Understanding the Use of a Bug Tracking System in a Global Software Development Setup

Dhaval Vyas[1], **Tara Capel**[1], **Deven Tank**[2] and **David Shepherd**[3]
[1] Queensland University of Technology (QUT), Brisbane, QLD, Australia.
[2]Schneider Electric, Bangalore, KA, India.
[3]ABB Corporate Research, Raleigh, NC, USA.
{d.vyas | tara.capel}@qut.edu.au; deven.tank@schneider-electric.com; david.shepherd@us.abb.com

ABSTRACT

Bug fixing is a highly cooperative work activity where developers, testers, product managers and other stake-holders collaborate using a bug tracking system. In the context of Global Software Development (GSD), where software development is distributed across different geographical locations, we focus on understanding the role of bug trackers in supporting software bug fixing activities. We carried out a small-scale ethnographic fieldwork in a software product team distributed between Finland and India at a multinational engineering company. Using semi-structured interviews and in-situ observations of 16 bug cases, we show that the bug tracker 1) supported information needs of different stake holder, 2) established common-ground, and 3) reinforced issues related to ownership, performance and power. Consequently, we provide implications for design around these findings.

Author Keywords
CSCW, Software Development, Bug trackers, Design

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

Software development is increasingly a distributed work. For multinational software companies and software offshoring services, this poses temporal, geographical and cultural challenges that may have strong effects on collaborative development efforts. The CHI and CSCW communities have been studying different collaborative aspects related to Global Software Development (GSD) for several years (Bjørn et al., 2014a,b). Of particular interest is the use of bug tracking systems or simply bug trackers (Avram et al., 2009; Bertram et al., 2010; Breu et al., 2010; Halverson et al., 2006). A bug tracker is a software tool that functions as a database of issues, defects or feature requests. It helps organizations manage the processes of bug reporting, tracking and resolving; and serves as an important coordinative mechanism through which multiple stake-holders such as developers, testers, product managers, and customers support professionals can work towards bug fixing activities. The use of bug trackers is fundamentally a social process. In addition to having a list of issues and defects, bug trackers incorporate a huge amount of organizational memory (Ackerman and Halverson, 1998) and serve as a boundary object (Star and Griesemer , 1989) for different communities of practices (e.g. developers, testers).

We carried out a small-scale ethnographic study at a multinational engineering company's software division that was distributed between India and Finland. The team specialized in developing and maintaining a SCADA (supervisory control and data acquisition) product. Over a period of three months, we studied their bug fixing activities using semi-structured interviews and observations of their *in-situ* bug fixing activities. In total we studies 16 bug cases. We took a holistic view on the bug fixing process: starting from when bugs were reported to when they were fixed. Our results show that the bug tracker supported information needs and exchanges between different stake-holders throughout the bug fixing process. The bug tracker was used as a dynamic, evolving source of information that was central to coordination of the stake-holders. It was used to support comprehensive reporting of all kinds of bug related activities, which helped in establishing common ground across different members. We also found that bug tracker embodied several organizational aspects such as product ownership, performance measures and power differences between Indian and Finnish sides.

This work contributes to the HCI research in two ways: 1) it provides an empirical account of how a bug tracker (as a cooperative tool) is used within a GSD setup; and, 2) it provides implications for designing collaborative features in bug tracking systems.

RELATED WORK

HCI researchers have long been highlighting different collaborative aspects of software development practices (Grinter, 1996; Gutwin et al., 2004; Halverson et al., 2006; Vyas et al. 2014). In particular, research on bug fixing and bug trackers have proved to be quite fundamental in understanding collaborative practices of software development. Based on a qualitative study involving 15 developers in a co-located setting, Bertram et al. (2010) highlighted that bug tracking systems were used as 1) a knowledge repository where activities from different stakeholders were getting stored, 2) a boundary

OzCHI '15 , December 07 - 10 2015, Melbourne, VIC, Australia
Copyright is held by the owner/author(s). Publication rights licensed to ACM.ACM 978-1-4503-3673-4/15/12... $15.00
http://dx.doi.org/10.1145/2838739.2838790.

object to fulfill the needs of different stakeholders, and 3) a communication and coordination hub. In their field study Halverson et al. (2006) highlighted how bug trackers provided a locus for negotiation between different stake holders and worked as a knowledge repository. Within the GSD setup, Avram et al. (2009) studied the use of two simultaneously used bug trackers and showed how developers created bridge between the two and how localized solutions were used to keep the coordination between two development sites intact.

