Extended Abstract:

The Next Step: From End-User Programming to End-User Software Engineering

Margaret Burnett
Elec. Engr. & Computer Science
Oregon State University
Corvallis, OR 97331 USA
burnett@eecs.oregonstate.edu

Mary Beth Rosson
Information Sciences & Technology
Pennsylvania State University
University State College, PA 16802
mrosson@psu.edu

Brad Myers
Human-Computer Interaction Inst.
Carnegie Mellon University
Pittsburgh, PA 15213 USA
bam@cs.cmu.edu

Susan Wiedenbeck
Information Science & Technology
Drexel University
Philadelphia, PA 19104 USA
Susan.Wiedenbeck@cis.drexel.edu

CHI 2006, April 22–27, 2006, Montreal, Canada.
ACM 1-xxxxxxxxxxxxxxxxxx.

Abstract

Is it possible to bring the benefits of rigorous software engineering methodologies to end users? End users create software when they use spreadsheet systems, web authoring tools and graphical languages, when they write educational simulations, spreadsheets, and dynamic e-business web applications. Unfortunately, however, errors are pervasive in end-user software, and the resulting impact is sometimes enormous. A growing number of researchers and developers are working on ways to make the software created by end users more reliable. This workshop brings together researchers who are addressing this topic with industry representatives who are deploying end-user programming applications, to facilitate sharing of real-world problems and solutions.

Keywords

End-User Software Engineering, Testing, Empirical Studies of Programming, Psychology of Programming, Programming by Demonstration.

ACM Classification Keywords

D.2.5 Testing and Debugging; H.1.2 User/Machine Systems—*Software psychology.*

Introduction

There has been considerable work in empowering end users to be able to write their own programs, and as a result, users are indeed doing so. The "programming" systems used by these end users include spreadsheet systems, web authoring tools, and graphical languages for demonstrating the desired behavior of educational simulations. Using such systems, end users create software, in forms such as educational simulations, spreadsheets, and dynamic e-business web applications.

Unfortunately, however, errors are pervasive in this software, and the resulting impact is sometimes enormous. When the software is not dependable, there can be serious consequences for the people whose retirement funds, credit histories, e-business revenues, and even health and safety rely on decisions made based on that software. Such problems are ubiquitous in spreadsheets [6], open resource coalitions [7] and dynamic web applications [8]. Two recent NSF workshops have determined that end-user software is in need of serious attention [1].

Researchers have begun to join together into a subarea of "end-user software engineering," to develop and investigate technologies aimed at this problem. We have already demonstrated some interesting progress in tools and techniques in this area.

Special interest group (SIG) meetings at CHI'04 and CHI'05 have successfully brought together these researchers with many others in the CHI community who are concerned about the user interfaces and reliability of software and software tools. At CHI'06, this second **W**orkshop on **E**nd-**U**ser **S**oftware **E**ngineering (WEUSE II) builds upon the interest expressed by these participants and those who attended WEUSE I at the ICSE'05 conference. We plan to organize follow-up events (WEUSE III, ...) at future CHI, ICSE, and related venues as well.

Example Technologies

There is a tremendous range of technologies that can be brought to bear on this problem. This section highlights a number that are being developed by the organizers and their collaborators in the EUSES Consortium[1], and we expect to find out about others at the workshop.

Traditional methods and tools for addressing software development and dependability problems for professional programmers are usually not suitable for end-user programmers. Rather, we envision systems that create software in collaboration with those users, in a software development paradigm that combines traditionally separate functions – blending specification, design, implementation, component integration, debugging, testing, and maintenance into tightly integrated, highly interactive environments. These environments employ new, incremental, feedback devices supported by analysis and inferential reasoning to help the user reason about the dependability of their software as they work with it, in a manner that respects the user's problem-solving directions to an

[1] The EUSES Consortium (End Users Shaping Effective Software) consists of researchers from Oregon State University, Carnegie Mellon University, Drexel University, Pennsylvania State University, University of Nebraska, and Cambridge University. See http://eusesconsortium.org.

extent unprecedented in existing software development environments.

The *End-User Software Engineering* project at Oregon State University aims to improve the reliability of software produced by end-user programmers in general, and by spreadsheet users in particular. Some results have included "What You See Is What You Test" (WYSIWYT) integrated with fault localization and with assertions for end-user programmers [2], and semi-automated detection of erroneous combinations of units in spreadsheets [3]. A recent emphasis has been on how to interest users in end-user software engineering devices without detrimentally interrupting their problem-solving efforts [9].

