177.

[95] Grinter, R. E. Supporting Articulation Work Using Software Configuration Management Systems. *Computer Supported Cooperative Work 5* (1996), 447–465.

[96] Grinter, R. E. Doing Software Development: Occasions for Automation and Formalisation. In *Proceedings of the European Conference on Computer Supported Cooperative Work* (Lancaster, 1997), pp. 173–188.

[97] Grinter, R. E., Herbsleb, J. D., and Perry, D. E. The geography of coordination: dealing with distance in R&D work. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work* (Phoenix, Arizona, United States, 1999), ACM, pp. 306–315.

[98] Gross, T., and Koch, M. *Computer-Supported Cooperative Work*. Oldenbourg Wissenschaftsverlag, 2007.

[99] Gutwin, C., Penner, R., and Schneider, K. Group awareness in distributed software development. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work* (Chicago, 2004), ACM, pp. 72–81.

[100] Handel, M., and Herbsleb, J. D. What is Chat Doing in the Workplace? In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (New Orleans, 2002), pp. 1–10.

[101] Handel, M., and Poltrock, S. Working Around Official Applications : Experiences from a Large Engineering Project. In *Proceedings of the 2011 Conference on Computer Supported Cooperative Work* (2011), pp. 309–312.

[102] Heath, C., and Luff, P. Collaboration and Control. Crisis Management and Multimedia Technology in London Underground Line Control Rooms. *Computer Supported Cooperative Work 1* (1992), 69–94.

[103] Hebdige, D. *Subculture the meaning of style*, repr. ed. Methuen, London [u.a.], 1981.

[104] Heinzl, A., and Dibbern, J. Mittelständische Unternehmen haben ein großes Outsourcing-Potential. *Netzguide, IT Economics & Managed Services 06/2002* (2002), 15–17.

[105] Herbsleb, J. Global Software Engineering: The Future of Socio-technical Coordination. In *2007 Future of Software Engineering* (Minneapolis, MN, USA, 2007), IEEE Computer Society, pp. 188–198.

[106] Herbsleb, J. D., Finholt, T. A., and Grinter, R. E. An Empirical Study of Global Software Development: Distance and Speed. In *Proceedings of the International Conference on Software Engineering* (Toronto, 2001), pp. 81–90.

[107] Herbsleb, J. D., Mockus, A., Finholt, T. A., and Grinter, R. E. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (Philadelphia, 2000), pp. 319–328.

[108] Herbsleb, J. D., Paulish, D. J., and Bass, M. Global Software Development at Siemens: Experience from Nine Projects. In *Proceedings of the International Conference on Software Engineering* (St. Louis, 2005), pp. 524–533.

[109] Hildenbrand, T., Rothlauf, F., Geisser, M., Heinzl, A., and Kude, T. Approaches to Collaborative Software Development. In *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems* (Mar. 2008), IEEE, pp. 523–528.

[110] Hinds, P., and Mcgrath, C. Structures that work: social structure, work structure and coordination ease in geographically distributed teams. In *Proceedings of the 20th Conference on Computer Supported Cooperative Work* (Banff, 2006), pp. 343–352.

[111] Hinds, P. J., and Mortensen, M. Understanding Conflict in Geographically Distributed Teams: The Moderating Effects of Shared Identity, Shared Context, and Spontaneous Communication. *Organization Science 16*, 3 (May 2005), 290–307.

[112] Hine, C. *Virtual ethnography*. Sage Publications Ltd, 2000.

[113] Hirschheim, R., Dibbern, J., and Heinzl, A. Foreword to the special issue on IS sourcing. *Information Systems Frontiers 10*, 2 (Apr. 2008), 125–127.

[114] Hirschheim, R. A., Heinzl, A., and Dibbern, J. *Information systems outsourcing: enduring themes, emergent patterns, and future directions*. Springer-Verlag, Berlin; New York, 2002.

[115] Hirschheim, R. A., Newman, M., Loebbecke, C., and Valor, J. Offshoring and its Implications for the Information Systems Discipline. In *Proceedings of the Twenty-Sixth International Conference on Information Systems* (2005), pp. 1003–1018.