Developers require different types of information in order to fix bugs. Bettenburg et al. (2008) explored a set of information required in bug reports based on the responses from 466 developers. Their study highlighted that bug reports often have a strong mismatch between what developers needed and what information was provided by users. Based on the analysis of 600 bug reports, Breu et al. (2010) developed categories of questions that are frequently asked by developers to the users who reported bugs. Frequently asked questions were related to missing information, clarifications, triaging, debugging, correction, status enquiry, resolution and process. Other similar studies included the use of card sorting methods (Just et al., 2008) for exploring how bugs can be reported and resolved in the form of design recommendation for new bug tracking systems.

THE STUDY

Settings

Figure 1 shows the overall setup of the SCADA product development division. The software development was divided between India and Finland, where each site had a smaller team involving developers, testers (including QA professionals) and local team managers. The product manager was based in Finland, as the software product was originated from Finland. He managed the overall development and release of the product, whereas the individual team managers managed local resources. The two development teams had been working together for 9 years and had been sharing workloads. The product had customers from around the world and they had support from localized customer support professionals. The bug tracker in use was a Lotus Notes based product maintenance tool.

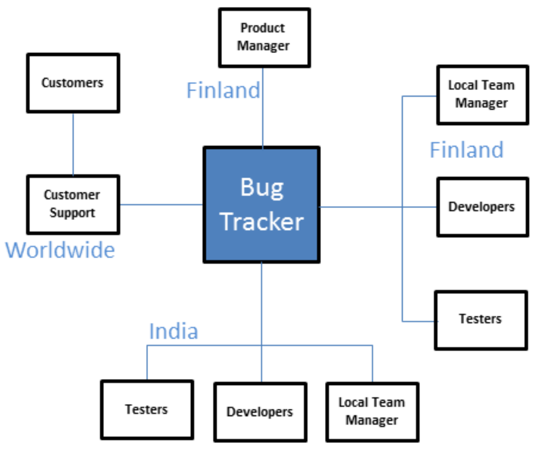
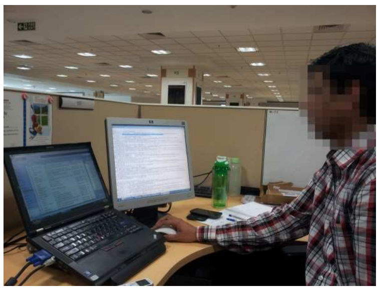


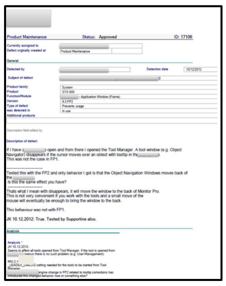
Figure 1: Setup of the SCADA development division

Bugs and issues were either reported through the customer support channel or were internally reported by the development teams. The customer bugs were given the highest priority due to contractual issues. Once a bug was reported to the bug tracker, it was automatically assigned to the team with the right expertise. The local team managers would allocate suitable developers and testers on the bug cases and the product manager would make decisions on how to proceed with the bug fixing plan. The development of the SCADA product was divided into several modules and specific developers from both the teams were given 'ownership'. Developers with specific module ownership were experts in fixing bugs related to those modules. The company would shift the ownership of different modules based on the performance of individual developers.

Methods

Over a period of three months, we conducted an ethnographic study at the Indian site of the company's development center. The Indian team had 5 developers, 3 testers, a team manager and two members of the Finnish development team who were working in India at that time. We aimed at gaining access to the team's natural practices related to the software bug-fixing process. We studied 16 bug fixing cases *in-situ*, where we followed the activities of developers and testers throughout the fixing process. We video recorded developers' real-time software bug-fixing activities at their workspace (figure 2). With minimum intrusion to their ongoing work, we captured different stages of bug-fixing activities such as bug reproduction, debugging, code change and testing. Following observations, we carried out semi-structured interviews at developers' workspace, where we asked our participants questions related to their bug-fixing process and practices.





