Interdisciplinary Design Research for End-User Software Engineering

Alan Blackwell

Dagstuhl seminar on End User Software Engineering, February 2007

My research style involves constantly drawing comparisons from one field to another – across academic disciplines, and also across application domains. In these terms, End-User Software Engineering is neither an application domain, nor an academic discipline, but a technological attitude or strategy, applicable in many domains, while also profiting from many research methods and theory bases. In this respect, it is an ideal opportunity for the multi-disciplinary enquiry and analogical comparisons on which I habitually base my own research [1].

The phrase "end-user software engineering" itself relies on an analogy, in the sense that software engineering is a professional discipline, whereas the end-users whom we hope to assist are defined precisely by the fact that they are not professionals (at least, not software professionals). Our aim in this research is to identify those techniques within software engineering that might offer most benefit to end-users, potentially including tools for specification, debugging, revision management and so on.

As a teacher of professional software engineering, I often draw on the experience of other professional fields, especially design disciplines such as architecture, typography and performance composition [2]. There are certain recurring themes across these design disciplines that I have found to offer substantial insights to professional software engineering. I believe that these same themes can also be productive sources of innovation, by making new analogies to end-user software engineering. In the remainder of this statement, I reflect on some of these analogies.

Design takes place in a social context, and is a social process. Our studies of end-user configuration and automation of domestic technologies demonstrate the extent to which family relations and gender roles spill over into practices of end-user programming [3].

Design processes involve modeling – simplifying or abstracting some aspects of the problem domain in order to plan and evaluate design decisions. The use of representations to reason about future consequences is fundamental to end-user software engineering. The constraints that representations place on design activities are described by the cognitive dimensions of notations framework [4], and in turn by a great variety of research into visual representations.

Abstract reasoning about the future can be described in terms of the attention investment model [5]. A productive approach to end-user software engineering is to modify users' perception of this investment, whether by Burnett's Surprise-Explain Reward strategy, or by the use of machine learning techniques to infer possible abstractions that might be suggested to the user [6].

Finally, I am interested in the extent to which all designers experience their work as creative. This experience should be available to end-users too, not only creative professionals. In studies of choreographers and musicians, my students and I research and develop new notations and programming languages that offer artistic experiences to their users [7]

Many of these activities extend well beyond the bounds of software engineering, empowering users to control and enhance their computer tools in new ways. This was the same motivation that led to the innovations of the modern graphical user interface [8], and I believe that EUSE research might well transform the general purpose user interfaces of the future.

References and Further Reading

1. Blackwell, A.F. and Good, D.A. (in press). Languages of innovation. To appear in H. Crawford & L. Fellman (Eds.). Artistic Bedfellows: Collaborative History and Discourse. University Press of America.
2. Blackwell, A., Bucciarelli, L, Clarkson, P.J., Earl, C.F., Eckert, C., Knight, T., Macmillan, S., Stacey, M. and Whitney, D. (2005). Comparative study of design - application to engineering design. Presented at International Conference on Engineering Design.
3. Rode, J.A., Toye, E.F. and Blackwell, A.F. (2005). The domestic economy: A broader unit of analysis for end user programming. In proceedings CHI'05 (extended abstracts), pp. 1757-1760
4. Blackwell, A.F. and Green, T.R.G. (2003). Notational systems - the Cognitive Dimensions of Notations framework. In J.M. Carroll (Ed.) HCI Models, Theories and Frameworks: Toward a multidisciplinary science. San Francisco: Morgan Kaufmann, 103-134.
5. Blackwell, A.F. (2002). First steps in programming: A rationale for Attention Investment models. In Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments, pp. 2-10.
6. Blackwell, A.F. (2001). SWYN: A Visual Representation for Regular Expressions. In H. Lieberman (Ed.), Your wish is my command: Giving users the power to instruct their software. Morgan Kauffman , pp. 245-270.
7. Blackwell, A. and Collins, N. (2005). The programming language as a musical instrument. In Proceedings of PPIG 2005, pp. 120-130.
8. Blackwell, A.F. (2006). The reification of metaphor as a design tool. ACM Transactions on Computer-Human Interaction (TOCHI), 13(4), 490-530.