[116] Hoffmann, B., and Wulf, V. Building Communities among Software Engineers: The ViSEK Apporach to Intra- and Inter-Organizational Learning. In *International Workshop on Learning Software Organizations (LSO 2002)* (Heidelberg, 2002), S. Henninger and F. Maurer, Eds., pp. 32ff.–32ff.

[117] Hofstede, G. H. *Culture's consequences: comparing values, behaviors, institutions, and organizations across nations*. Sage Publications, Thousand Oaks, Calif., 2001.

[118] Hsieh, Y. Culture and Shared Understanding in Distributed Requirements Engineering. In *Proceedings of the IEEE International Conference on Software Engineering* (2006), IEEE Computer Society, pp. 101–108.

[119] Huysman, M., and de Wit, D. Practise of Managing Knowledge Sharing: Towards a Second Wave of Knowledge Management. *Knowledge and Process Management 11/2* (2004), 81–92.

[120] Huysman, M., and Wulf, V. *Social capital and information technology*. MIT Press, Cambridge, Mass., 2004.

[121] Iivari, J., Hirschheim, R. A., and Klein, H. K. Towards More Professional Information Systems Development: ISD as Knowledge Work. In *Proceedings of the 9th Europaean Conference on Information Systems* (Bled, 2001), pp. 1025–1036.

[122] Imsland, V. *The Role of Trust in Global Software Outsourcing Relationships*. Ph.d. thesis, University of Oslo, 2003.

[123] Johri, A. Look Ma, No Email! Blogs and IRC as Primary and Preferred Communication Tools in a Distributed Firm. In *Proceedings of the 2011 Conference on Computer Supported Cooperative Work* (Hangzhou, China, 2011), pp. 305–308.

[124] Kämpgen, B., Ell, B., Simperl, E., Vrandecic, D., and Dengler, F. Enterprise Wikis: Technical Challenges and Opportunities. In *Proceedings of the 6th Conference on Professional Knowledge Management* (2011).

[125] Keil, P. Principal Agent Theory and its Application to Analyze Outsourcing of Software Development. In *Proceedings of the International Workshop on Economics-Driven Software Engineering Research* (St. Louis, 2005).

[126] King, W. R., and Torkzadeth, G. Information Systems Offshoring: Research Status and Issues. *MIS Quarterly 32/2* (2008).

[127] Knoblauch, H. Fokussierte Ethnographie. *Sozialer Sinn*, 1 (2001), 123–141.

[128] Kobylinski, R. *Building Group Awareness in Distributed Software Development Projects*. PhD thesis, 2005.

[129] Koch, C. AT&T Wireless Self-Destructs as Offshore IT Outsourcing Disaster kills Company. *CIO Magazine, 15. April 2004* (2004).

[130] Koch, M., and Richter, A. *Enterprise 2.0: Planung, Einführung und erfolgreicher Einsatz von Social Software in Unternehmen*. Oldenbourg, Oct. 2007.

[131] Kotlarsky, J., and Oshri, I. Social Ties, Knowledge Sharing and Successful Collaboration in Globally Distributed System Development Projects. *Europaean Journal of Information Systems 14/1* (2005), 37–48.

[132] Kraut, R. E., Fussell, S. R., Brennan, S. E., and Siege, J. *Understanding effects of proximity on collaboration: Implications for technologies to support remote collaborative work*. Cambridge, 2002, pp. 137–162.

[133] Krishna, S., Sahay, S., and Walsham, G. Managing Cross-Cultural Issues in Global Software Outsourcing. *Communications of the ACM 47*, 4 (2004), 62–66.

[134] Lacity, M., and Willcocks, L. *Global information technology outsourcing: In search of business advantage*. Wiley, 2001.

[135] Lacity, M. C. Lessons in Global Information Technology Sourcing. *Computer 35*, 8 (2002), 26–33.

[136] Lacity, M. C., and Hirschheim, R. A. *Information systems outsourcing: myths, metaphors, and realities*. Wiley, Chichester, New York, 1993.

