Empirical Software Engineering, 4, 5–10 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Introduction to Special Issue on Usability Engineering

JEAN SCHOLTZ

BEN SHNEIDERMAN

This Special Issue on Usability Engineering is meant to build a bridge between the software engineering (SE) and the human-computer interaction (HCI) research communities. Usability engineering is emerging as the commercial and practical discipline that incorporates participatory design, rapid prototyping, and iterative evaluation.

User-centered design and usability evaluations have become common practices in many organizations, but in the majority of software development shops they are still novel and not routinely practiced. Typical software engineering development cycles do not accommodate these practices. The Software Engineering Institute which influences government and commercial practice and education rarely mentions usability or user interface design methods.

So the question remains: How can usability engineering practices become part of the mainstream of software engineering? We believe that widespread inclusion of usability engineering practices in development will be fostered by empirical studies validating these practices.

Call to Action

We hope that this special issues reaches researchers in the SE and HCI domains and that they see the rich potential in applying empirical studies to usability engineering. The user interface is the end user's view of the software application. Functionality can be hidden or obscured by a less than optimal user interface. Robust, error-free code is useless if users do not know how to proceed with steps in an application.

The current user interfaces are improving, but users' expectations and dependency are growing still faster. Many observers, including us, feel that it is time to get angry about the quality of user interfaces. It seems likely that more time is wasted in front of computer screens than on highways. We can do better!

We should expect more from the developers of user interfaces. They should be responsible if the software crashes or damages data. They should also be responsible if users get confused by inconsistent terminology, chaotic screen designs, complex sequences of actions, or indecipherable error messages. The high safety record of airlines, excellence in manufacturing quality control, or telephone reliability are examples that usability engineers can follow. We should expect user interfaces that support users with short learning times for novices, rapid performance on benchmark tasks, low error rates for frequent users, and high retention over time for intermittent users. These qualities can lead to high levels of user satisfaction that encourage users to apply user interfaces for life-critical and high volume commercial tasks.

As electronic commerce becomes the common way of conducting business transactions, users must trust the software applications and search engines. As software becomes common in life-critical arenas such as air traffic control and medical applications, it is essential that tasks at hand, not the software applications, are the focus of the end user. Confusion, hesitation, and misinterpretation at the user interface can and have caused disasters.

It is time that software engineering practices and usability engineering practices merged. We hope that the empirical studies of usability engineering presented in this issue will serve as vehicles for software engineering practitioners and researchers to transport these practices into their environments.

An Historical Review

Our belief in the benefits of empirical testing in software engineering has a long history. The early and influential work of Jerry Weinberg (1971) led many people to get past the myths and wishful thinking. He advocated scientific approaches that encouraged careful observation and evaluation of software development methods, team organizations, and programming style.

NASA's Software Engineering Laboratory (founded in 1975) sought to promote advanced practices and rigorous evaluations (Basili and Reiter, 1979; Basili, Selby and Hutchens, 1986). By 1980 there was enough research to fill a book and the title "Software Psychology" (Shneiderman, 1980) caught on. In 1986 a group of devotees launched a conference on Empirical Studies of Programming, that produced six volumes of proceedings.

E. Soloway and S. Iyengar, (Editors), 1986. Empirical Studies of Programming, Ablex (available from Intellect Books).

Second Workshop, G. Olsen, S. Sheppard, E. Soloway (Editors), 1987. Empirical Studies of Programming, Ablex (available from Intellect Books).

No published proceedings from the third workshop.

J. Koenemann-Belliveau, T. Moher, S. Robertson, (Editors), 1991. Empirical Studies of Programmers: Fourth Workshop, Ablex (available from Intellect Books).

C. Cook, J. Scholtz, J. Spohrer, (Editors), 1993. Empirical Studies of Programmers: Fifth Workshop, Ablex (available from Intellect Books).

W. Gray, D. Boehm-Davis, (Editors), 1996. Empirical Studies of Programmers: Sixth Workshop, Ablex (available from Intellect Books).

S. Wiedenbeck, J. Scholtz, (Editors), 1997. Empirical Studies of Programmers: Seventh Workshop, ACM Press, New York.

The software engineering community expanded rapidly during the 1970s and 1980s with its lively annual International Conference on Software Engineering (starting in 1975) and the highly respected IEEE Transactions on Software Engineering (starting in 1975). While most writers celebrated empirical studies, too few of them actually carried them out (Zelkowitz and Wallace, 1998; Tichy, 1998). The emergence of this journal, Empirical Software Engineering in 1996, signaled increased interest in scientific evaluations of software engineering practices.

In the same period, human-computer interaction (HCI) emerged as a separate discipline. Invigorated by a highly successful 1982 conference, the ACM Special Interest Group on Computer Human Interaction (SIGCHI) became the fastest growing of ACM's 35 SIGs. Its annual conference grew to almost 2500 people and it spawned a half dozen related conferences. More than a dozen journals cover the topic of user interface design.

Unfortunately, the gulf between SE and HCI seemed to widen as each became more successful. Preece and Rombach (1994) tried to bring the two disciplines closer by showing the parallels. This special issue is intended to further this reconciliation.

The term "usability engineering" is meant to address the practical strategies for designing, implementing, testing, and reviewing user interface designs. In the commercial world, there are attempts to make software usability more visible. Magazines that rate software products often include ratings for the applications' usability. Many software vendors have usability testing laboratories where products are tested by users during development.

