To appear in: Boden A, Nett B, Wulf V. Operational and Strategic Learning in Global Software Development - Implications from two Offshoring Case Studies in Small Enterprises. *IEEE Software*. Accepted for publication.

Operational and Strategic Learning in Global Software Development

Implications from two Offshoring Case Studies in Small Enterprises

Submitted to IEEE Software

Alexander Boden

Institute for Information Systems, University of Siegen
alexander.boden@uni-siegen.de

Bernhard Nett

Institute for Information Systems, University of Siegen
alexander.boden@uni-siegen.de

Volker Wulf

Institute for Information Systems, University of Siegen
volker.wulf@uni-siegen.de

Abstract

Small to medium-sized software enterprises (SME) increasingly participate in offshoring activities. Detecting market niches and deploying highly flexible software development approaches are seen as key competitive abilities of SME. Therefore, it is of major importance to learn how offshoring affects these capabilities which are closely related to organizational learning. We present case studies from two German companies that engage in offshoring of software development. By comparing the cases with each other, we highlight the different structures the companies chose for their development work and how these structures were enacted in practice. Furthermore, we show how related practices affect strategic and operational aspects of Argyris et al.'s (1985) conception of single- and double-loop learning. Our case studies show that organizational learning is a problem for SME engaged in offshoring and that an inability for double-loop learning can even lead to failures in case of organizational restructuring.

Introduction

With increasing globalization, distributed software teams have become fairly common. Usually, companies which offshore their software development expect a reduction of costs and access to new markets. On the other hand, distributed teams are faced with problems related to the spatial, temporal and cultural barriers of globally distributed work.

For a long time, studies have mainly treated offshoring as a make-or-buy decision of large companies, offering recommendations and discussing best practices. While today offshoring is increasingly understood as a dynamic process, there are still few studies which focus on related long-term implications of offshoring on key organizational capabilities, especially with regard to small and medium-sized enterprises (SME) which make 99.8% of the German software industry (cf. German Federal Statistical Office, 2003). SME count their abilities to offer highly customized software solutions and to adapt quickly to changing customer demands amongst their most important competitive capabilities [1]. Hence, when SME decide to engage in offshoring, it has to be organized in a way that allows flexible adaptations of the organization to changing demands—or, in other words, ongoing operational and strategic learning.

We want to 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. After discussing related work, we describe our research methods. The subsequent sections present the two case studies as well as the different models of work organization. We conclude the paper with a discussion of the identified offshoring practices and learning strategies. The related "lessons learned" are meant to provide interested practitioners with ideas about what to expect in practice.

Related Work

Recent studies have increasingly focused on operational aspects of distributed cooperation [2], but there are still few studies on long-term consequences on key organizational capabilities [3], not to mention the learning necessary to secure them, for instance, in SME.

In software engineering, learning has been discussed as a major issue for software development in the context of knowledge management. Its early years were inspired by the belief that technology-intensive systems could accumulate and automatically provide knowledge "on demand". Learning was reduced to input-output-processing, and knowledge conceptualized merely in its explicit form [4].

However, although knowledge may be represented in the form of explicit content, such content can become knowledge only when being contextualized. Furthermore, knowledge needs to be framed in order to contribute to the expertise needed in practice [5]. Learning, therefore, cannot be reduced to data storing in the brain, but requires understanding in much broader sense, for example, opportunities to develop practical competences and expertise [6]. Hence, learners should not be considered as mere consumers, but as decisive actors, who develop cooperative activities of their own [7]; a transition which has been labeled as the "second wave of knowledge management" [8].

The related paradigm of self-organization was thus elaborated as a means and end in computer-mediated education. It has also been discussed in relation to organizational development and distributed teams. Orlikowski [9] has hinted at knowing-in-practice as an important element for organizational operation. By illustrating how knowledge was enacted and (re-)constituted by several practices of a distributed organization (such as sharing identity, interacting face-to-face, aligning efforts, learning by doing and supporting participation), she argues that instead of hypothetically constructing formal, de-contextualized "best practice" models, "useful practices" should empirically be identified in practice.

In our paper, we will complement Orlikowski's argument by addressing the question how organizational learning is enacted in practice by SME of the German software industry. In doing so, we refer to the framework of Argyris et al. [10] which states that learning can be identified when one compares the consequences of actions with the expectations that guided their planning. In their view, learning entails two different layers which need to be covered: single-loop learning refers to the operational level ("Are we doing things right?"), while double-loop learning comprises learning on the strategic level ("Are we doing the right things?"). This means that learning of enterprises can be derived from decisions dedicated to operational or strategic aspects of the cooperation.