[137] Lacity, M. C., and Hirschheim, R. A. *Beyond the information systems outsourcing bandwagon: the insourcing response*. Wiley, Chichester, New York, 1995.

[138] Lanubile, F., Ebert, C., Prikladnicki, R., and Vizcaino, A. Collaboration Tools for Global Software Engineering. *IEEE Software 27*, 2 (Mar. 2010), 52–55.

[139] Lave, J., and Wenger, E. *Situated learning: legitimate peripheral participation*. Learning in doing. Cambridge University Press, Cambridge, New York, 1991.

[140] Lee, G., Delone, W., and Espinosa, J. A. Ambidextrous coping strategies in globally distributed software development projects. *Communications of the ACM 49*, 10 (2006), 35–40.

[141] Lee, J.-N., Huynh, M., and Hirschheim, R. An integrative model of trust on IT outsourcing: Examining a bilateral perspective. *Information Systems Frontiers 10*, 2 (Apr. 2008), 145–163.

[142] Lee, O.-K., Banerjee, P., Lim, K. H., Kumar, K., van Hillegersberg, J., and Wei, K. K. Aligning IT components to archieve agility in globally distributed system development. *Communications of the ACM 49*, 10 (2006), 49–54.

[143] Lethbridge, T., and Singer, J. How Software Engineers Use Documentation: The State of the Practice. *IEEE Software 20/6* (2003).

[144] Levina, N., and Vaast, E. Innovating or Doing as Told? Status Differences and Overlapping Boundaries in Offshore Collaboration. *MIS Quarterly 32/2* (2008).

[145] Liao, Q., Pan, S., Lai, J. C., and Yang, C. Enterprise Blogging in a Global Context : Comparing Chinese and American Practices. In *Proceedings of the 2011 Conference on Computer Supported Cooperative Work* (2011), pp. 35–44.

[146] MacGregor, E., Hsieh, Y., and Kruchten, P. Cultural Patterns in Software Process Mishaps: Incidents in Global Projects. In *Proceedings of the International Conference on Software Engineering* (St. Louis, 2005), pp. 1–5.

[147] Mackay, W. Patterns of sharing customizable software. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (Los Angeles, California, United States, 1990), ACM, pp. 209–221.

[148] Mahnke, V., Overby, M. L., and Vang, J. Strategic Outsourcing of IT Services: Theoretical Stocktaking and Empirical Challenges. *Industry and Innovation 12*, 2 (2005), 205–253.

[149] Marczak, S., Damian, D., Stege, U., and Schröter, A. Information Brokers in Requirement-Dependency Social Networks. In *Proceedings of the 2008 16th IEEE International Requirements Engineering Conference* (2008), IEEE Computer Society, pp. 53–62.

[150] Marczak, S., Kwan, I., and Damian, D. Investigating Collaboration Driven by Requirements in Cross-Functional Software Teams. In *2009 Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills* (Aug. 2009), IEEE, pp. 15–22.

[151] McSweeney, B. Hofstede's Model of National Cultural Differences and Their Consequences: A Triumph of Faith - A Failure of Analysis. *Human Relations 55/1* (2002), 89–118.

[152] Mertens, P., Gross e Wilde, J., and Wilkens, I. *Die (Aus-)Wanderung der Softwareproduktion. Eine Zwischenbilanz.* Arbeitsberichte des Instituts für Informatik 38/3. Erlangen, 2005.

[153] Milewski, A. E., Tremaine, M., Egan, R., Zhang, S., Kobler, F., and O'Sullivan, P. Guidelines for Effective Bridging in Global Software Engineering. In *Proceedings of the International Conference on Global Software Engineering* (Los Alamitos, CA, USA, 2008), pp. 23–32.

[154] Mockus, A., and Weiss, D. M. Globalization by Chunking: Quantitative Approach. *IEEE Software 18*, 2 (2001), 30–37.

[155] Moe, N. B., Dingsø yr, T., and Dybâ, T. Overcoming Barriers to Self-Management in Software Teams. *IEEE Software 26*, 6 (2009), 20–26.

