CONSTRUCTING COMMON INFORMATION SPACES IN GLOBAL SOFTWARE DEVELOPMENT

Bannon, Liam J., Interaction Design Centre and Lero-The Irish Software Engineering Research Center, University of Limerick, Ireland, liam.bannon@ul.ie

Avram, Gabriela., Interaction Design Centre and Lero-The Irish Software Engineering Research Center, University of Limerick, Ireland, gabriela.avram@ul.ie

Abstract

This paper examines some of the issues involved in the coordination of geographically-distributed software development teams. In recent years, the phenomenon of global software development (GSD) has become widespread, for a variety of reasons such as "chasing the sun", purportedly lower costs in certain countries, labour availability, market accessibility, etc. It is often assumed that the problems of time and distance separating team members can be handled through modern computing and telecommunications hardware and software. However, many voices are now being raised pointing to on-going difficulties in maintaining shared awareness of project status, lack of contextual information, and difficulties in understanding across different cultures. We have recently begun a project investigating the social, organizational and cultural aspects of global software development, focusing on the work practices of the team members, in order to document and analyse concrete instances of communication and coordination difficulties. Our perspective on this area is informed by earlier work in the Computer Supported Cooperative Work (CSCW) research field, especially concerning the work that people must do to make sense of each other, whether in face-to-face, or computer-mediated interaction. In this paper we discuss one concept, that of Common Information Spaces (CISs), and show how we believe that this concept might be useful in helping to analyse some of the ways in which distributed teams achieve some level of mutual understanding. We briefly illustrate our account with reference to some early fieldwork material we have gathered.

Keywords: common information spaces, bug-tracking, translation, boundary objects

1 INTRODUCTION

Many companies in the ICT sector now have a global reach. This move towards globalization has been prompted, not simply to ensure a presence close to different customer bases, but also for economic reasons. Nowadays it is quite common to find software development projects that are distributed across the globe, with team members from North America, Europe and the Far East. Another phenomenon that is occurring is the outsourcing of significant software projects from EU and US corporations to independent Asian outfits, who offer turnkey solutions. The recent ACM Job Migration Task Force report on globalisation and software offshoring (Aspray et al. 2006) reports that globalisation of the software industry will increase in the coming years, fuelled by both information technology itself, by government actions and by economic factors. The reasons for offshoring stay the same: cost reduction, access to a larger labor pool, access to special expertise, good quality of work, closeness to particular markets. But at the same time, GSD poses specific challenges related to communication, coordination and collaboration that can impact on the supposed benefits of distributed work (Carmel and Agarwal 2001). One of the largest sources of problems in Global Software Development (GSD) is related to communication across sites, with poor communication hindering both coordination and control processes (Herbsleb and Mockus 2003). The belief that communication and collaboration among software teams spread across the globe can be ensured through the use of standard platforms, software tools, and procedures, augmented by occasional video-conferences, and emails, has come to be seen as naïve. Successful leaders of distributed teams argue for the crucial importance of additional "soft" factors, concerning such issues as respect, privacy, trust, empathy and care, which are essential to ensure successful international collaborations. The importance of dealing with cultural differences and putting in place effective knowledge sharing mechanisms has been noted by several authors (Herbsleb and Moitra 2001, Prikladnicki 2003). Several workshops dedicated to GSD research and practice have emphasized the need for tools and techniques that not only improve development processes, but also address organizational and social issues in global software development (Lanubile et al., 2003).

There are a variety of research programmes that are relevant to understanding these issues of distributed software development and global software work more generally. Within the software engineering domain, there are a number of individuals and research groups that have been studying aspects of the problem. The early work of Herbsleb, Grinter and colleagues at Bell Labs is an early case in point(Herbsleb and Grinter 1999, Herbsleb and Moitra 2001). Some of this work attempts to develop models of collaboration and to perform quantitative studies of performance in distributed teams. Erran Carmel has been one of the early figures to write and conduct interviews in this area, and his book on Global Software Teams (1999) is still a useful base. Within the computing field more generally, there have been some notable early empirical projects investigating issues of collaboration and communication in development teams. The area of CSCW has produced a number of interesting studies on articulation work and coordination theory (Schmidt and Bannon 1992, Malone and Crowston 1994) in general as well as the exploration of various kinds of media spaces to enhance collaboration (Xerox PARC, Bell Labs, MIT Media Lab). While much of this work is not exclusively focused on the software engineering domain, there are many useful lessons to be learnt from the detailed studies of coordination in the workplace, and the use of various media for collaboration support. Other literature relevant to the problem exists in the Information Systems field, for example Walsham (2001), and Sahay, Nicholson & Krishna (2003). These researchers often assume an interpretivist stance to their research object, and are influenced by conceptual frameworks such as actor-network theory and socio-technical systems theory.