In the case of offshore partners, reflecting on operational and strategic decisions can be particularly challenging. While learning in local teams often takes place implicitly, distributed actors may have to adopt explicit strategies to organize their knowledge exchange. This is especially important for SME needing to ensure their flexibility. As SME are often highly dependent on agile development methods, exhaustive communication and flexible interaction, implementing organizational learning can be difficult for them in the context of offshoring [11].

In this context, Hinds and McGrath [12] have highlighted the role of emerging informal hierarchies for smooth coordination of distributed teams. However, as offshoring projects do not per se lead to efficient informal hierarchies and smoothly coordinated cooperation, the question remains what offshore partners can actively do in order to secure the agility of their software development, and if there is any opportunity to use organizational learning for this purpose. As a related conceptual framework, one can use the prescriptive model of Argyris et al. as a descriptive one, trying to identify opportunities for related "useful practices" in the sense of Orlikowski.

To do so, we adopted Anselm Strauss' conception of articulation work [13] (see sidebar 1). Articulation work may contribute to learning on the operational level, as it underpins formal divisions of labor by informal, flexible adjustments. However, it can also contribute to learning on the strategic level, when collaborative actors reflect upon their articulation work, relating to shared experiences and discussing possible solutions [14].

Hence, by analyzing the role of articulation work within offshore software development of SME, we want to understand the impact of offshoring on the agility of these companies. We try to understand the opportunities offshoring has for the different types of learning and for organizational learning, which does not only require learning, but also the possibility of institutionalizing its results.

Sidebar 1: Articulation Work

The concept of articulation work was introduced by the sociologist Anselm Strauss for the analysis of interdependent actions of cooperating actors.

Articulation work is needed to regulate the division of labor: who does what, when, where, how, with which quality, etc. Yet, articulation work is a broader, more holistic concept than coordination: while the latter only governs the planned distribution of labor (in the sense of distributing responsibilities), articulation work also manages unexpected preconditions which emerge due to 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 much broader understanding of cooperative work in complex environments like offshoring projects.

Further Reading:

Schmidt, K., and Bannon, L. Taking CSCW Seriously: Supporting Articulation Work. Computer Supported Cooperative Work (CSCW): An international Journal, 1(1) (1992), 7-40.

Strauss, A. L. The Articulation of Project Work: An Organizational Process. The Sociological Quarterly, 29(2) (1988), 163-178.

Research Methods

We have been conducting our study in several phases since 2006. After an initial literature study we conducted semi-structured interviews with thirteen managers and developers of German SME, two interviews with representatives of an IT industry association and a large German company as well as four interviews with Eastern-European offshoring vendors. The preliminary results of the interviews were used to identify challenges of offshoring as well as strategies used by German SME to deal with them. From our sample, two companies were chosen for further analysis: Alpha and Beta.

For the next phase of the data collection we drew on a triangulation of ethnographic research methods, comprising interviews, on-site observation and artifact analysis. The on-site observation was conducted by visiting each of the German SME for a period of twelve working days. In addition, we visited the Russian partner company Alpha for one week. In order 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 are continuing our study in form of an action research approach in company Alpha.

The analysis of the data was based on Glaser's and Strauss' Grounded Theory [15]. After each step, the transcripts of the material were scrutinized and coded. At first, we composed categories (such as "knowledge exchange", "informal coordination", or "formal workflow") based on the findings in the collected data. Then, these categories were related to each other and evolved during the further research.

The Case Studies

We chose two companies, which expressed very different perceptions of the importance of formalization for successful offshore software development. Both companies had several years of experience with offshore development in Russia.

Company 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. The product line comprises databases, documentation and presentation systems used by archives or museums.

Since the late 1990s, the company has been employing four software developers in Tomsk, Siberia. The business relation started with an internship of a Russian developer who still works for the company. Due to the positive experiences with him, the German manager decided to expand the cooperation. The first project aimed at the reengineering of an existing standard software product. Despite unexpected delays in the development, the offshoring was expanded to several smaller customer-specific projects which involved a closer cooperation between German project leaders and offshore developers.

