

Coordination Practices in Distributed Software Development of Small Enterprises

Alexander Boden, Bernhard Nett and Volker Wulf

University of Siegen

Information Systems and New Media

alexander.boden | bernhard.nett | volker.wulf@uni-siegen.de

Abstract

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.

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 [1]. 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 [2].

The trend towards global software development has led to a discussion concerning the organizational means of distributed software development [3-6]. 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 [7].

There is still need for further empirical studies as well as theoretical approaches concerning strategies of organizing and managing globally distributed software engineering [4]. 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 [8, 9].

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).

2. Global Software Development in SME

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 [10].

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 in-

curing transaction costs [11]. 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 [12].

This risk appears to be even more relevant to SME because of their financial possibilities that are usually lower [7]. Relying on an unsuitable partner thus can lead to severe repercussions as back-sourcing 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 [13]. 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 [14].

Although different models and taxonomies are proposed [15], 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 [16]. 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.

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.) [17].

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 [18] 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 [19].

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 [20, 21]. However, it seems to be still unclear how those methods can be adapted efficiently in practice [22, 23].

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 [24] to our case studies. Articulation work is of great importance when plans have to be adjusted to unexpected occurrences or changed requirements [25]. 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.

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 [25]. Strauss’ framework aims at a better understanding of the interrelatedness of interdependent actions of cooperating actors [24] and has been of great benefit as a framework for ethnographic research into collaborative work environments [26, 27].

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 [28].

As work is constituted strongly by these mutual interdependencies, Schmidt and Simone [29] 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 [25].

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” [24].

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 [30]: 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, reconciliation and interdependency but also concerning individual interpretations, perspectives and accepted meanings of work [29], in short: all necessary endeavors to manage the distributed nature of cooperative work.

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 [25].

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 [31].

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 articulation work in an objective and deterministic manner [29] 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 [32]. 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 [33]. With geographic distance, vital informal communication becomes less frequent [33] 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 [34].

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 con-

cepts like awareness [35], or by a targeted combination of various communication channels [8]. 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 [26, 27].

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.

4. Research Method

To address articulation work as informal and situated practice [36], 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 [37]. Our research is strongly oriented on ethnographic methods, which offer many advantages for the analysis of differentiated relations in complex environments [38, 39].

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.

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.

4.3 Grounded Theory Analysis

For the analysis of the collected data, we drew on Glaser's and Strauss' Grounded Theory [40]. 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 [36].

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.

4.4. The Cases

4.4.1 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."¹

4.4.2 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. 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

¹ All citations from the interview and field note transcripts that we present in this paper were translated into English by the authors.

solution and thus requires much less communication with customers.

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.

5.2 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.

5.3 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 ob-

served 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 Same-time 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.

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 [23]. 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 [20]. 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 [36]. 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 be-

cause, 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 [34] 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 [33] 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.

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 collaboratively 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.

8. References

- [1] OECD, "Information and Communication Technologies. OECD Information Technology Outlook: 2004 Edition", 2004. <http://www.oecd.org/dataoecd/19/27/37741043.pdf>, 15.02.07.
- [2] Working Party of the Information Economy, "Potential Offshoring of ITC-intensive using Occupations", 2005. <http://www.oecd.org/dataoecd/35/11/34682317.pdf>, 15.02.2007.
- [3] P. J. Ågerfalk and B. Fitzgerald, "Flexible and distributed software processes: old petunians in new bowls?" *Communications of the ACM*, vol. 49, pp. 27-34, 2006.
- [4] D. D. Damian and M. Deependra, "Global Software Development: How Far Have We Come?" *IEEE Software*, vol. 23, pp. 17-19, 2006.
- [5] R. D. Battin, R. Crocker, J. Kreidler, and K. Subramanian, "Leveraging Resources in Global Software Development," *IEEE Software*, vol. 18, pp. 70-77, 2001.
- [6] A. Mockus and D. M. Weiss, "Globalization by Chunking: Quantitative Approach," *IEEE Software*, vol. 18, pp. 30-37, 2001.
- [7] J. Dibbern and A. Heinzl, "Selective Outsourcing of Information Systems in Small and Medium Sized Enterprises," in *Information Systems Outsourcing. Enduring Themes, New Perspectives and Global Challenges*, R. A. Hirschheim, A. Heinzl, and J. Dibbern, Eds., 2. ed. Berlin, Heidelberg: Springer, 2006, pp. 57-81.
- [8] D. D. Damian, F. Lanubile, and T. Mallardo, "The role of asynchronous discussions in increasing the effectiveness of remote synchronous requirements negotiations," presented at International Conference on Software Engineering, Shanghai, 2006.
- [9] P. Hinds and C. McGrath, "Structures that work: social structure, work structure and coordination ease in geographically distributed teams," presented at 20th Conference on Computer Supported Cooperative Work, Banff, 2006.
- [10] J. Dibbern, *The Sourcing of Application Software Services. Empirical Evidence of Cultural, Industry and Functional Differences*. Univ. Diss. Bayreuth 2003, Heidelberg: Physica, 2004.
- [11] O. E. Williamson and S. E. Masten, *The economics of transaction costs*. Cheltenham, UK; Northampton, Mass. USA: E. Elgar Pub., 1999.
- [12] P. Keil, "Principal Agent Theory and its Application to Analyze Outsourcing of Software Development," presented at International Workshop on Economics-Driven Software Engineering Research, St. Louis, 2005.
- [13] B. Cumps, S. Viaene, G. Dedene, and J. Vandenbulcke, "A Theoretical Exploration of the Relationship between Outsourcing and Business/ICT Alignment," presented at

