Supporting Reflective Practitioners

David Redmiles
Department of Informatics
School of Information and
Computer Science
University of California, Irvine
Irvine, CA 92697-3425 USA
redmiles@ics.uci.edu

Kumiyo Nakakoji
RCAST
University of Tokyo
and PRESTO, JST
4-6-1 Komaba, Meguro, Tokyo,
153-8904 Japan
kumiyo@kid.rcast.u-tokyo.ac.jp

Abstract

The theme and title for this panel is inspired by Donald Schön's writings about the reflective practitioner in which he describes professional practice as being a process of reflection in action. Ill-defined problems including design decisions lead to breakdowns, which become opportunities for reflection and modification of practice. This panel seeks to provide ICSE attendees with a broad cross section of the history, state of the art, and open issues with some of the methods and tools directed at supporting reflective software practitioners.

1. Panel Theme at a Glance – Supporting Reflective Practitioners

The theme and title for this panel is inspired by Donald Schön's writings in which he described professional workers as reflective practitioners [1]. He observed how professionals frequently work on ill-defined problems—problems that defy rote solutions. Design decisions fall into this category. Ill-defined problems lead to breakdowns, which become opportunities for reflection, modification, and improvement of practice. Schön described this particular kind of reflection as an activity that transcends technical rationality, a process that he named *reflection in action*. It was a different kind of thinking than "simple logic." This panel seeks to unriddle these concepts and ground them in the context of problems that software developers often face and problems that end users of software face.

2. Further Background Related to Design and Software Development

Authors such as Brooks, Simon, Suchman, and Winograd have articulated concepts and concerns similar to Schön's puzzle about the design activity that people engage in when grappling with ill-defined problems. For instance, Fred Brooks distinguished between accidental and essential complexity for designers of software systems [2]. Software development tools could support mundane aspects of designers' work, but the most creative aspects would still elude computer support. Herbert Simon also referred to the bounds of rationality and evoked the anecdote of the painter faced with a blank canvas to describe ill-defined problems, which required a different kind of thinking [3]. In her seminal characterization of situated action, Lucy Suchman demonstrated the limits of rationalized designs [4]. As she notes, anticipating all potential user behaviors is not a feasible approach to design. Winograd and Flores turned to models of conversation and theories of discourse analysis to explain design as an interpretive activity [5] [6].

In light of these observations, the panel will consider software development from the perspective of a human activity [7]. Reflective practitioners are needed in software design not so much for the accidental but the essential, to use Brooks' terms. Observing another aspect of the puzzle of design, Greenbaum and Kyng observed that "system development is difficult not because of the complexity of technical problems, but because of the social interaction between users and system developers as they learn to create, develop, and express their ideas and visions" [8]. Software engineering (especially its upstream activities) is a human-oriented field, and as such will always have the openness of other design disciplines, such as architecture and graphic design, rather than the hard-edged formulaic certainty of downstream engineering.

Equal to the problem of software developers being reflective practitioners is the problem of end users acting reflectively. Namely, there is a large class of software systems in which end users act as designers as they perform both professional and personal tasks. In these systems, end users, who become designers themselves,

act, reflect, and react because they do not have a definitive solution or clear-cut plan of action before they start interacting with a system. This growing class of systems raises important questions for ICSE community. Is this new class of systems different from old systems? Should there be a new foundation for this class of systems?

The challenge of designing computer support for reflective practitioners has been approached sporadically by the software community, often in an intuitive rather than intentional fashion. It needs a more cohesive and focused response by the software community. To date, some of the following work is related. Software critics are an interface technique intended to trigger reflection by end users, providing feedback on design tasks while designers are still in the context of making design decisions [9]. Critics are not intended to replace human decision making, but to complement it [10][11]. Similarly, software agents proactively coordinate in order to assist end users, including in some instances software designers as end users [12]. Techniques for supporting software process descriptions have evolved from rigid prescriptive systems to reflective models that can adapt to exceptions [13]. Even the open source movement might be interpreted as a style of software development that supports reflective practice by opening up the evolution of a software system to public criticism and improvement with many improvements being directly motivated by breakdowns (in Schön's sense) that the users experience [14].

Our target goal for this panel is to stimulate members of the ICSE community to think more deeply and comprehensively about software development as a human design activity especially in terms of the characteristics of design's reflective and situated nature. We also seek to provoke the audience to think explicitly about developing software for end users who are themselves reflective practitioners, along with the theoretical and practical implications of that kind of software. We believe that audience members will 1) learn about some new method or tool that they had not previously been familiar with, and 2) take away an understanding or how some problems of developing software systems can be framed in light of end users needing support for reflective action. We see the set of problems comprising this endeavor as the same comprising the broader topic of a science of design [3].

3. Biographies of the Organizers and Panelists

David Redmiles is an Associate Professor in the Department of Informatics at the University of California, Irvine. Most recently, his research has examined knowledge-based support for object-oriented design; automated, agent-based support for collecting usage data; tools for supporting awareness; event-based distributed architectures; and field studies of collaborative software engineering. He received his PhD from the University of Colorado, Boulder, and previously worked at the National Institute of Standards and Technology (NIST). He served as a program co-chair for the 1998 IEEE Conference on Automated Software Engineering and will be general chair of that conference in 2005. He has organized workshops on Collaborative Software Engineering and Automated Software Engineering. He is a joint editor on a 2002 special issue of the Journal of Computer Supported Cooperative Work on Activity Theory and Design.