[156] Morgan, G. *Images of Organization*. London, New Delhi, 1996.

[157] Narayanan, S., Mazumder, S., and Raju, R. Success of Offshore Relationships: Engineering team structures. In *Proceedings of the International Conference on Global Software Engineering* (Costo do Santinho, Brazil, 2006), IEEE, pp. 73–82.

[158] Nett, B. *Am Ende "Großer Theorien"? Überlegungen zur soziologischen Theoriediskussion*. LIT, 1998.

[159] Nett, B. *Konstruktion und Rekonstruktion von Technik. Business Ethnography als reflexives Forschungsdesign für Technikentwicklungsprojekte*. Habil. Univ. Siegen, Siegen, 2011.

[160] Nett, B., and Durissini, M. Wissensprozesse in kleinen Unternehmen der Softwarebranche aus der Sicht von Entwicklern. Tech. rep., 2004.

[161] Nett, B., and Wulf, V. *Wissensprozesse in der Softwarebranche. Kleine und mittelständische Unternehmen unter empirischer Perspektive*. Transcript, Bielefeld, 2005, pp. 147–168.

[162] Nicholson, B., and Sahay, S. Some political and cultural issues in the globalization of software development: case experience from Britain and India. *Information and Organization 11* (2001), 25–43.

[163] Nicholson, B., and Sahay, S. Embedded knowledge and offshore software development. *Information and Organization 14/4* (2004), 329–365.

[164] OECD. Information and Communication Technologies. OECD Information Technology Outlook: 2004 Edition. Tech. rep., 2004.

[165] Olson, G., and Olson, J. Distance Matters. *Human-Computer Interaction 15* (2000), 139–178.

[166] Orlikowski, W. J. Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organization Science 13*, 3 (May 2002), 249–273.

[167] Oshrin, I., Kotlarsky, J., and Willcocks, L. Before, During, and After Face-to-Face Meetings: The Lifecycle of Social Ties in Globally Distributed Teams. In *Proceedings of the Twenty-Sixth International Conference on Information Systems* (2005), pp. 395–407.

[168] Paasivaara, M., and Lassenius, C. Could Global Software Development Benefit from Agile Methods? In *International Conference on Global Software Engineering* (Costo do Santinho, Brazil, 2006), pp. 1–5.

[169] Peirce, C. S. *The Essential Peirce, Volume 2: Selected Philosophical Writings, 1893-1913*. Indiana University Press, 1998.

[170] Perry, D. E., Siy, H. P., and Votta, L. G. Parallel Changes in Large-Scale Software Development: An Observational Case Study. *ACM Transactions on Software Engineering and Methodology 10*, 3 (2001), 308–337.

[171] Philip, T., Schwabe, G., and Ewusi-Mensah, K. Critical Issues of Offshore Software Development Project Failures. In *Proceedings of the International Conference on Information Systems* (Phoenix, Arizona, Jan. 2009).

[172] Polanyi, M. *The tacit dimension*. Routledge & K. Paul, London, 1967.

[173] Prikladnicki, R. Exploring Propinquity in Global Software Engineering. In *Proceedings of the Fourth IEEE International Conference on Global Software Engineering* (2009), pp. 133–142.

[174] Putnam, R. D. *Bowling Alone: The Collapse and Revival of the American Community*. Simon & Schuster, New York, 2000.

[175] Ramesh, B., Cao, L., Mohan, K., and Xu, P. Can distributed software development be agile? *Communications of the ACM 49*, 10 (2006), 41–46.

[176] Randall, D., Harper, R., and Rouncefield, M. *Fieldwork for design: theory and practice*. Springer, 2007.

[177] Redmiles, D., Van Der Hoek, A., Al-Ani, B., Hildebrand, T., Quirk, S., Sarma, A., Silva Filho, R. S., de Souza, C., and Trainer, E. Continuous Coordination. A New Paradigm to Support Globally Distributed Software Development Projects. *Wirtschaftsinformatik 49* (2007), 28–38.