2 THE *SOC*GSD PROJECT

At the University of Limerick, Ireland, a group at the Interaction Design Centre has received national funding as part of a software engineering research consortium to study the social, organisational, and cultural aspects of global software development (socGSD). This project aims to explore through case studies, how organizations attempt to manage the coordination of engineering work via a variety of mechanisms, from the formation of closely-knit, though distributed, teams through to the use of outsourcing, which still requires project management skills in order to ensure the quality of the output, and the integration of the output into the overall product. In earlier work, we have studied issues in articulation and coordination work, information sharing and knowledge management practices in distributed work, and the role of organizational memory support tools. The socGSD project is investigating the nature of these issues through a variety of analytical and empirical methods highlighting both theory and practice in this domain.

The work is exploring the diversity of ways in which distributed teams shape their work practices, and come to a joint understanding of their objectives. The analysis of these cases is conducted utilising a variety of conceptual "lenses". A major orientation in the project is to the ways in which members through their practices, formal and informal, manage to cope with the organizational rules and formalizations they are required to use, even when these, at times create problems in accomplishing the work. Earlier CSCW work has shown the artful ways in which people mesh and interleave their work practices with and through technologies in order to accomplish their objectives. The CSCW work on interdependencies in work, the construction of common objects, “boundary objects”, and translation and interpretation of concepts among and between different communities of practice are being investigated. Methodologically, concepts such as virtual teams are being analysed, to understand whether, in practice, such concepts are useful analytical tools. The empirical studies are grounded in practice, and thus involve not simply interviewing, but workplace observations.

Research Approach

Over the past several months, we have been engaged in field work in several sites in Ireland where software development is being conducted involving geographically-distributed teams. Our research methods mainly rely on an interpretive, naturalistic approach to data collection and analysis. This means that we study the phenomenon in the actual settings where the work activity takes place, attempting to make sense of the work through the eyes of those actually doing it. There are a variety of methods which are being employed in this kind of qualitative research – interviews, both individual and group, introspective reports and diary and story analysis, observational studies, including shadowing of people during the course of their activities, analysis of documents, and workshops. In cases where video is allowed, we would hope to also perform occasional critical incident analyses, playing back the event and asking for interpretations from individuals involved in the incident, in order to build up a rich picture of daily work, and the ways in which people organize and make sense of their work lives. We also aim to perform a limited number of studies of a more quantitative nature within the organization, in the sense of examining the frequency of use of certain collaborative artefacts and media over a project time period. This data will be factored into our interviews with people about the utility and efficacy of various collaboration software tools in use. There is also a possibility of doing more detailed content analysis of communication over various media, although this raises a number of privacy concerns and ethical issues.

A major theme of our study is the ways in which various computer-based media and tools assist in, or perhaps disrupt, the ongoing accomplishment of group work across the different sites of development. Our approach will help to ground such general concepts as ”the virtual team“ and ”the death of distance“ in the lived working experiences and practices of people. We expect to be able to triangulate between different sources of information in order to build up a rich and meaningful picture of the ways in which distributed software work is actually accomplished in specific settings, and then analyse the commonalities and differences encountered in our different sites in order to elaborate our conceptual understanding of the domain. A detailed accounting of site procedures is not possible at this stage, but we attempt to provide some glosses on specific practices that we have observed.

