Empirical Studies in End-User Software Engineering and Viewing Scientific Programmers as End-Users
-- POSITION STATEMENT --

Jeffrey Carver
Mississippi State University
carver@cse.msstate.edu

Abstract

My work has two relationships with End User Software Engineering. First, as an Empirical Software Engineer, I am interested in meeting with people who do research into techniques for improving end-user software engineering. All of these techniques need to have some type of empirical validation. In many cases this validation is performed by the researcher, but in other cases it is not. Regardless, an independent validation of a new approach is vital. Second, an area where I have done a fair amount of work is in software engineering for scientific software (typically written for a parallel supercomputer). These programmers are typically scientists who have little or no training in formal software engineering. Yet, to accomplish their work, they often write very complex simulation and computation software. I believe these programmers are a unique class of End-Users that must be addressed

1. Introduction

In this position paper, I will address work in two main areas related to End-User Software Engineering. The first area, discussed in Section 2, is related to the need for and use of empirical studies in End-User Software Engineering. This section provides the motivation for performing empirical studies, an overview of the types of studies that can be useful, and an example from my own experience.

The second area, discussed in Section 3, is related to a class of users who are not always considered in the discussion of End-User Software Engineering, the scientists and engineers. I argue that these users are not professional programmers, but rather they are a special class of End-Users that deserve unique attention and research.

2. Empirical Studies

The use of empirical studies is necessary in End-User Software Engineering for the same reasons that it is necessary in more traditional software engineering. An empirical study provides a researcher with the hard data necessary to make informed decisions, rather they relying only on hype or argumentation. Different types of empirical studies provide different types of evidence. Choosing the appropriate study and the appropriate evidence is important based on the goal of the research inquiry.

There are two main types of empirical studies that can be of use in this domain. Studies that are more exploratory and studies that are more confirmatory. In an exploratory study, the goal of the researcher is to understand the environment. This understanding could provide insight into identification of requirements for a new tool or interface or identification of necessary improvements in an existing interface. By gathering information about how the target users perform the task, the researcher can better understand the type of interface or tool that will best serve them. In addition, by observing users who are working with an existing interface or tool, researchers can understand how that tool or interface can be improved.

Software Engineering researchers have been doing these types of studies for a long time. Our studies focus on professional developers rather than end-users. And, our goals are typically to better understand or improve particular aspects of the software engineering process. But, the approaches used in study design and data analysis are similar to what is needed in the end-user domain [3, 6].

One important aspect of empirical studies that I believe I can offer to members of the EUSE community is independence and objectivity. One benefit of being independent, that is, not developing the end-user technologies myself, is that I have no

Dagstuhl Seminar Proceedings 07081
End-User Software Engineering
http://drops.dagstuhl.de/opus/volltexte/2007/1079

vested interest in the outcome. One danger of a researcher performing empirical validation on his or her own tools or interfaces is that positive results are viewed with some skepticism. Studies conducted by an objective third party will lend additional validity to the results.

My experience in this domain comes from performing a series of experiments on the WYSIWYT prototype in Excel [7, 8]. In our study, we were interested in evaluating the use of WYSIWYT within the Excel environment to understand how the results from the Forms/3 environment translated. The goal of the study was to determine whether people would create a more correct, more tested spreadsheet when using WYSIWYT than they would when using the normal facilities provided by Excel [1].

In this study, the subjects were given the task of creating a spreadsheet based on a provided specification. They were instructed that their goal was to make the spreadsheet as correct as possible. The subjects were students in the Business Technology department at Mississippi State University who were taking a course on Spreadsheets. Therefore, they were representative of novice spreadsheet users, which is an interesting population for this study. The results of the study indicated that, while the WYSIWYT add-in did not improve overall correctness, it did decrease the amount of time required to reach the same level of correctness.

3. Scientists and Engineers as End-Users

High performance computing systems are used to develop software in a wide variety of domains including nuclear physics, crash simulation, satellite data processing, fluid dynamics, climate modeling, bioinformatics, and financial modeling. The TOP500 website (http://www.top500.org) lists the top 500 high performance computing systems. The diversity of government, scientific, and commercial organizations present on this list illustrates the growing prevalence and impact of HPC applications on modern society. These software systems are largely developed by experts in the scientific or engineering domain that is being modeled. Therefore, they have little or no training in formal software engineering.

This class of developers should be considered as a special type of end-users for the following reasons. First, they lack training in formal software engineering and often lack the interest in following correct software engineering principles. Second, for these developers, the production of software is a secondary goal. Their main interest is the science or engineering. To accomplish their goal, they must often write simulation code or computation code. While this code may often be shared and used by others, it is not the end goal of their work [2, 4, 5].

4. References

[1] Carver, J., Fisher II, M., and Rothermel, G. "An Empirical Evaluation of a Testing and Debugging Methodology for Excel". In *Proceedings of 2006 International Symposium on Empirical Software Engineering*. Rio de Janeiro. Sept. 21-22, 2006, 2006. p. 278-287

[2] Carver, J., Hochstein, L., Kendall, R.P., Nakamura, T., Zelkowitz, M.V., Basili, V.R., and Post, D., "Observations about Software Development for High End Computing." *CTWatch*, 2006. **November**: 33-37.

[3] Carver, J., Shull, F., and Basili, V.R., "Can Observational Techniques Help Novices Overcome the Software Inspection Learning Curve? An Empirical Investigation." *Empirical Software Engineering: An International Journal*, 2006. **11**(4): 523-539.

[4] Carver, J., Kendall, R.P., Squires, S., and Post, D. "Software Development Environments for Scientific and Engineering Software: A Series of Case Studies". In *Proceedings of 2007 International Conference on Software Engineering*. Minneapolis. 2007. p.

[5] Hochstein, L., Nakamura, T., Basili, V.R., Asgari, S., Zelkowitz, M.V., Hollingsworth, J.K., Shull, F., Carver, J., Voelp, M., Zazworka, N., and Johnson, P., "Experiments to Understand HPC Time to Development." *CTWatch*, 2006. **November**: 24-32.

[6] Maldonado, J., Carver, J., Shull, F., Fabbri, S., Doria, E., Martimiano, L., Mendonca, M., and Basili, V., "Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness." *Empirical Software Engineering*, 2006. **11**(1): 119-142.

[7] Rothermel, G., Li, L., and Burnett, M. "Testing Strategies for Form-Based Visual Programs". In *Proceedings of 8th International Symposium on Software Reliability Engineering*. Albuquerque, NM USA: IEEE-CS. Nov., 1997. p. 96-107

[8] Rothermel, G., Burnett, M.M., Li, L., DuPuis, C., and Sheretov, A., "A Methodology for Testing Spreadsheets." *ACM Transactions on Software Engineering and Methodology*, 2001. **10**(1): 110-147.