[178] Reichling, T., Veith, M., and Wulf, V. Expert Recommender: Designing for a Network Organization. *Computer Supported Cooperative Work 16*, 4 (Oct. 2007), 431–465.

[179] Richardson, I., Avram, G., Deshpande, S., and Casey, V. Having a Foot on Each Shore - Bridging Global Software Development in the Case of SMEs. In *Proceedings of the 2008 IEEE International Conference on Global Software Engineering* (2008), IEEE Computer Society, pp. 13–22.

[180] Richardson, I., and Ryan, K. Software-Process Improvements in a Very Small Company. *Software Quality Professional 3/2* (2001), 22–35.

[181] Riemer, K., Richter, A., and Seltsikas, P. Enterprise Microblogging: Procrastination or productive use? In *Proceedings of the Americas Conference on Information Systems* (Aug. 2010).

[182] Rittel, H. *Second-Generation Design Methods.* John Wiley & Sons, New York, 1984, pp. 317–327.

[183] Rittenbruch, M. Atmosphere: towards context-selective awareness mechanisms. In *Proceedings of HCI International Conference on Human-Computer Interaction* (Munich, Aug. 1999), pp. 328–332.

[184] Rönkkö, K. Ethnography and Distributed Software Development. In *Proceedings of the ICSE Workshop Beg, Borrow or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research* (Limerick, Ireland, 2000).

[185] Rönkkö, K. *Ethnography.* Taylor and Francis Group, New York, 2010.

[186] Rottman, J., and Lacity, M. A US Client's learning from outsourcing IT work offshore. *Information Systems Frontiers 10*, 2 (Apr. 2008), 259–275.

[187] Ruiz Ben, E., and Claus, R. Offshoring in der deutschen IT Branche. Eine neue Herausforderung für die Informatik. *Informatik Spektrum* (2005), 34–38.

[188] Rumelt, R. P. *Strategy, structure, and economic performance.* Harvard Business School classics; 5. Harvard Business School Press, Boston, Mass., 1986.

[189] Sabherwal, R., and Choudhury, V. *Governance of Remotely Outsourced Software Development: A Comparison of Client and Vendor Perspectives.* Springer, Berlin, Heidelberg, 2006, pp. 187–222.

[190] Sargent, A. Outsourcing relationship literature: an examination and implications for future research. In *Proceedings of the 2006 ACM SIGMIS CPR Conference on Computer Personnel Research* (Claremont CA, 2006), ACM, pp. 280–287.

[191] Sarma, A., Maccherone, L., Wagstrom, P., and Herbsleb, J. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In *2009 IEEE 31st International Conference on Software Engineering* (May 2009), IEEE, pp. 23–33.

[192] Sauer, J. Agile Offshore Outsourcing - Concepts and Practices for Flexible Integration of Offshore Development Services. In *Proceedings of the Agile Business Conference* (London, 2006).

[193] Sauer, J. Agile Practices in Offshore Outsourcing - An Analysis of Published Experiences. In *Proceedings of the 29th Information Systems Research* (Helsingoer, 2006).

[194] Sauer, J. *Architekturzentrierte agile Anwendungsentwicklung in global verteilten Projekten.* Univ. Diss. Hamburg, 2010.

[195] Scheitor, D., Müller, W., Leppeck, M., and Reimer, H. *Offshore. Total global?* IG Metall, Munich, 2003.

[196] Schmidt, K. Riding a Tiger, or Computer Supported Cooperative Work. In *Second European Conference on Computer Supported Cooperative Work* (Amsterdam, 1991), L. R. Bannon and K. Schmidt, Eds., pp. 1–16.

[197] Schmidt, K. Of maps and scripts. The status of formal constructs in cooperative work. *Information and Software Technology 41* (1999), 319–329.

[198] Schmidt, K. Asking for the moon: Or model-based coordination in distributed design. In *Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW)* (2009), pp. 383–402.

[199] Schmidt, K. *Cooperative Work and Coordinative Practices: Contributions to the Conceptual Foundations of Computer-Supported Cooperative Work (CSCW).* Springer, 2011.