3 COMMON INFORMATION SPACES AND COLLABORATION

Ever since the founding of the CSCW field, there have been debates as to the fundamental nature of cooperative work. e.g., whether all work is not, ultimately, cooperative, or whether the term denotes a specific form of work (Bannon & Schmidt, 1991). While we do not wish to re-visit this debate here, we wish to note that in any cooperative work situation, there is a need for some form of communication or information sharing between actors, implicit or explicit, in order to ascertain what features of the work are of note in that specific situation. Exactly what constitutes this "information space" is not agreed. For some, it simply refers to information, events, or objects that are tangible, external, "out there" in the world, that can be described extensively. For others, the "space" necessarily involves an interpretative component - the meaning of the terms or objects are not simply "given", but require an effort of interpretation on the part of the human actors who inhabit this space. It is this latter view which we emphasize here. Thus, to the extent that multiple actors can construct and maintain a common information space (CIS), they are able to articulate their work, and thus perform cooperative work. The concept of Common Information Space emerged from the CSCW community and was meant to designate both the material representations of information (artifacts), and the meaning attributed by the participants to these representations (Schmidt and Bannon 1992, Bannon & Bodker, 1997). The CIS literature argues that shared meanings are "achieved by specific actors on specific occasions of use", and that these meanings are "being debated and resolved, at least locally and temporarily" (Schmidt and Bannon 1992).

The problems encountered when different groups of people are involved in the production and maintenance of an information space extending over time and space have surfaced in a number of quite disparate studies concerning the relation between people and technology, many of which have not been conducted by people in the CSCW field. Within the field of social studies of science and technology, the problems of "alignment" of human and technical actors has been noted, and the way artefacts both shape and are shaped by the actor networks within which they participate (Callon, 1991). The work of Leigh Star and others on the concept of "boundary objects", and that of Latour and colleagues on the creation of "immutable mobiles", both can be viewed as being concerned with how communities develop means for sharing items in a common information space. For example, based on a study of a zoological museum, its creation, use and representations, Star & Griesemer (1989) introduce the concept of boundary objects characterising common intellectual tools, which play the role of containers and carriers:

...both plastic enough to adapt to local needs and constraints of the several parties employing them, yet robust enough to maintain a common identity across sites. They are weakly structured in common use, and become strongly structured in individual site-use. Like a blackboard, a boundary object 'sits in the middle' of a group of actors with divergent viewpoints (Star & Griesemer, 1989, p. 46).

The ordering and registration of the animals in the museum is one example of a boundary object, a map is another. They are both available to all users of the museum, though these people use the boundary objects in very different ways. As we shall see, discussions of boundary objects can be a vehicle for further studies of common information spaces across organizational boundaries.Understanding how people work together in networked communities is another area of investigation that has relevance for our discussions here. The concept of “community of practice” developed by Lave & Wenger (1991) to indicate the learning and working environment(s) in which most people work has important implications for the kinds of shared spaces that we might wish to develop for particular purposes. Whether we are moving information within or between communities of practice becomes of central concern. Robinson and Bannon (1991), within the CSCW field, explore some of the difficulties that can occur in sharing representations across different communities. While their paper discusses these issues in the context of systems development methodologies and practices, the problems noted pervade almost any distributed cooperative work setting where there is a requirement for the maintenance of some shared understanding of objects, events, information etc. within an information space peopled by actors from different communities of practice. Usually, CISs do not exist in isolation; they are overlapping and interconnecting, each individual participating in several CISs. In GSD environments, the transient and locally constructed character of the shared meanings attributed to an artifact is especially visible.

4 CONSTRUCTING COMMON INFORMATION SPACES IN GSD ENVIRONMENTS

In many software development organizations, teams and individuals working from different locations collaborate on releasing complex software applications or on providing software services. In our example, a small European team developed a software product that had to be integrated with another complex application still under development by a larger team in the US. The two teams are part of the same organization, but as the present organization was the result of several mergers and acquisitions, they had inherited different cultures and different development tools from their mother companies. The US team includes a very large number of developers working from different locations all over the US, while the small European team is collocated (with one exception). While the US team is distributed itself, it has the advantage of having started with a shared vision of the application to be developed, which, of course, has evolved over time. In what follows, we describe how various local software tools supported the development of locally shared information spaces among local developers, yet created some difficulty in the emergence of a common information space between the US and European teams.

The European team started the development of the new product approximately two years ago, using a configuration management system and a bug tracking system of their own choice. The company policy is to grant its teams as much freedom as possible in selecting the tools they want to use, so different parts of the organization and different teams use different tools. In time, the team developed a common understanding of what their product was meant to do, of the effort needed, the challenges involved and of the role of every team member in getting the work done. A common awareness of the different roles and tasks developed in time. A common information space emerged this way, enabling the team members to articulate their work and to dynamically assume specific tasks and responsibilities.