[redacted]

Product Maintenance | Status: Approved | ID: 17106

Currently assigned to: [redacted]
Defect orginally created at: Product Maintenance

General

Detected by: [redacted] | Detection date: 10/12/2012
Subject of defect: [redacted]2
Product family: System
Product: SYS 600
Function/Module: [redacted] - Application Window (Frame)
Version: 9.3 FP2
Type of defect: Prevents usage
was detected in: In use
Additional products

Description field edited by

Description of defect:

If I have a[redacted] open and from there I opened the Tool Manager. A tool window (e.g. Object Navigator) disappears if the cursor moves over an oblect with tooltip in th[redacted].
This was not the case in FP1.

Tested this with the FP2 and only behavior I got is that the Object Navigation Windows moves back of the [redacted]
Is this the same effect you have?

Thats what I mean with disappears, it will move the window to the back of Monitor Pro.
This is not very convenient if you work with the tools and a small move of the mouse will eventually be enough to bring the window to the back.

This behaviour was not with FP1.

JK 10.12.2012: True. Tested by Supportine also.

Analysis

Analysis *
JH 10.12.2012:
Seems to affect all tools opened from Tool Manager. If the tool is opened from [redacted] menus there is no such problem (e.g. User Management).

MVI 2.1[redacted]
_LEADER_DIALOG setting needed for the tools to be started from Tool Manager.
[redacted] engine change in FP2 related to tooltip corrections has introduced this changed behavior now or something else?

Figure 2: Developer fixing a bug and an example Bug Report.

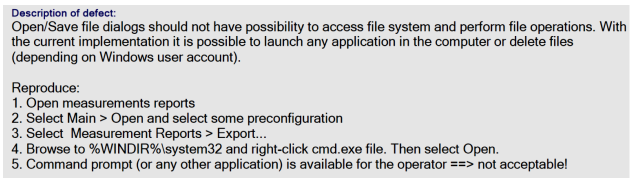
At the end of our sessions, we collected artefacts such as bug reports and documents that our participants were using in their bug-fixing activities. Below, we present results from a qualitative analysis of our collected data, where we used affinity diagraming technique (Holtzblatt et al., 2005) to develop patterns and themes from the data.

FINDINGS

Supporting Information Needs

In a distributed setup, different stake-holders involved in the bug fixing process require different types of information at different stages. We observed that the Lotus Notes based bug tracker was used as a tool to support information needs and exchanges between stake-holders.

One of the first steps developers take in the bug fixing process is to carry out bug reproduction. Bug reproduction is a process by which developers try to recreate the situation within which a bug was encountered. It also allows developers to experience firsthand the steps that lead to a bug and how the bug behaves. When a bug is reported by a customer, it is through bug reproduction that developers can determine how the software was being used by the customer and what configuration and settings the customer had in place. To carry out bug reproduction, developers require access to specific information which led to the problem. The most important information in this case would be 'steps for reproduction', 'screenshots of the bug', 'trace logs', 'system configurations' and in some cases highly specific details from customers' sites. Often, bugs were reassigned from the Finnish team to the Indian team and vice versa. We observed that the bug tracker was used to support such information exchange between developers and customer support professionals (who have direct access to customers) and between the two development teams when bugs were reassigned. The following is an example of 'steps for reproduction' found on one bug report. It was send by one development team to another.



Description of defect:
Open/Save file dialogs should not have possibility to access file system and perform file operations. With the current implementation it is possible to launch any application in the computer or delete files (depending on Windows user account).

Reproduce:
1. Open measurements reports
2. Select Main > Open and select some preconfiguration
3. Select Measurement Reports > Export...
4. Browse to %WINDIR%\system32 and right-click cmd.exe file. Then select Open.
5. Command prompt (or any other application) is available for the operator ==> not acceptable!

Figure 3. A screenshot from a bug report

The bug tracker was used to contain a highly diverse set of information including textual descriptions of bugs, images, screenshots, and even videos in cases when a dynamic behavior needed to be communicated. A developer commented:

> "Our partner team in Finland received this bug from a customer. When they could not fix it, they sent this bug to us. They also sent a video and trace-logs. This bug occurs once in 20 times. So, it is really hard for us to find out the exact reasons. The video gives a dynamic view of the bug."

