

Defect Tracking Systems in Global Software Development – a work practice study

Gabriela Avram, Anne Sheehan and Daniel K. Sullivan
Interaction Design Centre, Department of Computer Science & Information Systems, University of Limerick, Ireland.
{gabriela.avram, anne.sheehan, daniel.sullivan}@ul.ie

Abstract. This paper focuses on how the use of common tools within the software development process facilitates the emergence of a common language and shared understanding in distributed software development teams. The paper presents the case of a software development team forced by circumstances to use two bug tracking systems in parallel in a pre-release phase; the consequences of this unusual situation make visible the practices and culture that evolve around tool use. It also illustrates that the adoption of a new tool is not simply a straightforward matter of replacing one with another, tools being not only technical artifacts, but also supporting mechanisms for the articulation and emergence of work practice.

Introduction

The presence of common systems for handling content has been indicated as a vital first step in establishing a successful distributed software development project[5]. Casey and Richardson [3] suggest that selecting a shared configuration management system is an appropriate step to supporting the distribution effort and that sharing similar levels of access across sites could aid the establishment of a common team perspective and counter the challenges presented by the distances inherent to the situation.

In recent years, several studies have emphasized the role of social, cultural and organizational issues in working with distributed software teams [2] [6] [8] [10].

A detailed account of actual work practices in addition to the technical and strategic aspects of global software development is needed for a more in depth understanding of the challenges encountered and the impact of potential solutions. Schmidt [11] advocates that the study of work practice seeks to dismantle the common-sense conceptions of work. The hidden practices are disclosed and the intricate ways and means by which the work is actually accomplished are made discernible.

In the case presented below we examine the actual work practices relating to defect resolution within the global software development process. We suggest that in order to understand the differences in how work is performed at the different distributed sites engaged in the software development, we can make use of what is common, such as the types of software applications in use at the sites. This paper will focus on one set of tools that are used to support that development work, namely defect or 'bug' tracking systems.

2. Research Methods

We mainly used qualitative methods in our field research. The use of qualitative data in software engineering is becoming more prominent [4] . As indicated by Dittrich et al. [4] qualitative research has its main strength in exploring and illuminating the locally situated practice of software engineering. The research approach we adopted stresses the value of examining a situated focus, of seeing the global in the context of local work, and the potentially broader picture in the specific evidence of the work that is performed.

Our data collection and analysis methods include observation, document analysis, in-context interviews, audio recording, focus groups and workshops. In the case described below, we observed the team in its own work environment. The research team was granted access to the project document repository and its members were added to the development team's mailing list. We were also able to install and use the instant messaging system used by the developers. This enabled us to reach a better understanding of the ongoing activities and to continue our observation even when we were not present in the site. We also conducted both formal interviews and informal discussions with various team members and were allowed to observe various teleconferences. We maintained a diary and took detailed notes for every day spent in the field.

3. Defect Tracking Systems

Configuration management, version control and defect tracking systems are among the most common tools used in software engineering. It is through the use of these tools that architects, developers, quality engineers and managers build a shared understanding of the work to be completed and the status of product

development. In this observational study we focus particularly on the use of defect tracking systems.

Defect tracking is the process of finding defects in a product. Defect tracking systems can come in the form of discrete systems where details of the defect and the status of work associated with resolving it are stored or they may be part of an integrated suite of configuration management tools, where the status of the defect may act as a trigger or key for other events within the system.

Lethbridge et al. [9] indicated that developers view defect tracking applications as important repositories of historical information. They consciously or subconsciously make value judgments that place defect-tracking applications in the prime position of documentation that is worthwhile maintaining. Their research based on interviews with developers found that defect tracking applications contained the most frequently updated documents in practice. Hence defect tracking documents form an accurate and up to date account of the work that has been performed.

