Collaborative Learning in a Software Bug-Tracking Scenario

Shilpa V. Shukla
sshukla@ics.uci.edu

David F. Redmiles
redmiles@ics.uci.edu

Department of Information and Computer Science
University of California, Irvine

KEYWORDS: Software Engineering, Bug Tracking, Computer Supported Collaborative Work (CSCW), Collaborative Learning, Situated Learning, Distributed Learning

1. Introduction

Bug-tracking in software engineering is a distributed work process that involves collaborative learning. We seek to understand this process better through theories which view learning as a situated activity [Suchman 87, Lave 88]. Lave and Wenger argue that learning is a function of the activity, context, and culture in which it occurs (i.e. is situated) [Lave, Wenger 91]. A critical ingredient of situated learning is social interaction [Nardi 93]. People become involved in a community of practice which embodies certain roles of individuals and norms of behaviors. Within this community, learning occurs incidentally to a task. Thus, we examine the bug-tracking scenario with respect to the knowledge that individuals must acquire, and the collaborations necessary to acquire and apply this knowledge. Our long-term goal is to develop this analysis sufficiently to guide the adaptation of software engineering tools (e.g., see [Taylor et al. 96]) to support learning in the collaborative context of bug-tracking and software maintenance.

2. Collaboration in Software Engineering

Software engineering is not a solitary activity but a “multi-person construction of multi-version software [Ghezzi, Jazayeri, Mandrioli 91].” Although a programmer may write a complete program independently from other programmers, a software engineer “writes a software component that will be combined with components written by other software engineers to build a system [Ghezzi, Jazayeri, Mandrioli 91].” Although “programming” may be viewed as an individ-

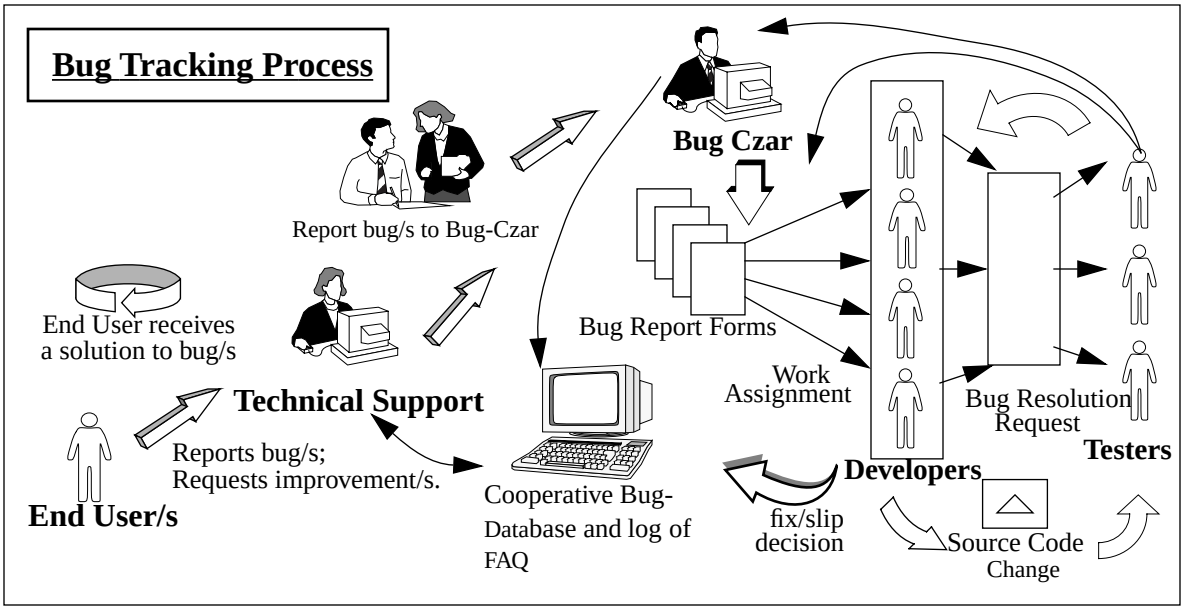

FIGURE 1. Bug Tracking Process

ual activity, “software engineering” is a collaborative activity among members of a team. The members of this team are often distributed among various organizational units and physical locations.

The process of software engineering does not end with the creation of the software product. In fact, of all the phases of a software life cycle (requirements, system and software design, implementation and unit testing, integration and system testing and operation and maintenance), maintenance engages the largest amount of time and money. Maintenance involves correction of software faults as well as modifications to accommodate changes in requirements. These corrections and modifications are time consuming and complex. Discovery of bugs and plans for redesign require technical workers to comprehend existing implementations and designs and plan changes. Current end-users, previous designers and implementers, as well as managers are some of the stakeholders who must be consulted either for approval or to learn what is needed to detect a fault or plan a change. The process is illustrated in Figure 1 and is generally referred to as a bug-tracking process.

3. Collaborative Learning Situations in Software Bug Tracking

The process displayed in Figure 1 shows the interaction among various members of a software bug-tracking team. The roles are as follows: end user, technical support person, bug czar or liaison, and development team members, testers, and marketing team.

When an end user (EU) encounters a bug, the first thing that he or she must do is contact the technical support person to report the bug. The report is usually done either by sending a description of the bug over email or by speaking to someone over the phone. The technical support person (TSP) will first enter this problem into some sort of cooperative bug database. At this point, one of four basic situations may occur.

1. The TSP finds the solution to the bug in the manual for solutions or in the FAQ database (frequently asked questions).

2. The TSP encounters a new bug and solves the problem by using experience combined with rudimentary knowledge about the problem. Thus, the TSP is able to transfer knowledge from past to draw analogies that lead to a solution to the current bug problem.