During the interview in the first phase of our study, the German owner underlined the reliance on flat hierarchies and flexible self-organized work. Formal work processes and the use of development models will be considered if the customer insists on them, but from the company's perspective this is not necessary. Instead, the company emphasizes informal and flexible work practices, allowing the project managers to run their projects with great levels of (semi-) autonomy. For the handling of specifications, the company relies on plain word documents which are sent to the developers via email, or in some cases on a defect-tracking system.

Company Beta

Company 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. Four offshore developers in Saint Petersburg carry out the software development under the supervision of a German project leader in Berlin. Another seven employees at the Berlin office provide testing and support.

According to the German manager the decision for offshoring mainly aimed at the reduction of development costs. Due to 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 couple of months. After their return, they took over the software development. Since then, the Russian team has grown quickly to an amount of fifteen developers, which required certain adjustments in the formal division of labor (see below).

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

Different Models of Work-Organization

The models (1-4) in table 1 provide an overview over the different formal models chosen by Alpha and Beta to organize their offshore software development. In this regard, models 1 and 2 describe patterns of cooperation implemented by Alpha. Models 3 and 4 represent the adaptation Beta had to conduct in order to deal with emerging necessities of their offshoring project.

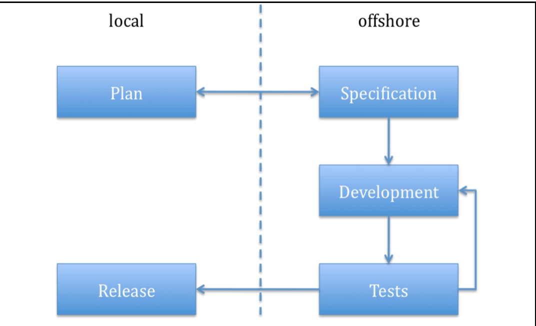
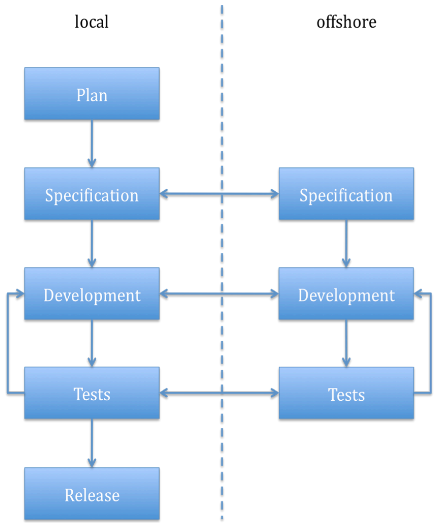
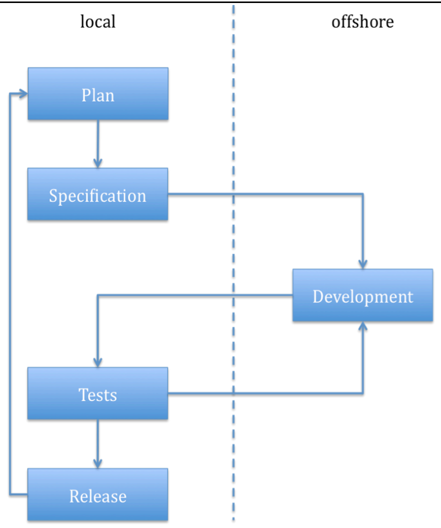
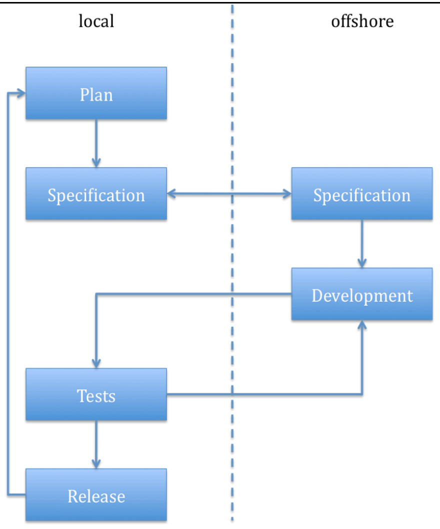
Alpha	 local offshore Plan Specification Development Release Tests **Model 1: Division of labor in case of Alpha's standard software solution**	 local offshore Plan Specification Specification Development Development Tests Tests Release **Model 2: Division of labor in case of Alpha's customer-specific projects**
Beta	 local offshore Plan Specification Development Tests Release **Model 3: Initial division of labor in case of Beta**	 local offshore Plan Specification Specification Development Tests Release **Model 4: Division of labor in case of Beta after the reorganization**