4. The “Broken Bridge” Case

Our field site was the Irish subsidiary of a large multinational company. The development project in focus here started in January 2005 with a core team consisting of five developers and a quality engineer who had worked together previously under the same team leader. When our observational study began in January 2006 this collocated team was collaborating with both globally distributed individual team members and other development teams. The product under development utilized the functionality of two additional underlying innovative technologies, which were being developed by two separate project teams distributed across Germany, the east coast of the USA and Australia.

When our study took place the Irish team was using CVS, a well-known version control system, as a code repository, and a separate in-house developed defect tracking system, which will be referred to here as BTS (Bug Tracking System). The BTS was a stand-alone system based on a Lotus Notes application, which had evolved over more than a decade of in-house use. As is common in most defect tracking systems, when a defect was uncovered all relevant information was captured and included in BTS in the form of a defect report. The BTS also allowed the addition of rich context information (history, images). Furthermore, the ability to manipulate the documents based on the concept of differing “views” was used as a key aid in allocating, managing and undertaking work. Views within databases are a means of selecting and sorting items placed in the database. Various “shared Views” which were considered useful had evolved in the team over many years and provided a common and shared perspective on how to view the status of the development work. Changes to a defect status caused the related documents to move from one View to another.

The core development team had extensive experience using the BTS System. As the larger multinational company was the result of various mergers and acquisitions, it was company policy to grant its teams as much freedom as possible in selecting the tools they wanted to use. Hence, there were no standard toolsets in use across the company.

The challenge for the Irish team arose in January 2006 when the decision was taken to integrate their application with the two underlying but separately developed technologies for a release scheduled for July 2006. Whilst the different teams were part of the same global organization, they had inherited significantly different cultures and used different software development toolsets locally. One development team was distributed across several locations (Germany, US, Australia) and included approximately 300 people. A second team involved in the release was collocated in Germany. Both of these teams were using a version control/configuration management system - which we will refer to as CDTS (code & defect tracking system) - which included integrated defect tracking functionality.

Once the integration decision had been made, the source code developed by the Irish team had to be checked into the CDTS. The use of the CDTS as the unique code repository was unavoidable as the teams had to have a common framework and standard when referring to problems caused by the integration of the three software products. However, in the short-term, with deadline pressure mounting, the team in Ireland had to decide whether to continue using the BTS in parallel with CDTS until the first integrated release date.

When the integration decision was taken, a considerable number of bugs (around 300) had already been raised in the BTS and the local developers were already working on resolving these. The team believed there was insufficient time to stop the ongoing effort and transfer the existing number of bugs to the new CDTS system. Furthermore, local work practices were heavily reliant on the BTS, as it incorporated the experience of the team gained over several years. The lack of knowledge and familiarity with the CDTS, combined with the perceived lack of time to investigate the CDTS, led to a belief that there was a lack of functionality to support existing key work practices such as the triage of defects which was considered critical with a release date approaching. The perception amongst the local team was that giving up BTS would have required a radical change in their work practices. At the time, keeping the local system in operation in parallel appeared the only reasonable solution, providing a safety net for getting the internal work done, whilst also adapting to a new tool and insuring good communication and coordination with their colleagues overseas.

The initial plan was to adapt an existing software application that would have acted as a form of “bridge” between the two defect tracking systems. The intention was that this “bridge” would populate the CDTS with the existing defects from BTS and subsequently automatically ensure synchronization of both the defect tracking systems. However, it never worked as anticipated and impacted on both existing work practice and the adoption of CDTS. To illustrate the difficulties encountered, we will present two situations that arose during our observations.

1. During the two months prior to the release, bug fixing became a critical activity. The team leader, two architects and quality assurance leader met twice a week – and as the deadline approached, daily - to prioritize unresolved defects. One of the architects was located in Germany and so these meetings had to be computer-mediated. The QE leader also worked from home occasionally and participated in meetings from there. The routine was to have the on site participants collocated in a meeting room and a teleconference line open with the remote participant(s). The team leader usually chaired these meetings. He told the other participants what specific View he was using in the BTS, and this enabled them to look at the same list of defects, ordered in the same way. Switching between Views was easy and this shared practice made the triage easier. When the team started using CDTS in parallel, the same defect had to be looked at in both BTS and CDTS, because on some occasions, the information contained in the two systems was different. Furthermore, in order to coordinate with the other participants, CDTS defect codes and SQL queries had to be shared via instant messenger – practice that was created ad-hoc to enable participants to “stay on the same page”.