3. The TSP does not recognize the bug as one encountered previously (either from personal experience or by searching the database) and there is no known solutions. The TSP either personally or electronically interacts with the bug czar (BZ) when such a bug is encountered. From this point, the BZ will be responsible for carrying out the bug-tracking process with the help of the many other stakeholders such as the development team (DT) and the testers (T). The BZ is the member responsible for assigning the job of solving the bugs to the appropriate programmer and maintaining an update of known bugs and solutions to new bugs.

4. The TSP encounters a new problem or bug and can not provide a solution without the help of members from the DT. Programmers of the DT are assigned the job of solving the bug via the BZ. Members of the DT are given “work assignments” described in bug report forms, written by the BZ. DT members rely on many sources to understand how to solve the bug. They may refer to design rationale or other documentation. Once members of the DT complete a work assignment, the cooperative bug database is updated with any bug fixes or changes. DT members also interact with the BZ regarding any questions, requests, or negotiations on making changes. Testers also report back to the BZ after verifying changes or fixes made by the DT as well as updating the cooperative bug database. Other members may also be involved in the above process. For instance once the programmers of the DT solve a bug, Testers verify that the bug no longer exists and that new bugs have not been introduced. The marketing team (MT) may also be involved in informing the EU on bug-fixes.

4. Situated Learning Theory and Bug Tracking

The bug-tracking process described above is a collaborative learning process [Bruffee 93]. In the collaborative learning, “collaborative activities lead to emergent knowledge, which is the result (not summation) of

interaction of the understanding of those who contribute to its formation [Alavi 94, p. 161].” The bug-tracking process involves social (interpersonal) processes by which various stakeholder groups work together (i.e. cooperate and work as a team) to complete the task of fixing bugs.

Alvari describes three attributes of effective learning: 1) active learning and construction of knowledge; 2) cooperation and teamwork in learning; and 3) learning via problem solving [Alavi 94, p. 161]. All three of these attributes occur during the bug-tracking process. Figure 1 illustrates how the knowledge about solving bugs and the process for bug-tracking is stored collaboratively in the organization’s cooperative database. Each stakeholder contributes to the evolving knowledge base of the organization, through various computer mediated tools (such as e-mail, electronic agents that distribute work assignments, a cooperative data base that keeps logs of frequently asked questions (FAQ), documentations of code, rationale of system design, etc.). The access and contribution to this knowledge reflects the collaborative learning process that occurs incidentally during the bug-tracking process.

Although other learning theories may explain some aspects of the behaviors which arise during bug tracking, it is our opinion that situated learning theory is most appropriate. In this theory, learning is “viewed as a special type of social practice associated with the kind of participation” which is referred to by Lave as legitimate peripheral participation (LPP) [Lave, Wenger 91]. Learning according to Lave and Wenger, can be viewed as a “feature of practice, which may be present in all sorts of activities, not just in clear cases of training [Lave, Wenger 91, p. 18].” This theory explains how the effective learner learns through practice. As Figure 1 indicates the bug czar is one of the most active learners during the process of solving a bug. This stakeholder must accept various roles in order to learn to solve a bug. According to Lave, LPP is “not a simple participation structure in which an apprentice occupies a particular role ... rather an interactive process in which the apprentice engages by simultaneously performing several roles [Lave, Wenger 91, p. 23].” LPP is “a way of acting in the world which takes place under widely varying conditions [Lave, Wenger 91, p. 24].” Thus, learning according to Lave and Wenger is “a way of being in the social world, not a way of coming to know about it [Lave, Wenger 91, p. 24].” This theory implies that without engagement, there is no learning. In the case of bug tracking, learning is clearly dependent upon participation and interaction (engagement) among stakeholders in the broader context of the software development and maintenance process.

5. Conclusion

Many researchers view software engineering activities as collaborative in nature, but few have focused on the requirements of learning. Although learning may not be intentional on the part of stakeholders in the bug-tracking process, many of the activities are in fact collaborative learning situations. We seek to explicitly focus on these learning situations, applying the theories of collaborative and situated learning to determine where existing tools for software engineering may be improved.

ACKNOWLEDGMENTS

Thanks are due to David Hilbert, Jason Robbins, Tony Kutscher, and Michael Kantor for providing feedback and thoughtful suggestions for the content of the paper.

REFERENCES

[Alavi 94] M. Alavi, Computer-Mediated Collaborative Learning: An Empirical Evaluation, MIS Quarterly, Vol. No. June 1994, pp. 159-173.

[Bruffee 93] K. Bruffee, Collaborative Learning, The Johns Hopkins University Press, Baltimore, MD, 1993.

[Ghezzi, Jazayeri, Mandrioli 91] C. Ghezzi, M. Jazayeri and D. Mandrioli, Fundamentals of Software Engineering, Prentice Hall, Englewood Cliffs, NJ, 1991.

[Lave 88] J. Lave, Cognition in Practice, Cambridge University Press, Cambridge, UK, 1988.

[Lave, Wenger 91] J. Lave and E. Wenger, Situated Learning, Cambridge University Press, New York, NY, 1991.

[Nardi 93] B. Nardi, A Small Matter of Programming, The MIT Press, Cambridge, MA, 1993.

[Suchman 87] L.A. Suchman, Plans and Situated Actions, Cambridge University Press, Cambridge, UK, 1987.

[Taylor et al. 96] R. Taylor et al., Foundations for the Arcadia Environment Architecture, in P. Garg and M. Jazayeri (eds.), Process-Centered Software Engineering Environments, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 229-241.