[200] Schmidt, K., and Bannon, L. Taking CSCW Seriously: Supporting Articulation Work. *Computer Supported Cooperative Work 1*, 1 (1992), 7–40.

[201] Schmidt, K., and Simone, C. Coordinaton Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design. *Computer Supported Cooperative Work 5* (1996), 155–200.

[202] Schmidt, K., and Wagner, I. Ordering Systems: Coordinative Practices and Artifacts in Architectural Design and Planning. *Computer Supported Cooperative Work 13*, 5-6 (Dec. 2004), 349–408.

[203] Schutz, A. *The structures of the lifeworld.* Northwestern Univ. Pr., Evanston Ill., 1983.

[204] Shapiro, D. The limits of ethnography: combining social sciences for CSCW. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW)* (1994), 417–428.

[205] Sieber, A. Kleine Softwareunternehmen und ihre Erfahrungen mit Softwareentwicklung in Osteuropa und Indien. In *Informatik 2006. Informatik für Menschen. Beiträge der 36. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2.-6. Oktober 2006 in Dresden* (Bonn, 2006), C. Hochberger and R. Liskowsky, Eds., vol. 1 of *Lecture Notes in Informatics*, pp. 642–649.

[206] Simone, C., Divitini, M., and Schmidt, K. A notation for malleable and interoperable coordination mechanisms for CSCW systems. In *Conference on Organizational Computing Systems* (Milpitas, California, 1995), pp. 1–12.

[207] Sohlenkamp, M., Prinz, W., and Fuchs, L. PoliawaC: design and evaluation of an awareness-enhanced groupware client. *AI & Society 14*, 1 (Mar. 2000), 31–47.

[208] Spanjers, H., ter Huurne, M., Graaf, B., Lormans, M., Bendas, D., and Van Solingen, R. Tool Support for Distributed Software Engineering. In *Proceedings of the International Conference on Global Software Engineering* (Costo do Santinho, Brazil, 2006), pp. 1–10.

[209] Star, S. L. *The Sociology of the Invisible: The Primacy of Work in the Writings of Anselm Strauss.* Aldine de Gruyter, Hawthorne, 1991, pp. 265–283.

[210] Star, S. L., and Griesemer, J. R. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science 19*, 3 (Aug. 1989), 387–420.

[211] Star, S. L., and Strauss, A. L. Layers of Silence, Areas of Voice: The Ecology of Visible and Invisible Work. *Computer Supported Cooperative Work 8* (1999), 9–30.

[212] Stark, J., Arlt, M., and Walker, D. H. T. Outsourcing Decisions & Models - Some Practical Considerations for Large Organizations. In *Proceedings of the International Conference on Global Software Engineering* (Costo do Santinho, Brazil, 2006), pp. 1–6.

[213] Stevens, G. *Understanding and Designing Appropriation Infrastructures: Artifacts as boundary objects in the continuous software development.* Diss. Univ. Siegen, 2010.

[214] Stevens, G., and Nett, B. *Business Ethnography as a research method to support evolutionary design.* Navigationen. 2009.

[215] Stevens, G., Pipek, V., and Wulf, V. Appropriation Infrastructure: Mediating appropriation and production work. *Special Issue on End-User Development of the International Journal of Organizational and End User Computing 22*, 2 (2010), 58–81.

[216] Stevens, G., and Wiedenhöfer, T. CHIC - a pluggable solution for community help in context. In *Proceedings of the 4th Nordic Conference on Human-Computer Interaction* (Oslo, Norway, 2006), ACM, pp. 212–221.

[217] Stopford, J. M., and Wells, L. T. *Managing the multinational enterprise; organization of the firm and ownership of the subsidiaries.* The Harvard multinational enterprise series. Basic Books, New York, 1966.

[218] Storey, M.-A., Treude, C., van Deursen, A., and Cheng, L.-T. The Impact of Social Media on Software Engineering Practices and Tools. In *Proceedings of the Workshop on the Future of Software Engineering Research* (Santa Fe, 2010), pp. 359–364.

[219] Strauss, A. L. *Social organization of medical work.* University of Chicago Press, Chicago, 1985.