Kumiyo Nakakoji is a Full Professor at the Research Center for Advanced Science and Technology, University of Tokyo, Japan. Her current research interests include the knowledge interaction design framework for the development of interactive systems for creative knowledge work and for supporting collective creativity. She is active in the ICSE community and is a program committee member. She has served as Workshop Co-chair at CHI 2003, Tutorial Co-chair at CHI 2002, Associate Paper Chair at CHI 1997 and CHI 1994, and CHI Asia-Pacific Regional Liaison for a number of years. She has served as editor-in-chief for the Information Processing Society of Japan, guest editor for International Journal of Human-Computer Studies, Knowledge Based Systems Journal, and program committee member for a number of conferences. She co-organized three previous CHI workshops: the 2001 Workshop on Tools, Conceptual Frameworks, and Empirical Studies for Early Stages of Design, the 1995 Workshop on Knowledge-based Support for User Interface Design, and the 1993 Workshop on Cross-Cultural Issues on Human-Computer Interaction.

Gerhard Fischer is a Full Professor of Computer Science, a fellow of the Institute of Cognitive Science, and director of the Center for LifeLong Learning & Design (L3D) at the University of Colorado at Boulder. His current research interests include creativity and meta-design, computer-assisted technologies for people with cognitive disabilities, and the integration of computational and physical artifacts for collaboration and learning. He is a pioneer of the concept of software critics in design and has authored numerous publications on domain-oriented design environments and lifelong learning and design. He is on the editorial board of several journals in areas of learning, software engineering, and human-computer interaction. Recently, he chaired the 2002 Computer-Supported Collaborative Learning Conference held in Boulder, Colorado.

Yunwen Ye is a Chief Researcher at SRA Key Technology Laboratories, Tokyo, Japan. His current research interests include developer-centric software development environments, software reuse, evolution of open source software systems and communities, social and cognitive aspects of software development, and

knowledge management issues in software development communities and organizations. He has published many journal and conference papers in the above research areas. Recently, he co-organized the International Symposium on Social Creativity in 2002, and served as a panelist in the State-of-the-Art in Open Source Forum, held jointly by Software Engineer's Association (Japan) and Free Software Initiative Japan.

Alistair Sutcliffe is a Full Professor of Systems Engineering, and Director of the Centre for HCI Design, in the School of Informatics, University of Manchester, UK. His research spans software engineering, human computer interaction and cognitive science, with current interests in scenario based design, methods for requirements engineering, and creative design for the Internet. He is a leading authority on multimedia user interfaces, has authored 6 books and over 200 publications on user interface design, requirements engineering, software and domain knowledge reuse. He serves on the editorial boards of several journals in the software engineering and human computer interaction, and recently co-chaired the ACM conference Designing Interactive Systems 2002.

Sol Greenspan is a Program Director in the Foundations of Computing Processes and Artifacts Cluster at the National Science Foundation. Previously, he worked at GTE Laboratories, consulted in the health care industry, and taught graduate courses in both London (UK) and Massachusetts. He recently became a Research Affiliate at MIT in the Software Design Group. He has served on the editorial boards of the Automated Software Engineering Journal and the Requirements Engineering Journal, and IEEE Communications Magazine. He has served on the Steering Committee for the International Workshop on Software Specification and Design, chaired IFIP Working Group 2.9 (Software Requirements Engineering) from 1998-2002, and program committees for many conferences. He received his Ph.D. at the University of Toronto in 1984 on work using concepts of AI knowledge representation in the area of requirements analysis. This work led to a paper, "Capturing More World Knowledge in the Requirements Specification," which later received the award for most influential paper from ICSE-6.

6. References

[1] Schön, D. The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York, 1983.

[2] Brooks, F. No Silver Bullet: Essence and Accident in Software Engineering. *IEEE Computer,* 20(4), 10-19, 1987.

[3] Simon, H. The Sciences of the Artificial, The MIT Press, Cambridge, MA, 1996.

[4] Suchman, L. Plans and Situated Actions, Cambridge University Press, Cambridge, UK, 1987.

[5] Winograd, T. and Flores, F. Understanding Computers and Cognition: A New Foundation for Design. Ablex Publishing Corporation, Norwood, NJ, 1986.

[6] Winograd, T., Ed. Bringing Design to Software. ACM Press and Addison-Wesley, New York, NY, 1996.

[7] Fischer, G. Desert Island: Software Engineering — A Human Activity, *International Journal Automated Software Engineering,* Kluwer Academic Publishers, Dordrecht, Netherlands, 10(2): 233-237, 2003.

[8] Greenbaum, J. and Kyng, M., Eds. Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1991.

[9] Fischer, G. Domain-Oriented Design Environments, *Automated Software Engineering,* Kluwer Academic Publishers, Boston, MA, 177-203, 1994.

[10] Fischer, G., Nakakoji, K. Beyond the Macho Approach of AI: Empower Human Designers - Do Not Replace Them, *Knowledge-Based Systems Journal, Special Issue of AI in Design,* 5(1), 15-30, 1992.

[11] Terveen, L. An Overview of Human-Computer Collaboration, *Knowledge-Based Systems Journal, Special Issue on Human-Computer Collaboration,* 8(2-3): 67-81, 1995.

[12] Hilbert, D., Redmiles, D. Large-Scale Collection of Usage Data to Inform Design, *Eighth IFIP TC 13 Conference on Human-Computer Interaction* (INTERACT 2001, Tokyo, Japan), 569-576, July 2001.

[13] Nutt, G. The Evolution Towards Flexible Workflow Systems, *Distributed Systems Engineering,* 3(4), 276-294, December 1996.

[14] Raymond, E.S, Young B. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly & Associates. Sebastopol, CA, 2001.