When the decision to integrate their application with the one developed by the US team was taken, a considerable number of problems (commonly known as “bugs”) had already been raised by the quality assurance team, and the developers were already working on solving them. At this point in time, the source code had to be checked in the other version control/configuration management system used by the American team. This “in-house” developed system also included the bug tracking functionality. While switching to the configuration management system used by the American team was mandatory for code integration, a decision had to be taken regarding the bug tracking system. The use of the American bug tracking system was unavoidable, because the two teams had to have a common language when referring to problems caused by the interlock of the two applications. But the European team decided to also continue using its own bug tracking system – providing a richer context and role-specific functionalities - in parallel, because the time left until the release date was only a few months. Their practices were heavily reliant on this bug tracking system, as it incorporated the sedimented experience of the team gained over several years, and giving it up would have meant a radical change in their practices. At the time, keeping the local system in play in parallel appeared the only reasonable solution, providing a kind of safety net for getting the internal work done, while adapting to a new tool and insuring good communication and coordination with their American colleagues.

The initial plan was to build a third software application that would have automatically updated each bug tracking system with the changes made in the other. This application was built, but it never worked properly, due to the huge number of alternative situations that had to be covered by it. Manual updating (or at least checking) proved to be the only viable solution for keeping the two bug tracking systems synchronized. This led to extra work and became annoying and time consuming.

Bugs could be viewed as boundary objects by the European team as they constructed their common information space, helping to coordinate work and build a shared understanding of the problems to be solved, of their priority, severity and of the possible solutions. Developers saw bugs as tasks assigned to them; testers saw them as outcomes of their work; software architects were monitoring them, in order to quickly detect areas with problems and come up with new solutions; the team manager was the one prioritizing them and managing the resources for solving them; at the higher management levels, bugs were seen as indicators of team performance. Their local bug tracking system allowed the addition of rich context information (history, images), allowing people to answer such questions as - where was the problem found, what kind of problem is it, who found it, who's responsible for solving it? This illustrates how the bug reporting system operated as a mechanism for articulating different practices (software architecture, development, testing, management). For different actors, bugs have different meanings at different moments in the project lifecycle. Each instantiation of a bug has specific characteristics (the "closed" aspect of a boundary object), and imposes a specific action (changes in the code, re-testing, confirming the elimination of the problem). But on the same time, as long as the problem continues to exist, the status of the bug is permanently re-negotiated (the "open", fluid aspect of a boundary object).

After code integration, complexity grew due to the new version control system with its embedded bug tracking system that had to be adopted. Each problem (bug) got two different codes in the two systems. One was used to coordinate work inside the team, the other to coordinate in the wider context and to negotiate a shared understanding of the problem and of the necessary actions for solving it.

The American team had its own culture (materialized in the tools they were using), shared a common goal, had developed their own language and a common understanding of roles, tasks and challenges. Over time, they evolved their own common information space. The bugs served as a coordination mechanism between the two teams, but because their different perspectives, the local meaning associated with a particular bug wasn't always the same.

In several situations, one team or the other had to make changes in its code in order to enable the two applications to function together. The negotiation always started at the developers' level, but if an agreement couldn't be reached, the problem had to be brought to the attention of team managers and sometimes escalated to upper levels. The two parties had different interests and different perspectives on the same problem, but a shared meaning had to be achieved in order to agree on a specific course of action. Thus we see the evolution of the common information spaces of a software development team during the duration of the project, expanding to include the use of new tools, to assimilate new people and to cover new assignments.

Despite all the problems created by the integration of the two applications, they were both released on time, and the teams are now working on the next version. For the new phase, the European team decided to give up its old bug tracking system and to rely only on the one embedded in the version control/configuration management system used jointly with the American team.

There are a number of lessons that were learned from this experience.

The situation showed that the simple use of collaborative tools is far from being enough for coordinating and articulating distributed work. Actors' interaction –at different levels, both inside and outside the software development team - is a vital element for getting the work done. In various occasions, assumptions about possible automation of operations prove to be false, and human actors are required to do extra work. This was the case with a proposed software application that would have synchronized the two bug tracking systems - a very time consuming manual interface was the only possible replacement.

The experience gained is expected now to provide a smooth passage from one tool to another; both developers and testers are now aware of the features missing in the new tool and are expected to find ways to cope with this situation. The old bug tracking system provided richer context information about bugs and the situations in which they occurred. But as long as their American counterparts can do their work relying on this limited information, the European team is confident that they could do the same.