[220] Strauss, A. L. Work and Division of Labor. *The Sociological Quarterly 26*, 1 (1985), 1–19.

[221] Strauss, A. L. The Articulation of Project Work: An Organizational Process. *The Sociological Quarterly 29*, 2 (1988), 163–178.

[222] Strauss, A. L. *Continual permutations of action.* Communication and social order. Aldine de Gruyter, New York, 1993.

[223] Strauss, A. L., and Corbin, J. M. *Basics of qualitative research: techniques and procedures for developing grounded theory.* Sage Publications, Thousand Oaks, 1998.

[224] Strübing, J. *Pragmatistische Wissenschafts- und Technikforschung: Theorie und Methode.* Campus Verlag, 2005.

[225] Stuttgart, L. V., and Schmidt, R. Praktiken des Programmierens. Zur Morphologie von Wissensarbeit in der Software-Entwicklung. *Zeitschrift für Soziologie 37*, 4 (2008), 282–300.

[226] Suchman, L. A. *Plans and situated actions: the problem of human-machine communication*. Cambridge University Press, Cambridge, New York, Port Chester, Melbourne, Sidney, 1987.

[227] Suchman, L. A. Making Work Visible. *Communications of the ACM 38*, 9 (1995), 56–64.

[228] Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. Distributed scrum: Agile project management with outsourced development teams. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences* (2007), IEEE Computer Society.

[229] Takeuchi, H., and Nonaka, I. o. *Hitotsubashi on knowledge management*. John Wiley & Sons (Asia), Singapore, 2004.

[230] Thom-santelli, J., Millen, D. R., and Street, R. Organizational Acculturation and Social Networking. In *Proceedings of the 2011 Conference on Computer Supported Cooperative Work* (2011), pp. 313–316.

[231] Tichy, W., and Höfer, A. Status of Empirical Research in Software Engineering. Tech. rep., 2006.

[232] Timonen, H., and Jalonen, M. A Critical Review of Knowledge Management Literature: Introducing a Practice-based Approach on Knowledge Sharing. In *Proceedings of the 9th European Conference on Knowledge Management* (Southhampton, UK, 2008).

[233] Treude, C., and Storey, M. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *Proceedings of the International Conference on Software Engineering* (2010), ACM, pp. 365–374.

[234] Twidale, M. B. Over the Shoulder Learning: Supporting Brief Informal Learning. *Computer Supported Cooperative Work (CSCW) 14*, 6 (Nov. 2005), 505–547.

[235] Vähäniitty, J., and Rautiainen, K. T. Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development. *Proceedings of the International Conference on Software Engineering* (2008), 25–28.

[236] Volery, T., and Jakl, M. L. *Chancen und Problemfelder der Internationalisierung für KMU - eine Grundsätzliche Betrachtung*. Erich Schmidt Verlag, Berlin, 2006, pp. 1–18.

[237] Weick, K. *Making Sense of the Organisation*. Blackwell, 2001.

[238] Wenger, E. *Communities of practice: learning, meaning, and identity*. Learning in doing. Cambridge University Press, Cambridge, New York, 1998.

[239] Wiener, M. *Critical success factors of offshore software development projects: The perspective of German-speaking companies*. Duv, 2006.

[240] Williamson, O. E., and Masten, S. E. *The economics of transaction costs*. An Elgar critical writings reader. E. Elgar Pub., Cheltenham, UK; Northampton, Mass. USA, 1999.

[241] Williamson, O. E., Winter, S. G., and Coase, R. H. *The Nature of the firm: origins, evolution, and development*. Oxford University Press, New York, 1991.

[242] Wilson, S., Galliers, J., and Fone, J. Not all sharing is equal. In *Proceedings of the Conference on Computer Supported Cooperative Work* (New York, New York, USA, Nov. 2006), ACM Press, pp. 25–28.

[243] Winkler, J., and Dibbern, J. Herausforderungen und Chancen der Internationalisierung für mittelständische Softwareunternehmen in Deutschland. In *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI)* (2008), pp. 751–762.

