Hit Squads & Bug Meisters: Discovering New Artifacts for the Design of Software Supporting Collaborative Work

Shilpa V. Shukla†, Bonnie A. Nardi†† and David F. Redmiles†

†Department of Information
and Computer Science
University of California
Irvine, CA 9269 USA
+1 714 824-2703
sshukla, redmiles@ics.uci.edu

††1402 Nilda Avenue
Mountain View, California 94040 USA
nardi@best.com

ABSTRACT

We argue that it is critical to re-evaluate the way we think about artifacts while designing software systems. The notion of artifacts should include aspects of social practice and personal reflection. This new approach to design is especially needed in the design of collaborative systems such as workflow process systems, such as a software bug management system.

Keywords

Artifacts, workflow processes, infrastructure, activity theory, bug tracking, ethnography

INTRODUCTION

Recently, usability research has focused on the study of situated work, including the use of tools, artifacts and the impact of social context of work [1]. Designers of software systems often focus on "basic software tools" such as database software or programming languages. Within a software development scenario, however, the artifacts which serve as the main tools for getting work done are often not these "basic software tools". A simple analogy is that a carpenter places great importance on the use of the hammer while building something. However, the hammer or other such physical tools are not the only artifacts that help get work done. Apprenticeship with other carpenters, informal discussions with other peers, and manuals on "how to do things" are all examples of artifacts that can facilitate a carpenters process of building objects. These artifacts are just as critical as the basic tools such as a hammer.

We use a four level taxonomy of artifacts derived from activity theory from the work of Engeström [3] and Miettinen [4] to recognize the various artifacts, including organizational processes, that sustain a workflow process. Some examples of organizational processes from our study include bug review meetings (coordinated by "Bug Meisters"), issue meetings (such as "Hit Squad" meetings), and informal discussions during lunch or in the hallway. These artifacts include *basic tool* artifacts, *how-to* artifacts, *why* artifacts and *where-to* artifacts.

Our study of Apple's Bug Management process revealed but artifacts which facilitate work are often less obvious, including for example, voice mail, email, informal or non-mandated work practices, impromptu meetings [5]. Software bug management is facilitated by more than just the "basic software tools".

APPLE'S BUG MANAGEMENT PROCESS: AN EXAMPLE PROBLEM CONTEXT

The highly complex and collaborative nature of bug management made for an interesting field study to understand how and what artifacts really get used for software management. Artifacts such as the bug logging and tracking systems are what we view as the "basic software tools". Our study allowed us to expand our view of software tools to consider a broader set of artifacts as tools critical to the bug management of software.

We used an ethnographic approach to look at the bug tracking process at Apple Computer in Cupertino, California. According to an Apple engineer we interviewed, bug fixing makes up 33% of the effort of a typical software development process at Apple-a huge share of the software development process. Bug management at Apple is viewed as a process of recording, tracking, resolving all bugs throughout the course of development.

We define the general roles, of those involved in the bug management process, as the end user, technical support person, bug czar or liaison, development team members, testers, and marketing team. At Apple, we discovered that bug management consists of three primary levels of essential interdependencies.

(1) Essential Organizational Interdependencies — such as Apple's Technical Assistance Centers, Apple Developer Relations, Third parties, and Customer care.

(2) Essential Process Interdependencies — such as Pre-released software failure analysis.

[3] Essential System Interdependencies — such as Radar, Sonar, Vantive. Radar is used for bug-resolution and

tracking, Sonar is used for external developer and escalated bug issue tracking, and Vantive is used for call logging and issue management.

ACTIVITY THEORY

Activity Theory places particular emphasis on the way artifacts and social practices are integrated within a matrix of human activity. Researchers and practitioners need a way to describe artifacts found within organizational processes. Our goal is to use our empirical work on the Apple's bug management system to illustrate different levels of artifacts and to suggest how an awareness of the different levels of artifacts can enhance our understanding of the design process.

STUDY METHODOLOGY

Our study took place over the span of four months. We conducted approximately 33 interviews, took part in several participant observations, and studied Apple's bug management systems and associated documents. For our data analysis we used Activity Theory. This theory helped us develop the expanded definition of tools to build a framework for software designers.

We define four levels of artifacts. Level one artifacts can be described as the physical tools that are tangible and obvious. Level two artifacts include procedures, norms, methods, and rules that serve as tools to support work processes. Level three artifacts serve as models to explain phenomenons such as the motivations and incentives that lead individuals to making particular decisions. And level four artifacts represent reflection in action or future envisionments that may facilitate work processes.

ORGANIZATIONAL ISSUES

The study of organizational issues helped us identify and categorize artifacts (or tools) that facilitate bug management. We present a taxonomy of artifacts we observed based on our observations. The "basic software tools" such as Radar (Apple's bug tracking system) and other hooked on organizational processes (such as bug review board meetings) together carry the software management processes.

Organizational processes help carry the software development process. The bug-tracking systems (the artifacts that are the "basic tools") alone does not help achieve resolutions of bugs. Our study indicates that it is the various bug review meetings, issue meetings, informal discussions (at lunch and in the hall ways) and the use of voice-mail and email, that helps bring resolution to problems.

There is no holy grail. A sophisticated client-server architecture of a bug-tracking work-flow process alone is not sufficient to make bug resolution successful. A good software system does not guarantee that the users of the systems will not need support from other artifacts. Designers of software systems need to identify other artifacts which support work processes.

Apple bug management artifacts. Our study of Apple's bug-management indicates that colorful and rich set of colloquialisms used throughout the bug-management process are phrases such as: Hit-squads, Bug-Meisters, Hot-potatoes, Bug-Review Boards, Beta-blockers, and Quality War room. These buzz words describe bug-management artifacts (or tools) that crucially support workflow.

CONCLUSION

The next phase of our work will be to use the artifact taxonomy in designing a workflow process for requirements engineering at other software development companies. We will study the implementation of processes using Endeavors (2), a workflow process tool developed at University of California, Irvine. We envision the use of the artifact taxonomy as a tool to aid design and implementation of workflow processes such as document routing and approval processing.

ACKNOWLEDGMENTS

Some supported provided by the University of California, Irvine. Effort also sponsored by the Defense Advanced Research Projects Agency, and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-2-0021. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Air Force Research Laboratory or the U.S. Government.

REFERENCES

1. Beyer, H. and Holtzblatt K. *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufmann Publishers, Inc., San Francisco, Ca., 1997.
2. Bolcer, G. and Taylor, R. *Endeavors: A Process System Integration Infrastructure*, In The 4th International Conference on Software Process (CSP4); Brighton, U.K., 1996.
3. Engeström, Y. (1990). *Learning, working and imagining: Twelve studies in activity theory*. Helsinki: Orienta-Konsultit.
4. Miettinen, R.(1997). *Object Construction and Networks in Research Work: The case of research on cellulose degrading enzymes.* Social Studies of Science forthcoming.
5. Shukla, S., Nardi, B., Redmiles, D.(1997). *Beta Blockers and Hot Potatoes: Redefining Artifacts in Sociotechnical Design*, Technical Report UCI-ICS-97-45 Department of Information and Computer Science, University of California, Irvine, CA, September 1997.