- European Conference on Information Systems, Göteborg, 2006.
- [14] R. P. Rumelt, *Strategy, structure, and economic performance*, Rev. ed. Boston, Mass.: Harvard Business School Press, 1986.
- [15] S. Narayanan, S. Mazumder, and R. R., "Success of Off-shore Relationships: Engineering team structures," presented at International Conference on Global Software Engineering, Costão do Santinho, Brazil, 2006.
- [16] J. Stark, M. Arlt, and D. H. T. Walker, "Outsourcing Decisions & Models - Some Practical Considerations for Large Organizations," presented at International Conference on Global Software Engineering, Costão do Santinho, Brazil, 2006.
- [17] B. Nett, M. Durissini, and M. Klann, "Wissensprozesse in kleinen Unternehmen der Softwarebranche aus der Sicht von Entwicklern", 2004. <http://www.softwarekompetenz.de/?21522>, 15.02.07.
- [18] B. Nicholson and S. Sahay, "Some political and cultural issues in the globalization of software development: case experience from Britain and India," *Information and Organization*, vol. 11, pp. 25-43, 2001.
- [19] E. Ferneley and F. Bell, "Tinker, Tailor: Information Systems and Strategic Development in Knowledge-Based SMEs," presented at European Conference on Information Systems, 2005.
- [20] B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can distributed software development be agile?" *Communications of the ACM*, vol. 49, pp. 41-46, 2006.
- [21] O.-K. Lee, P. Banerjee, K. H. Lim, K. Kumar, J. Van Hillegerberg, and K. K. Wei, "Aligning IT components to achieve agility in globally distributed system development," *Communications of the ACM*, vol. 49, pp. 49-54, 2006.
- [22] M. Paasivaara and C. Lassenius, "Could Global Software Development Benefit from Agile Methods?" presented at International Conference on Global Software Engineering, Costão do Santinho, Brazil, 2006.
- [23] G. Lee, W. Delone, and J. A. Espinosa, "Ambidextrous coping strategies in globally distributed software development projects," *Communications of the ACM*, vol. 49, pp. 35-40, 2006.
- [24] A. L. Strauss, *Continual permutations of action*. New York: Aldine de Gruyter, 1993.
- [25] K. Schmidt and L. Bannon, "Taking CSCW Seriously: Supporting Articulation Work," *Computer Supported Cooperative Work (CSCW): An international Journal*, vol. 1, pp. 7-40, 1992.
- [26] R. E. Grinter, "Supporting Articulation Work Using Software Configuration Management Systems," *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 5, pp. 447-465, 1996.
- [27] A. Crabtree, J. O'Neill, P. Tolmie, S. Castellani, T. Colombino, and A. Grasso, "The Practical Indispensability of Articulation Work to Immediate and Remote Help-giving," presented at Conference on Computer Supported Cooperative Work, Banff, Alberta, 2006.
- [28] A. L. Strauss, "Work and Division of Labor," *The Sociological Quarterly*, vol. 26, pp. 1-19, 1985.
- [29] K. Schmidt and C. Simone, "Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design," *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, vol. 5, pp. 155-200, 1996.
- [30] A. Fjuk, O. Smørdal, and M. I. Nurminen, "Taking Articulation Work Seriously - an activity theoretical approach", 1997. <http://heim.ifi.uio.no/~ftp/publications/others/OSmordal-5.pdf>, 15.02.07.
- [31] S. L. Star and A. L. Strauss, "Layers of Silence, Areas of Voice: The Ecology of Visible and Invisible Work," *Computer Supported Cooperative Work*, vol. 8, pp. 9-30, 1999.
- [32] E. M. Gerson and S. L. Star, "Analyzing due process in the workplace," *ACM Transactions on Office Information Systems*, vol. 4, pp. 257-270, 1986.
- [33] J. D. Herbsleb, D. J. Paulish, and M. Bass, "Global Software Development at Siemens: Experience from Nine Projects," presented at International Conference on Software Engineering, St. Louis, 2005.
- [34] J. D. Herbsleb, T. A. Finholt, and R. E. Grinter, "An Empirical Study of Global Software Development: Distance and Speed," presented at International Conference on Software Engineering, Toronto, 2001.
- [35] H. Spanjers, M. ter Huurne, B. Graaf, M. Lormans, D. Bendas, and R. van Solingen, "Tool Support for Distributed Software Engineering," presented at International Conference on Global Software Engineering, Costão do Santinho, Brazil, 2006.
- [36] L. A. Suchman, *Plans and situated action*. Cambridge, New York, Port Chester, Melbourne, Sidney, 1987.
- [37] C. Argyris, R. Putnam, and D. M. Smith, *Action science*, 1st ed. San Francisco: Jossey-Bass, 1985.
- [38] W. Tichy and A. Höfer, "Status of Empirical Research in Software Engineering", 2006. <http://www.ipd.uka.de/Tichy/uploads/publikationen/131/StatusEmpiricalResearch2006.pdf>, 15.02.2007.
- [39] L. A. Suchman, "Making Work Visible," *Communications of the ACM*, vol. 38, pp. 56-64, 1995.
- [40] A. L. Strauss and J. M. Corbin, *Basics of qualitative research: techniques and procedures for developing grounded theory*, 2 ed. Thousand Oaks: Sage Publications, 1998.