It was quite often the case that developers did not receive adequate information through the bug trackers and they had to request more information. In those cases, developers used bug trackers to communicate these issues so that local and global managers and other team members can provide directions. The role of local and global managers is very important in determining ways to fix bugs. They determine the importance and criticality of bugs and

<u>Implication</u>: *Connecting bug trackers with other tools and communication channels to provide information from multiple sources.*

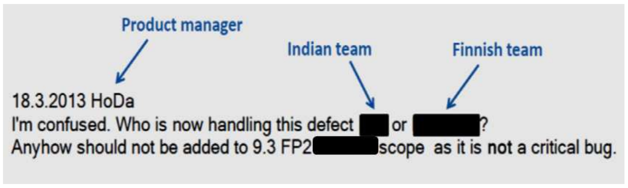
Establishing Common Ground

We found that the team members from both the sides made a large amount of information available through the bug trackers. In addition to the information such as description of bugs, steps to reproduce, trace logs and test results which are quite common to see in the bug trackers, we saw that team members provided email trails, informal discussions between members, corrected code patches, changes in planning and work allocations and so on. During the interview it was found that developers were required to do comprehensive reporting on the bug trackers in cases when there are legal issues with customers. As we mentioned earlier, it was often the case that developers missed relevant information in the bug tracker from customers. At the time of audit, a bug tracker with a comprehensive description of all the interactions and exchange of informational artefacts (e.g. test results, trace logs) was useful.

There are several cases where bugs were reassigned from one team to another. As a good practice, both the teams provided a comprehensive information into the bug tracker, so when the bug is reassigned, the other team can readily use information from the bug trackers. A developer gave the following comment:

> "These code patches are not for our managers. They do not need to go into such details. We provide these for the developers and testers in Finland. When they have to take over from us, then we don't have to spend time in explaining such things."

We observed that even when comprehensive information was archived in bug trackers, keeping track of the progress and extracting important and relevant information was getting difficult for some members. The bug tracker did not have a mechanism that allowed automatic notification of changes or updates on progress made in bug fixing activities. The developers needed to use separate communication channels such as instant messaging and emails to notify relevant updates. In some cases, both the development teams worked together to resolve bugs. This often led to confusions about the progress in bug fixing. Figure 4 is an extract form a bug case that we studied. Here the product manager indicates that he is not aware of the recent updates on the bug.



18.3.2013 HoDa
I'm confused. Who is now handling this defect [redacted] or [redacted]?
Anyhow should not be added to 9.3 FP2[redacted]scope as it is not a critical bug.

Figure 4. A screenshot from a bug report

<u>Implication</u>: *Bug trackers should incorporate context-sensitive 'awareness' features where it can send out relevant notification messages to appropriate members who may be affected by the changes made in the bug tracker.*

Performance, Ownership and Power

Interestingly, our analysis of the bug reports and interviews with team members revealed that the bug tracker embodied several organizational and social constructs. As we mentioned earlier, the company had given ownership of different modules of the SCADA software to expert developers. Based on the performance of developers this ownership was transferable to a more suited developer. Developers with ownership of specific software modules were responsible for all the bugs and issues related to their modules. It was visible in our findings that although the Indian team started merely as an additional support to the Finnish team, over a period of time it started getting ownership of several software modules of the product. Ownership served as a sense of pride and identity issue for Indian developers and was a high performance indicator, locally. In the bug tracker, this was highlighted very clearly as the 'buck' stopped with developers who had ownership of specific modules. These developers also used the bug tracker as an opportunity to showcase their depth of knowledge but providing detailed insights about how to fix bugs, multiple ways of fixing them and effects they may have on the software product itself. A developer gave following account on his ownership of a few software modules:

> "I have been working here for five years. I have ownership of the 'trends' module and about get ownership of one more important module. I have a few junior developers with me but when I am assigned a customer bug, I don't rely on anybody. Even when the bug is easy to fix."

Having the ownership of a module also meant that the local development team would have a much stronger say in suggesting new development features to the product manager who was based in Finland. This was apparent from the bug reports too, where developers with ownerships would guide bug fixing in specific ways. During our interviews and analysis of bug reports, we found that the number of successful bug fixes was a performance indicator. Some novice developers tended to create smaller, less significant bugs to showcase an increase in their bug fixing ability. They often struggled with difficult bugs and had to wait for help from expert and more senior developers who may be sitting in Finland.