Table 1: Offshoring models

The models represent different divisions of labor, resulting workflows, and inter-relationships between the cooperating teams. This involves the exchange of artifacts such as plans, specification documents, bug-descriptions and source code/prototypes. The arrows indicate the direction of the transfer of artifacts and are usually embedded in articulation work, as discussed below. Two-pointed arrows indicate exchange processes in close cooperation, while the dashed line represents the barrier(s) between the local and the remote teams.

In regard to our analysis, the models can be used as hints to necessities of learning in practice. On the one hand, these practices are related to the different tasks the teams have to accomplish during their development work, for example in regard to 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 challenges of learning. Hence, arrows crossing the dashed line will be the focus of our analysis.

Model 1: Division of Labor in case of Alpha's standard software solution

Alpha's offshoring started as a reengineering of an outdated legacy product. The corresponding model 1 is rather simple, entailing inter-site connections mainly concerning the project plan being transferred to the Russian team which in turn was to deliver the reengineered product back to Germany (see table 2 for examples).

In regard to 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 since the existing product was only to be re-engineered. This could easily be fully done offshore, mostly avoiding inter-site cooperation. This also included direct communication between the customers of the German company and the Russian developers in cases of bug-reports or feature requests. As the Russians had ample opportunities for self-organization—as long as they kept up with deadlines and requirements—they were able to use their experiences for operational decisions, for example, concerning the choice of development tools, the documentation of the development as well as the distribution of tasks.

While there was related space for single-loop learning, the strategic project planning and the formal coordination of the company (this is: double-loop learning) relied on regular personal visits of the German manager at the offshore site in Tomsk, as well as on visits of the Russian team to Germany for strategic workshops (which technology to choose for the next version, rough project roadmap etc.).

Model 2: Division of labor in case of Alpha's customer-specific projects

Due to positive experiences, the cooperation 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 2 contains many inter-relationships 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 initial articulation of the project work was mainly done 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 both in operational as well as strategic terms. It was apparent that the German team members valued the technical knowledge of the Russian team and involved them in product finding, too.

Due to the direct contact between the Russian teams and the customers (in model 1), the Russian team learned about the German user domains. This eased the handling of the complex model 2 projects, 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 Russians to influence the trajectories of the project work and to bring in their own ideas.

Summary (Alpha)

Looking at the different models of work organization in Alpha, it gets apparent that both model 1 as model 2 attempt to keep the core of software production integrated, and thus not to separate the responsibilities according to phases such as specification, development and testing. On the contrary, these three elements are held together thus allowing for agile, iterative proceeding.

Alpha's strategy of offshoring thus rather was aiming at a replication of their own organizational structure (adhocracy) at the offshore site. That means, in-house activities of everyday software development, such as the specification of features, the development of code, and testing are performed in a close cooperation between the sites.

Ongoing articulation work played a pivotal role for the accomplishment of everyday work, as the division of tasks was always negotiated ad-hoc between the teams. Potential benefits of specialization, on the other hand, were only exploited at a very low rate. Learning was dependent upon considerable articulation work, which mainly remained focused upon limited, situated problems at single-loop and double-loop learning, as well. Therefore, it could not contribute to organizational learning, as it did not even consider structural changes of the given adhocracy.

Model 3: Initial division of labor in case of Beta

The development of the standard software solution of company Beta followed a fixed release cycle of six month. Initially, the formal offshoring model 3 was introduced, which aimed at the offshoring of the development work to Russia, while all other tasks (such as the definition of new features and the description and classification of bugs) were to remain in Germany. Hence, interdependency between the teams merely concerned 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.

In regard to coordination and control, the company tried to apply a much more single-sided approach. The key for this practice was sought to be the preparation of exhaustive specification documents for the Russians, which included ample information concerning the inter-relations with other modules of the software, the design of the interface, expected behavior, and so on. The Russians in turn were to document their progresses in terms of monthly reports and to review their code on a regular basis for quality assurance. In addition, the German project leader was to visit the offshore team on a regular basis, usually shortly before new releases. During these visits, the German project leader mainly helped handling the bugs (usually discovered in the last minute) and—if time allowed—discussed the features of the following release with the Russians. Strategic questions (related to double-loop learning) where mainly discussed in Germany.