[244] Winkler, J., Dibbern, J., and Heinzl, A. The impact of cultural differences in offshore outsourcing - Case study results from German-Indian application development projects. *Information Systems Frontiers 10*, 2 (Apr. 2008), 243–258.

[245] Wittel, A. Ethnography on the Move: From Field to Net to Internet. *Forum Qualitative Sozialforschung 1*, 1 (2000).

[246] Womack, J. P. *Die zweite Revolution in der Autoindustrie: Konsequenzen aus der weltweiten Studie aus dem Massachusetts Institute of Technology*. Campus-Verl., Frankfurt/Main [u.a.], 1994.

[247] WPIE (Working Party of the Information Economy). Potential Offshoring of ITC-intensive using Occupations. Tech. rep., 2005.

[248] Wulf, V., and Mark, G. The Emergence of Conventions within Processes of Organizational and Technological Change. In *Proceedings of the International Conference on Human-Computer Interaction* (Amsterdam, 1997), pp. 293–296.

[249] Wulf, V., and Rohde, M. Towards an integrated organization and technology development. *Designing Interactive Systems* (1995), 55.

[250] Wulf, V., and Rohde, M. Engaging with Practices : Design Case Studies as a Research Framework in CSCW. In *Proceedings of the 2011 Conference on Computer Supported Cooperative Work* (2011), pp. 505–512.

[251] Yang, X.-H., and Xu, B. Towards Adaptive Tasks Arrangement in Offshore Outsourcing Software Development. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetiks* (Gouangzhou, 2005), pp. 654–657.

[252] Zhao, D., and Rosson, M. B. How and why people Twitter: the role that micro-blogging plays in informal communication at work. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work* (Sanibel Island, Florida, USA, 2009), ACM, pp. 243–252.

[253] Zolin, R. Interpersonal trust in cross-functional, geographically distributed work: A longitudinal study. *Information and Organization 14*, 1 (Jan. 2004), 1–26.

Appendix I: List of Publications

Parts of this dissertation have already been published as conference or journal papers. The chapters of part II resemble the accepted versions of these publications:

- **Chapter 4:** Boden, A., Müller, C. & Nett, B., 2011. Conducting Business Ethnography in Global Software Development Projects of Small German Enterprises. Information and Software Technology 53/9, pp. 1012-1021.
  http://www.sciencedirect.com/science/article/pii/S095058491100036X
  Publisher: Elsevier B.V.
- **Chapter 5:** Boden, A., Nett, B. & Wulf, V., 2007. Coordination Practices in Distributed Software Development of Small Enterprises. In Proceedings of the Second IEEE International Conference on Global Software Engineering (ICGSE 2007). pp. 235-246.
  http://ieeexplore.ieee.org/search/freesrchabstract.jsp?tp=&arnumber=4299859
  Publisher: IEEE
- **Chapter 6:** Boden, A., Nett, B. & Wulf, V., 2010. Operational and Strategic Learning in Global Software Development - Implications from two Offshoring Case Studies in Small Enterprises. IEEE Software, 27(6), pp. 58-65.
  http://ieeexplore.ieee.org/search/freesrchabstract.jsp?tp=&arnumber=5204064
  Publisher: IEEE
- **Chapter 7:** Boden, A., Nett, B. & Wulf, V., 2009. Trust and Social Capital: Revisiting an Offshoring Failure Story of a Small German Software Company. In Proceedings of the Eleventh Europaean Conference on Computer Supported Cooperative Work (ECSCW 2009). London: Springer, pp. 123-142.
  http://www.springerlink.com/content/978-1-84882-853-7#section=99856&page=1&locus=0
  Publisher: Springer
- **Chapter 8:** Boden, A., Avram, G., Bannon, L. & Wulf, V., 2010. Knowledge Sharing Practices and the Impact of Cultural Factors: Lessons from Two Case

Studies of Offshoring in SME. Software Maintenance and Evolution: Research and Practice, doi: 10.1002/smr.473.
http://onlinelibrary.wiley.com/doi/10.1002/smr.473/abstract
Publisher: John Wiley & Sons, Ltd.