This finding was particularly novel as it showed how the bug tracker embodied certain organizational and social constructs such as the product ownership, which was connected to developers' social identity. We believe that adding a feature in the design of bug trackers that can facilitate communication between different levels of developers can benefit the bug fixing process.

Implication: Bug trackers should facilitate communication with expert developers within organization. Expert developers can indicate their area of interests. The bug trackers can show experts' availability and whenever they help is required.

DISCUSSION AND CONCLUSION

It is clear that a bug tracker is not just a database of prioritized list of issues and feature list of a software product. Our efforts echo previous studies, which showed bug trackers as a coordination (Bertram et al., 2010) and negotiation (Halverson et al., 2006) tool. Where the study by Bertram et al. (2010) focused on understanding the different roles a bug tracker plays (knowledge repository, boundary objects, and so on) and stake-holders' perceptions of bug trackers (a to-do list, priority list, and so on), the current study focused on providing empirical evidences of how a bug tracker carries certain useful information that is being communicated between different stake-holders (e.g. steps for bug reproduction). We also showed that bug trackers embody social and organizational constructs such as ownership, performance and power within the developmental setup. In particular to the GSD setup, our findings showed how the bug tracker balanced the power issues between the Indian and Finnish development divisions. Our findings also showed that bug trackers are dynamic, evolving digital artefacts that support information needs and builds common-ground between different stake-holders located across different geographical sites.

ACKNOWLEDGEMENTS

We thank all the software developers and testers who participated in our study. We also thank Thomas Fritz for collaborating and providing useful guidance in this project.

REFERENCES

Ackerman, M. S., & Halverson, C. Considering an organization's memory. In Proceedings of the ACM conference on Computer supported cooperative work, pp. 39-48. (1998), ACM.

Aranda, J., & Venolia, G. The secret life of bugs: Going past the errors and omissions in software repositories. In Proc. of ICSE'09. IEEE Computer Society. (2009) pp. 298-308.

Avram, G., Bannon, L., Bowers, J., Sheehan, A., & Sullivan, D. K. Bridging, patching and keeping the work flowing: defect resolution in distributed software development. Computer Supported Cooperative Work, 18 (5-6), 477-507, (2009).

Bertram, D., Voida, A., Greenberg, S., & Walker, R. Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. In Proc. CSCW'10, ACM. pp. 291-300.

Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., and Zimmermann, T. What makes a good bug report? In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. (2008) pp. 308-318.

Bjørn, P., Esbensen, M., Jensen, R. E., & Matthiesen, S. Does Distance Still Matter? Revisiting the CSCW Fundamentals on Distributed Collaboration. ACM Transactions on Computer-Human Interaction (2014a), 21(5), ACM.

Bjorn, P., Bardram, J., Avram, G., Bannon, L., Boden, A., Redmiles, D., and Wulf, V. Global software development in a CSCW perspective. In Proc. CSCW'14. ACM, (2014b), 301-304.

Breu, S., Premraj, R., Sillito, J., and Zimmermann, T. Information needs in bug reports: improving cooperation between developers and users. In Proceedings of the 2010 ACM conference on Computer supported cooperative work (2010), 301-310. ACM.

Grinter, R. E. Supporting articulation work using software configuration management systems. Computer Supported Cooperative Work (CSCW), 5(4), (1996). 447-465.

Gutwin, C., Penner, R., & Schneider, K. Group awareness in distributed software development. In Proc of CSCW'04. (pp. 72-81). ACM.

Halverson, C. A., Ellis, J. B., Danis, C., & Kellogg, W. A. Designing task visualizations to support the coordination of work in software development. In Proc. of CSCW'06. (2006), 39-48. ACM.

Holtzblatt, K., Wendell, J. B., & Wood, S. Rapid contextual design: A how-to guide to key techniques for user-centered design. (2005), Elsevier.

Just, S., Premraj, R., & Zimmermann, T. Towards the next generation of bug tracking systems. In IEEE Symposium on Visual Languages and Human-Centric Computing, (2008). VL/HCC. 82-85. IEEE.

Star, S.L. and Griesemer, J.R. Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907–39. Social Studies of Science 19, 3 (1989), 387–420.

Vyas, D., Fritz, T., & Shepherd, D. Bug Reproduction: A Collaborative Practice Within Software Maintenance Activities. In Proc. of COOP' 14. (2014), Springer International Publishing, 189-207.