2. During their daily activity, developers received automated e-mails from both the BTS and the CDTS regarding assigned defects. This updated information had to be reconciled with the priorities assigned to defects by the team managers. While the BTS included a View that allowed them to see the list of defects assigned to them ordered by priority, they also had to check the defect status in the CDTS. On occasions, the two systems were out of sync:

- defects that had been reassigned to other developers were still showing under the same developer name in CDTS;
- defects had a different status in the two systems (closed versus open).

Resolving these irregularities created extra-work and made the integration and release effort much harder. Manual updating (or at least cross-checking between BTS and CDTS) proved to be the only viable solution for keeping the two defect tracking systems synchronized.

In July 2006, the target was met as planned. In August, the Irish team participated in a post-mortem analysis. The use in parallel of the two defect tracking systems was highlighted by all the developers as an experience not to be repeated. In hindsight, the decision to keep using the BTS appeared to be inappropriate. However, the team members concluded that they learned a lot about both their work practice and the tools from the experience. By continuously comparing the two systems, they learned how to perform the same operations in CDTS as in BTS. Even though the CDTS had less functionality (e.g. did not support Views) they were able to develop new work practices to cope with the use of the new system.

5. Discussion

In distributed software development, having a defect tracking application that is equally accessible to all team members irrespective of their role and which is shared across all sites is extremely important for a number of reasons. In recent research [3] those reasons have been highlighted as being even more pertinent in the global environment. Amongst the possible roles played by such systems, Kobitzsch et al. [7] highlight their contribution as a means to initially introduce and later to further inculcate common standards across an organisation that is engaged in collaborative work. The common standards create a basis for shared codes and meanings, and facilitate communication and coordination. In our case the Irish team was not unwilling to adopt the new tool (CDTS) as they recognized the key role of a common defect tracking system to ensure good communication and coordination with their global colleagues. It was simply deadline pressure that forced the decision to work with both systems in parallel for a time.

In the case that was examined here, a tool (BTS) had to be replaced with another (CDTS) due to distributed collaboration constraints. Our observations illustrate the fact that tools are highly embedded in their context of use, and each group creates its own practices around a tool. When a tool is set aside and another adopted, people attempt to re-create the old practices around the new tool, which may not be always possible. During the initial integration period, the Irish team maintained two different perspectives on each defect: that provided by the BTS, where defect codes had a structure and a meaning they could read instantaneously, and where there was a rich history for each defect, and that of CDTS, where the information was definitely less rich, but the defect codes were shared with their counterparts from the other teams. The members of the Irish team learned how to use the CDTS while continuously comparing it to the BTS and trying to find equivalents for each of their existing practices. From this perspective, they are observed to go through a process of enculturation [1]. Their appreciation of the functionality exhibited by BTS which was found lacking in CDTS made them more aware of and externalized those work practices centered on it.

For the team, defect reports represent a check and reference point external to the development work on how the work is conducted. As explored by Grinter [5] the model of work adopted is reflected in the way in that work is performed and also in how the tools available are used by those performing the work. Developers saw bugs as tasks assigned to them; testers saw them as outcomes of their work; software architects were monitoring them via Views in order to quickly detect areas with problems and come up with new solutions; the team manager was prioritizing them and managing the resources for resolving them; at the higher management levels, bugs were seen as indicators of team performance. This illustrates how the defect tracking system operated as a mechanism for articulating different work practices.

6. Conclusion

In the example presented we draw attention to the practices people develop around specific tools, in this case, defect tracking systems. Though there are a variety of perspectives through which the case could be examined we highlighted just two, namely: the key role of common standards and tools in ensuring good communication and coordination, as well as, the role tools play in supporting the articulation of work practice. Other points currently under discussion include the use of tools as a form of organizational memory and the value of acting in public.