From the perspective of the German team, the main challenge turned out to be the writing of 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 leader "one day of development required one day of writing specifications". As it had become increasingly difficult to specify new features quickly enough to keep the growing offshore team busy, the decision was taken by the German management to change the formal division of labor (see model 4).

Model 4: Division of labor in case of Beta after the reorganization

After the restructuring, the inter-connections between the teams became much more complex than initially intended. Since 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 the necessary context knowledge and thus more articulation work between the teams.

In practice, the articulation work turned out to be difficult: since the Russians lacked most of the necessary context information about the practical usage (e.g. the demands of the customer) and the technical background of the product (e.g. interdependencies with certain modules) they found it hard to write proper specifications. This led to frustration on both sides: the Germans were discontent with the quality of the Russian specification documents and had to assist and correct the work of the other team, requiring much time for articulation work; the Russians, on the other hand, felt overstrained and fulfilled their new tasks only reluctantly.

In an attempt to improve the specifications and reduce the need for ample articulation work, the company introduced an even higher level of standardization. By providing standardized examples and checklists, which were meant to help the Russian developers with their tasks, the company expected to reduce articulation work (visible in the amount of communication) and to ensure the quality of the produced documentation. However, since the underlying problem of lacking knowledge could not 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) wherever possible. The ongoing problems contributed to the decision to terminate the cooperation in 2008 (although it has to be stressed that the termination was decided for several reasons, including fast rising wages in Saint Petersburg).

Summary (Beta)

Looking at Beta's models of work organization, it became apparent that this company followed a fundamentally different approach in comparison to Alpha. Instead of 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 was to concentrate on planning and controlling activities, while the actual development work was to be performed exclusively by the offshore site with its lower wages.

As the modus 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, an even higher) level of formalization to reduce the demands of articulation work. In contrast to Alpha, they did neither accept more frequent meetings nor more intense communication between the sites.

That means that company B did not give up its demand to exploit the benefits of specialization. It tried instead to re-organize specialization in a formal, top-down manner. When the amount of the necessary articulation work exceeded its ambitious expectations, the management reacted with structural changes of its specialization model. As innovative product finding was not 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, it was only the manager whose learning counted, while for example feedback from the Russian team was not considered.

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 [9]. For example, intense face-to-face contacts, broad participation in meetings as well as the importance of aligning efforts were important factors for the two teams (see table 2). However, there were also differences which were related to the attempts of the companies to deal with these practices, and also to the different types of products the companies were developing.

While enterprise A accepted the related articulation work of its small customer oriented projects by intensifying the personal visits between the sites through organizing personal visits and workshops, B tried to reduce it by increasing formalization of specialization in the development of its standardized product. The strategy of company B turned out to be problematic, as it did not account for the necessary mutual learning about important context information and domain knowledge, or the necessary contact to the customer.

Model	Operational decisions	Strategic decisions	Example of Articulation Work between the sites
1	Taken by the offshore developers (e.g. task assignment, development tools, etc.).	Taken by the Russian team manager in cooperation with the German manager (e.g. system framework, deadlines, etc.).	Russian developers visit Germany to discuss strategic questions (which technology to choose, rough project roadmap etc.) with German developers under supervision of the German manager. Offshore team mainly operates on its own afterwards.
2	Negotiated between the German project leaders and the offshore developers (e.g. task assignment, development tools, etc.).	Taken by the German project leaders, with consultancy of the offshore developers (e.g. system framework, deadlines, etc.).	German project manager visits the team in Tomsk to explain to them his vision of a new project. Requirements and project plan are specified cooperatively during several meetings. Ongoing chats or, if necessary, even prolonged visits for coordinating the later development.
3	Taken by the Russian team manager (e.g. task assignment), and by the German developers (e.g. bug assignment).	Taken by the German project manager (e.g. deadlines, specifications, etc.)	German project manager writes specifications. The Russian team manager assigns them as tasks to the Russian developers, results are tested by the German team. Regular visits of the German project manager before finishing new versions.
4	Taken by the Russian team manager (e.g. task assignment), and by the German developers (e.g. 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 the Russian developers during his personal visits. Offshore developers have to write the specifications, which are in turn checked by the German project manager. As Russian developers are lacking knowledge to write proper specifications, this requires much supervision. Forms are introduced to help the Russians writing complete specifications.