In 1998 the National Institute of Standards and Technology held two workshops attended by representatives from large software development companies and representatives from large software purchasing organizations. The goal of these workshops was to make usability data visible to software purchasers. Usability problems cause users to be less productive; corporate information technology departments must support these users and these efforts add to the total cost of ownership of software. Informed customers can make better estimates about the cost to their organization. Software suppliers hope that visibility will encourage purchasing organizations to work closely with suppliers; suppliers who understand the needs of their customers will be able to deliver more usable software. Efforts such as this will necessitate changes in software development and require an integration of software and usability engineering.

Generating this Special Issue

We sought papers for this special issue by advertising widely for research results on empirical testing, field studies, or thorough case studies. We suggested topics that we thought would be useful developments in usability engineering, such as:

Design

- Frameworks and methodologies for user interface design and development
- Incorporation of usability engineering into software engineering life cycles
- Novel methods for obtaining user requirements
- Cost-benefit analysis of usability engineering methods
- Use of existing data from other domains (sociological, demographic, market analysis, cognitive and social psychology) in product design and evaluation
- Use of field research methods in product design
- International and cross-cultural software engineering

Testing and Reviews

- Evaluation of strategies for expert reviews and usability testing
- Utility of preference vs. performance measures
- Validation of surveys and metrics
- Web-based remote usability testing

Our call for papers produced an interesting set of submissions and our reviewers provided thoughtful and detailed feedback. Three papers were selected and the authors were given detailed comments to guide their revisions. The interaction with authors and reviewers generated lively discussions and we thank our reviewers:

Robin Jeffries
James Herbslab
William Hefley
Deborah Hix
Scott Isensee
Laura Leventhal
Michael Muller
Shari Lawrence Pfleeger
Barbee Teasley
Dolores Wallace

The three papers we chose provide useful empirical evaluations of the issues they raise:

The paper by Paterno and Mancini develops a systematic method for designing hypermedia. Promoters of software engineering methodologies should be pleased to see how their concepts have been extended to cover hypermedia design. The authors apply a metrics based evaluation approach and empirical testing to refine their design.

The paper by Zhang, Basili, and Shneiderman provides empirical validation that the principles of perspective-based reviews that were successful in reviewing software are also effective in reviewing user interfaces.

The paper by Keenan, Hartson, Kafura and Schulman offers a Usability Problem Taxonomy (UPT). The 28 categories emerged by analyzing more than 400 usability problems. The UPT is likely to be useful as a guide to reviewers and testers of user interfaces, just as software engineering problem taxonomies have been useful in finding flaws in software.

These three papers represent usability engineering practices in design, methods for usability reviews, and a methodology for classifying usability problems. We hope that from this broad range of topics, ESE readers will be able to see how usability practices could be incorporated into their area of software engineering.

References

Basili, V. R., and Reiter, R. W., Jr. 1979. An investigation of human factors in software development. *IEEE Computer* 12(12): 21–38.

Basili, V. R., Selby, R. W., and Hutchens, D. H. 1986. Experimentation in software engineering. *IEEE Transactions on Software Engineering* SE-12(7): 733–743.

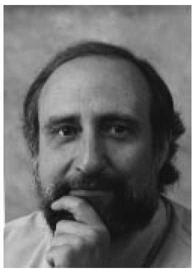
Preece, J. J., and Rombach, D. 1994. A taxonomy for combining Software Engineering (SE) & Human-Computer Interaction (HCI) measurement approaches: Towards a common framework. *The International Journal of Man-Machine Studies* 14: 553–583.

Shneiderman, B. 1980. *Software Psychology: Human Factors in Computer and Information Systems*. Boston: Little, Brown and Co., (formerly Winthrop Publ.).

Tichy, W. 1971. Should computer scientists experiment more? *IEEE Computer* 31(5): 32–40.

Weinberg, G. M. 1971. *The Psychology of Computer Programming*. New York: Van Nostrand Reinhold.

Zelkowitz, M., and Wallace, D. 1998. Experimental models for validating technology. *IEEE Computer* 31(5): 23–31.



Dr. Ben Shneiderman is a Professor in the Department of Computer Science, Head of the Human-Computer Interaction Laboratory, and Member of the Institutes for Advanced Computer Studies and for Systems Research, all at the University of Maryland at College Park. He is the author of *Software Psychology: Human Factors in Computer and Information Systems* (1980) and *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (1987, second edition 1992, third edition 1998), Addison-Wesley Publishers, Reading, MA. He is on the Board of Directors of Spotfire Inc. and has been on the Editorial Advisory Boards of nine journals. He has consulted and lectured for many organizations including Apple, AT&T, Citicorp, GE, Honeywell, IBM, Intel, Library of Congress, Microsoft, NASA, and university research groups. He is an ACM fellow.

photo
not
available
at time of
print

Dr. Jean Scholtz is currently a program manager in the Information Technology Office at the Defense Advanced Research Projects Agency (DARPA). She is on loan from the National Institute of Standards and Technology where her research is in the evaluation of interactive systems. Prior to moving to the east coast, Dr. Scholtz worked as a human factors engineer for Intel Corporation. After receiving her Ph.D. in Computer Science, Dr. Scholtz was a faculty member at Portland State University where she started a masters track in Human-Computer Interaction. Dr. Scholtz is on the executive committee of the SIGCHI organization, the ACM special interest group on Computer-Human Interaction. She is also on the Editorial Advisory Board for several journals.