5. CONCLUDING REMARKS

In terms of common information spaces, we note that the existence of separate software tools for bug tracking allowed for the emergence of distinct information spaces for the participants at different sites. We are noticing many cases where, through language and action, the team members at different sites display distinct perspectives on the problems as a result of their experience with the different systems. Different issues are highlighted and given salience, as the specific tools provide different views and affordances on the object code of interest. This hinders the creation of common information spaces amongst developers. Even sharing technologies does not alleviate the problem, as there still must be a shared set of activities around the use of the artefacts in order for shared understandings to emerge – a common information space. We put particular emphasis on the active construction and management of the CISs by the actors.

Acknowledgments

We wish to thank our informants in the observed companies for allowing us to interview and live with them on site for many days, to all the members of the socGSD Project in IDC. This research was supported by Science Foundation Ireland under PI grant 03/IN3/1408C.

References

Aspray, W., Mayadas, F. and Vardi M.Y. (ed.) (2006). Globalization and Offshoring of Software – A Report of the ACM Job Migration Task Force. Association for Computing Machinery. Available: http://www.acm.org/globalizationreport/ (2006-09-15).

Bannon, L. (2000). Understanding Common Information Spaces. Paper presented at the Workshop on Common Information Spaces, Copenhagen, Denmark. Available: http://www.itu.dk/~schmidt/ciscph2000/Bannon.pdf (2006-09-15).

Bannon, L. and Bødker, S. (1997). Constructing Common Information Spaces. In Proceedings of the European Conference on Computer-Supported Cooperative Work (ECSCW'97) .Kluwer Academic Press, 81-96, Lancaster, England.

Bannon, L.J. and Kuutti, K. (1996). Shifting perspectives on organizational memory: from storage to active remembering. In Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences, 156-167.

Callon, M. (1991). Techno-Economic Networks and Irreversibility. In J. Law (ed.) A Sociology of Monsters, pp.132-161. London: Routledge.

Carmel, E. (1999). Global Software Teams: Collaborating Across Borders and Time Zones. Prentice Hall.

Carmel, E. and Agarwal R. (2001). Tactical Approaches for Alleviating Distance in Global Software Development. IEEE Software, IEEE, March/April 2001, 22-29.

Goodall, K. and Roberts, J. (2002). Repairing management knowledge-ability over distance. Judge Institute of Management, Cambridge University, Research Paper No.2002/5

Gulati, R. (1995). Does familiarity breed trust? The implications of repeated ties for contractual choices in alliances. Academy of Management Journal, 38: 85-112.

Herbsleb, J.D. and Grinter, R.E. (1999). Architectures, Coordination and Distance: Conway's Law and Beyond. IEEE Software, IEEE, September/October, 63-70.

Herbsleb, J.D. and Moitra, D. (2001). Global Software Development. IEEE Software, IEEE, March/April, 16-20.

Lanubile, F., Damian, D. and Oppenheimer H.L. (2003). Global Software Development: Technical, Organizational and Social Challenges. ACM SIGSOFT Software Engineering Notes, 28 (6), November 2003.

Lave, J., & Wenger, E. (1991). Situated learning: Legitimate peripheral participation. Cambridge: Cambridge University Press.

Malone T. W. and Crowston, K.(1994) The interdisciplinary study of coordination. Computing Surveys. 26: 87--119.

Parkhe, A. (1993). 'Messy' research, methodological predispositions, and theory development in international joint ventures. Academy of Management Review, 18: 227-268

Prikladnicki, R., Audy J.L.N. and Evaristo, R. (2003). Global Software Development in Practice Lessons Learned . In Software Process: Improvement and Practice, 8, 267-281.

Robinson, M. and L. Bannon (1991). Questioning representation. In Ecscw '91: The european conference on computer supported collaborative work.Amsterdam: Kluwer Academic Press.

Sahay, S., Nicholson, B., and Krishna, S. (2003) Global It Outsourcing: Management of Software Development Projects. Cambridge University Press.

Schmidt, K. and Bannon, L. (1992). Taking CSCW seriously. CSCW Journal, 1 (1), 7-40.

Star, S. L. and Griesemer J. R. (1989). Institutional Ecology, 'Translations,' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907 - 1939.

Walsham, G. (2001) Making a world of difference: IT in a global context. Chichester: Wiley.