Analysis of Work Practices of a Collaborative Software Development Team

Cleidson R. B. de Souza[1,3] John Penix[2] Marteen Sierhuis[3] David Redmiles[1]

[1]*Department of Information and Computer Science*
University of California, Irvine
Irvine, CA, USA

[2]*Computacional Sciences Division*
NASA Ames Research Center
Moffett Field, CA, USA

[3]*Research Institute for Advanced Computer Science*
NASA Ames Research Center
Moffett Field, CA, USA

Abstract

This paper reports preliminary results of a field study of a software development team. This team develops a suite of tools called CTAS, designed to help air traffic controllers manage complex air traffic flows at large airports. We observed that CTAS developers employ two main tools for coordinating their work: a configuration management system and a bug tracking system. These tools allow them to coordinate their activities supporting a high level of parallel development. Communication and cooperation among developers with different roles is achieved using product requests. Future results from our study will provide insights into the complexities of cooperative software development and help to design tools to support it.

Keywords*: Field Study, Empirical Studies, CSCW, Cooperative Work, Cooperative Software Development.*

1. Introduction

Software development is a typical cooperative activity where experts from different domains are necessary. Indeed, developers of large systems spend about 70% of their time working with others[10]. At NASA, this problem is much more difficult because of the increasing complexity of the software being developed. In fact, several efforts to improve the software engineering practices through the dissemination of best practices and infusion of new technologies are taking place at NASA. However, these efforts will only be effective if they address the real ways that people work together to develop software.

Gerson and Star[1] observe that no matter how formal and well-defined a process may seem, there is always a set of informal practices by which individuals and groups monitor and maintain the process, keep it on track, recognize opportunities for action and the necessity for intervention or deviation. In order to understand these practices, the first author conducted an eight-week qualitative study of a software development team using non-participant observation and informal interviews for data collection. This paper reports our initial findings.

2. The Setting

The group observed develops an application called CTAS (Center TRACON Automation System). CTAS is a suite of advisory tools designed to help air traffic controllers manage the complex air traffic flow at large airports. The source code is developed in C and C++ and is about 1,000 K lines long. The development team is divided in two groups: developers and the verification and validation (V&V) staff. Developers are responsible for writing new code, for performing bug fixing and enhancements, and so on. There are 25 developers, including researchers that write their own code. The V&V staff is responsible for testing the software and reporting, keeping a running version for demonstration purposes and maintaining user manuals. This group is composed of six engineers.

3. The Methods

The first author spent eight weeks at the field site. We adopted non-participant observation[3] and informal interviews[5]. In addition to field notes generated by the observations and interviews, we collected software development tool manuals, ISO 9000 procedures, product requests for software changes (PR's), and e-mails exchanged regarding the data and documents.

Initial data collection was centered on understanding the daily work of the developers. During this stage, it became clear that the configuration management (CM) and the bug tracking tools combined with e-mail communication were central to the **coordination** of activities. These results are not surprising based on previous studies by Grinter[2]. Therefore, later data collection was focused on understanding exactly how developers use these tools to perform their work. The interviews focused on observing and understanding the usage patterns of these tools.

4. Initial Findings

An important aspect in fieldwork is the interplay between data collection and analysis: data collection is directed by on-going analysis of the data[9]. Our initial re-

† The authors would like to thank the CTAS group for their help during the fieldwork and the NASA Research Grant NAG2-1555 for the financial support. The first author would also like to thank CAPES (grant BEX 1312/99-5) for the financial support. He is also a faculty at the Department of Informatics, Federal University of Pará, Belém, Pará, Brazil.

sults of this analysis are described in this section. Further analysis is still necessary to obtain more reliable results.

4.1 Parallel Development

We noted that software developers often engage in parallel development. This confirms the results from Perry *et al*.[6], but contrasts with the groups studied by Grinter[2], where developers avoided this situation. Parallel development usually happens when more than one developer has to make changes in the same file. Conflicts might occur when one of these developers check the file back in the repository, because the current version of other developer will be outdated and his modifications might be based on the code that was modified. To update his version, one only needs to merge the other changes back in his code. According to the developers, these conflicts are infrequent and not likely to occur. We plan to use log of the tool usage to test this assumption.