The *Natural Programming Project* at Carnegie Mellon University is investigating a variety of techniques around the idea of applying computer-human interaction principles to the design of programming languages and environments. In 2004, we reported on the "WhyLine," a debugging tool that helped end users find bugs in 1/8 the time, and increased programmer productivity by about 40% [5]. Current work is looking at more effective tools for supporting the editing and construction of code [4] and for users' investigations of new SDKs.

Penn State researchers in the *Informal Learning in Software Construction* project are studying real world situations and communities that can motivate and aid non-programmers in learning and using end-user programming tools. Prior work characterized the problems of public school teachers learning to build visual simulations, and designed minimalist training materials and reusable code to serve as scaffolding [11]. Recent research is studying the mental models of web software construction held by sophisticated end users, and is using these results to develop a tool for building simple web applications [10].

Researchers at Drexel University are studying cognitive and social factors that may affect end users' acceptance of end-user programming tools and their effectiveness in using them. Research on school teachers has investigated strategies that teachers use in programming [13] and has identified facilitators and inhibitors to end-user programming in the school setting [12]. Current research in collaboration with researchers at Oregon State University is focusing on the effect of culture and gender on success in end-user programming.

Researchers in end-user software engineering are working on a variety of other approaches as well. Among them are new surveys of end-user programmers in real organizations, fault detection through statistical methods and through program analysis, pedagogical methods to encourage a quality-control culture for users of technology, and motivational and attention allocation issues for end-user programmers.

Workshop Goals

The workshop's goals are: (1) to generally share information and raise awareness among researchers already in this area with researchers in the related areas of Empirical Studies of Programming and Psychology of Programming, and with practitioners interested in current and future techniques that can be embodied in tools and development processes; and (2) to concretely match end-user software engineering

problems in industry with potential solutions drawn from new and emerging research findings. One outcome of the first goal, in addition to shared knowledge, will be the groundwork for a new collaborative effort, involving interested attendees at the workshop, for a survey paper on the state of end-user software engineering research. At the CHI'04 and CHI'05 SIGs and ICSE'05 workshop, initial categorizations of existing research and the problem space began to emerge, and these will form as a starting point for this workshop.

We hope to also make one or more matches resulting in future collaborations that apply research findings to problems that industrial participants would like to solve.

References

[1] Boehm, B. and Basili, V., "Gaining Intellectual Control of Software Development." Computer, 2000. 33(5): pp. 27-33.

[2] Burnett, M., Cook, C., and Rothermel, G., "End-User Software Engineering," Communications of the ACM, 2004. 47(9): pp. 53-58..

[3] Erwig, M. and Burnett, M. "Adding Apples and Oranges," Fourth International Symposium on Practical Aspects of Declarative Languages. 2002.

[4] Ko, A.J., Aung, H.H., and Myers, B.A. "Design Requirements for More Flexible Structured Editors from a Study of Programmers' Text Editing," Extended Abstracts CHI'2005: Human Factors in Computing Systems. Portland, OR, April 2-7, 2005. pp. 1557-1560.

[5] Ko, A.J. and Myers, B.A. "Designing the Whyline, a Debugging Interface for Asking Why and Why Not Questions About Runtime Failures," CHI'2004: Human Factors in Computing Systems. 2004. Vienna, Austria: pp. 151-158.

[6] Panko, R., "Finding Spreadsheet Errors: Most Spreadsheet Models Have Design Flaws That May Lead to Long-Term Miscalculation." Information Week, 1995. p. 100.

[7] Raz, O. and Shaw, M. "An Approach to Preserving Sufficient Correctness in Open Resource Coalitions," 10th International Workshop on Software Specification and Design. 2000.

[8] Ricca, F. and Tonella, P. "Analysis and Testing of Web Applications," International Conference on Software Engineering. 2001. pp. 25-34.

[9] Robertson, T., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, J., Beckwith, L., and Phalgune, A. "Impact of Interruption Style on End-User Debugging," CHI 2004: Human Factors in Computing Systems. 2004. Vienna, Austria: pp. 287-294.

[10] Rode, J. and Rosson, M.B. "Programming at Runtime: Requirements and Paradigms for Nonprogrammer Web Application Development," IEEE Symposium on Human-Centric Computing Languages and Environments. 2003.

[11] Rosson, M.B. and Seals, C. "Teachers as Simulation Programmers: Minimalist Learning and Reuse," CHI'2001: Human Factors in Computing Systems. 2001. Seattle, WA: pp. 237-244. .

[12] Wiedenbeck, S. "Facilitators and inhibitors of end-user development by teachers in a school environment." *IEEE Symposia on Visual Languages and Human-Centric Computing*, 2005, pp. 215-222.

[13] Wiedenbeck, S. and Engebretson, A. "Comprehension strategies of end-user programmers in an event driven application." *IEEE Symposia on Visual Languages and Human-Centric Computing*, 2004, pp. 207-214.