Yamauchi et al. [12] have examined what was used in successful situations such as open-source development and suggested that quite lean technological solutions in combination with a willingness to make an effort on the part of all participants can overcome the majority of difficulties. This is consistent with our findings and leads to a number of questions regarding the impact on research and design of future tools:

- Is it possible to design collaboration tools that would make work practices more visible? In the situation described, work practices were made visible by the transition process. Would more visible work practices bring any benefits? (for example: facilitate the enculturation process? make the work more traceable? contribute to the organizational memory?)
- What should be the balance between global standards, codes, organizational rules, and local work practices in GSD?

7. Acknowledgments

This work is part of the socGSD project at the University of Limerick. socGSD is one of the LERO (the Irish Software Engineering Research Institute) cluster projects funded under PI grant 03/IN3/1408C by the Science Foundation of Ireland (SFI). We wish to thank our informants in the observed company for allowing us to interview and live with them on site for many days.

8. References

- [1] Brown, J. S. and P. Duguid (1991). "Organizational Learning and Communities-of-Practice: Toward a unified view of working, learning and innovating." Organization Science 2(1): p40, 18p.
- [2] Carmel, E. and R. Agarwal (2001). "Tactical approaches for alleviating distance in global software development." IEEE Software 18(2): 22-29.
- [3] Casey, V. and I. Richardson (2004). Practical Experience of Virtual Team Software Development. EuroSPI 2004 Industrial Proceedings, Trondheim, Norway.
- [4] Dittrich, Y., M. John, et al. (2007). "Editorial: For the Special issue on Qualitative Software Engineering Research." Information and Software Technology 49(6): 531-539

- [5] Grinter, R. E. (1995). Using a Configuration Management Tool to Coordinate Software Development. Organizational computing systems, Milpitas, California, United States.
- [6] Herbsleb, J. D. and D. Moitra (2001). "Global Software Development." IEEE Software **18**(2): 16-20.
- [7] Kobitzsch, W., D. Rombach, et al. (2001). "Outsourcing in India." IEEE Software.
- [8] Krishna, S., S. Sahay, et al. (2004). "Managing Cross Cultural Issues in Global Software Outsourcing." Communications of the ACM.
- [9] Lethbridge, T. C., J. Singer, et al. (2003). "How Software Engineers Use Documentation: The State of the Practice." IEEE Software **Volume 20** (Issue 6).
- [10] Sahay, S., B. Nicholson, et al. (2003). Global It Outsourcing: Management of Software Development Projects, Cambridge University Press.
- [11] Schmidt, K. (1999). The critical role of workplace studies in CSCW. Workplace Studies: Recovering Work Practice and Informing Design. C. Heath, J. Hindmarsh and P. Luff. Cambridge, UK, Cambridge University Press.
- [12] Yamauchi, Y., M. Yokozawa, et al. (2000). Collaboration with Lean Media: how open-source software succeeds. Computer Supported Cooperative Work, Philadelphia, Pennsylvania, United States, ACM Press.

Gabriela Avram is a senior research fellow at the Interaction Design Centre, University of Limerick, currently working on a project focusing on human and social aspects of globally distributed software development called socGSD (Social, Organizational and Cultural Aspects of Global Software Development). She is particularly studying collaborative work practices in this context. Coming from a Knowledge Management background, Gabriela's area of expertise also includes social software, social networks and online facilitation.

Anne Sheehan works on the socGSD project in the Interaction Design Centre at the University of Limerick. Her current research focus is primarily on knowledge and learning within distributed software development, in particular mentoring and the evolution of expertise within the context of software development.

Daniel Sullivan is a master student working on the socGSD project in the Interaction Design Centre at the University of Limerick. His current research is focused primarily on the use of software tools in the context of collaborative work practices. He has over a decade of professional experience in the software development industry.