In order to avoid these conflicts, the group adopted the convention that before checking one file in, a developer must send an e-mail to their mailing list describing the files that were changed and the product request associated with the changes. Developers even go to their co-worker's office to talk about the changes that they made or browse the CM repository in order to understand these changes.

Another strategy used by the developers is the *partial check-in*, i.e., to check files in, even when their work is not completely finished. This strategy is employed by those who work with files that are constantly changed by several developers, which makes conflicts more likely. This helps them to prevent those conflicts and avoid several back merges to update their code.

4.2 Conventions

The team studied adopts several conventions in order to cooperate effectively. Conventions are rules or arrangements established in the group, common and accessible to its members[4]. Examples of such conventions are the e-mail that has to be sent before the check-in, or the naming conventions that must be followed when dealing with the CM and bug tracking tools. However, these conventions are not properly supported by their tools which is a source of complaints by the developers. For example, the creation of branches in the CM tool must be based on the PR number recorded in the bug tracking tool. This creation is a cumbersome process that could be easily automated since this is a standard procedure.

4.3 Product Requests (PR's) as Boundary Objects

During the fieldwork, we also identified that PR's are used as boundary objects by members of the team with different roles. Boundary objects are objects both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites[8]. In this group, PR's are used by end-users *liaisons*, developers and testers serving for different functions. For example, when a bug is identified, it is associated with a specific PR. Whoever identified the problem is also responsible for describing '*how to repeat*' it. The developer assigned to repair the bug uses this description to identify and fix it. After that, he must fill a field in the PR that describes how the testing should be performed to properly validate the fix. This information is expanded by the test manager to create the test matrices that are later used by the testers. Another field conveys what needs to be checked by the manager when closing the PR. Therefore, it is a reminder of the aspects that need to be validated.

5. Conclusions and Future Work

Data analysis will be performed using grounded theory[9] and analytical tools like boundary objects and social networks. We also plan to use the Brahms work practice modeling and simulation method[7], in order to simulate the impact of inserting collaborative tools into the development activity. We are particularly interested in understanding the consequences of the parallel development identified in this group: reasons why these developers engage in parallel development might be social, organizational, technological or combinations there of. It is important to identify and understand these reasons so that this practice might be improved, made more effective or safely adopted by other development groups. Initial results indicate that the branching strategy employed in the CM tool properly supports such development, but this is still an open question. We collected log usage data and plan to apply statistical techniques to validate this hypothesis.

6. References

[1] Gerson, E. M. and Star, S. L. *Analyzing Due Process in the Workplace.* ACM Trans. on Office Information Systems, 1986. **4**(3): p. 257-270.

[2] Grinter, R., *Supporting Articulation Work Using Configuration Management Systems.* Journal of Computer Supported Cooperative Work, 1996. **5**(4): p. 447-465.

[3] Jorgensen, D. L., *Participant Observation: A Methodology for Human Studies*. 1989, Thousand Oaks: SAGE Publications.

[4] Mark, G., *et al. Supporting Groupware Conventions through Contextual Awareness*. in *(ECSCW'97)*. 1997, p. 253-268.

[5] McCracken, G., *The Long Interview*. 1988: SAGE Publications.

[6] Perry, D. E., Siy, H. and Votta, L. G. *Parallel Changes in Large Scale Software Development: An Observational Case Study*. in ICSE 1998.

[7] Sierhuis, M., *Modeling and Simulating Work Practices. BRAHMS: a multiagent modeling and simulation language for work system analysis and desing*. 2001: Ponsen & Looijen BV.

[8] Star, S. L. and Griesemer, J. R. *Institutional Ecology, Translations and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology.* Social Studies of Science, 1989. **19**.

[9] Strauss, A. and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Second. ed. 1998, Thousand Oaks: SAGE Publications.

[10] Vessey, I. and A. P. Sravanapudi, *CASE Tools as Collaborative Support Technologies.* CACM, 1995. **38**(1): pp. 83-95.