Table 2: Different kinds of learning and articulation work

In 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 also some strategic questions (“Can we reuse something we already have?”). In general, the focus was on operational aspects of the cooperation, but basic questions such as the organization of work itself were barely covered, if covered at all. As the company did not 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, double-loop learning remained limited, as it did not allow for restructuring, a major domain for organizational learning [10]. Beta, on the other hand, did engage in restructuring, but this was reduced to a single-sided top-down decision by the German management, and it did neither include the expertise of the Russian side nor that the expertise of the developers.

The inability of both companies to implement structural changes may be related to the particular work practices of small software enterprises [1]. The practices of articulation work as well as the models of cooperation we observed seem to be highly specific for small software companies. SME often work in very flexible ways and usually cannot afford to engage in too much specialization, or institutionalized self-reflection [14]—with the possible consequences we found in our study.

Conclusion

According to Argyris et al.'s framework, 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 have in order to develop their organizational structure in case of offshoring. By comparing the cases it gets also apparent that it can be worse to restructure the offshore relation in an inappropriate way (as in case of Beta) than to stick 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 company Alpha, which did not even try to change it's structure. From our perspective, this is an interesting finding as offshoring strategies are often discussed in relation to the benefits of a restructuring of the organization.

Our cases show that the related organizational learning can be problematic for offshore cooperation. Decisions (as in company A) can be based upon 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, decisions are taken without taking into account the full offshore expertise, there is a high risk of failure. In both cases, offshoring can endanger the agility of companies. This shows that further research on opportunities for organizational learning, which fit the demands of SME, remains an important task.

References

[1] Friedewald, M.; Rombach, H. D.; Stahl, P. et al.: Softwareentwicklung in Deutschland: Eine Bestandsaufnahme. In: Informatik Spektrum 24, vol. 2 (2001), 81-90.

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

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

[4] Bjørnson, F.O. and Dingsøyr, T.: Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used.

[5] Suchman, L.A.: Plans and situated actions: the problem of human-machine communication, Cambridge University Press, 1987.

[6] Polanyi, M.: The tacit dimension, Routledge & K. Paul, 1967.

[7] De Paula, R., Fischer, G., Ostwald, J., 1999. Courses as Seeds: expectations and Realities, Proceedings of CSCL 99

[8] Huysman, M. H., Wulf, V. (Ed.): Social Capital and Information Technology, Cambridge MA, 2004.

[9] Orlikowski, W.: Knowing in Practice: Enacting a Collective Capability in Distributed Organizing, Organization Science 13 (2002), 249-273.

[10] Argyris, C.; Putnam, R. and Smith, D. M. Action science. Jossey-Bass, 1985.

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

[12] Hinds, P. and McGrath, C.: Structures that work: social structure, work structure and coordination ease in geographically distributed teams, Proc. Conference on Computer Supported Cooperative Work (2006), 343-352.

[13] Strauss, A. L. Social organization of medical work. University of Chicago Press, 1985.

[14] Boden, A., Nett, B., and Wulf, V. Coordination Practices in Distributed Software Development of Small Enterprises. Proc. International Conference on Global Software Engineering (2007), 235-246.

[15] Strauss, A. L. and Corbin, J. M. Basics of qualitative research: techniques and procedures for developing grounded theory. Sage Publications, 1998.

Author Biographies

Alexander Boden is a research associate at the Institute for Information Systems and New Media, University of Siegen. His research areas focus on Global Software Engineering, Ethnographic Methods and Computer-supported Cooperative Work. He studied Cultural Anthropology in Bonn and is working on his Ph.D. thesis.

Bernhard Nett is holding a research position at the University of Siegen. His research foci are Sociology, Media Appropriation, Business Ethnography, Action Research, Computer-Supported cooperative work and Requirements Engineering. He is currently accomplishing a postdoctoral lecture qualification process.

Volker Wulf is a professor in Information Systems and Director of the Media Research Institute at the University of Siegen. At Fraunhofer FIT, he heads the research group User-centred Software-Engineering (USE). His research interests lie primarily in the area of Computer-Supported Cooperative Work, Knowledge Management, Human Computer Interaction, and Participatory Design.

Information Systems and New Media
University of Siegen
Hoelderlinstr. 3
D-57076 Siegen, Germany

www.artos.uni-siegen.de