Position paper delivered at the ACM-CHI'04 Workshop: "Identifying Gaps between HCI, Software Engineering and Design, and Boundary Objects to Bridge Them", April 2004.

Boundary Object Context is what Counts

Allen E. Milewski
Monmouth University
Department of Software Engineering
Monmouth University
West Long Branch, New Jersey
amilewsk@monmouth.edu

BACKGROUND

Before recently returning to academics, I have worked in the telecommunications Industry for more than twenty years either as a Human-Computer practitioner or as a Systems/Software Engineer. During this time, all of my positions have involved interdisciplinary teamwork, but in three distinct categories. Each of these categories has involved a different role and different boundary-object favorites. Each has given me insight about different aspects of the gaps that exist between Human-Computer Interaction and software practices.

- As a Usability Engineer in several focused, tight-deadlined, product-oriented software development organizations--- This product-oriented experience involved the pragmatic and commonplace HCI role of doing whatever is necessary to form relationships with software developers to get the product built on schedule. The actual work on these teams involved user-oriented design and evaluation, and some software prototyping and development. Example projects include (i) a data network-management and maintenance system and (ii) a trouble-monitoring system for voice-communications equipment. As part of large, structured organizations, the common boundary objects typically included
    - requirements documents (UI and other),
    - a project glossary, and
    - user-interface-oriented modification request (MRs).
    - In one such organization, software prototyping of the UI was a very successful grassroots addition.
- As part of a Corporate-level Architecture Group chartered with negotiating cross-organization user-interface styles, standards and practices as well as processes for productizing emerging technologies-- This role, being cross-organizational and somewhat distant from product deadlines, permitted a more analytic stance about improving the working relationship between HCI and software organizations. The work in this group involved planning and process implementation (and a great deal of what was termed "electro-political-engineering"). Example projects include: (i) corporate standards and common software tool architecture for network management system user interface modules, and (ii) design of a corporate-wide, C-language software library for developing voice-response applications. Official boundary objects included
    - corporate standards and guidelines.
    - However, there evolved an emphasis on software toolkits as an effective way to promulgate the standards.
- As part of a research organization tasked with inventing revolutionary telecommunications services and seeing them into the marketplace. While the first two roles involved large organizations, the research-team experience involved small-group dynamics in an environment with minimal management direction. This work varied enormously, involving UI design, software development and project management as well as some corporate-internal marketing/sales. Projects include: (i) a Video Email system for thin-clients with minimal storage capabilities , (ii) a Web-based messaging system featuring structured response objects such as meeting announcements and invitations, and (iii) a ubiquitous telephone address-book that displays live presence information about those persons listed in it. Favorite artifacts in this small group setting were:
    - A shared project blackboard (a real blackboard)

- UI sketches
- Partially-working, software prototype

POSITION

Motivated by common need and significantly intersecting tasks, the fields of Human-Computer Interaction and Software Engineering have evolved the effectiveness of their relationship. However, improvements are still needed, as is evidence by Kazman et al's [2] survey reflecting the current situation as Software Engineers and HCI practitioners work together on projects in Industry. In general, this survey has revealed a marked schism between Software Engineers and HCI practitioners. Despite working on overlapping problems, they often differ in their perceptions of collaboration amount, they often feel they work separately, and they sometimes feel non-collaboration to be an acceptable situation.

In a variety of environments, the concept of Boundary Objects [4] has been a rich theoretical vehicle for describing how it is that diverse organizations come to cooperate productively. Similarly, the Boundary Object notion is a useful tool for conceptualizing the HCI-SE working relationship. First, Boundary Objects serve as a common point of reference and as a means of translation between organizations. The emphasis on objective artifacts puts constraints on the collaboration that are needed in order to actually deliver a product. Second, rather than melding disciplines, Boundary Objects describe how a common artifact can be useful to distinctly separate organizations by maintaining different "meanings" to each. Both [5] and [3] have argued that combining Software Engineering and HCI into a single discipline or role is unlikely to be productive. For one thing, experience has shown the utility of having a "user advocate", semi-separated from the schedule and budget demands of the rest of the project. Third, Boundary Objects are flexible enough to be useful under conditions of rapid change. This is critical since both HCI and SE fields place increasing emphasis on iterative and continual redesign, development and testing.

While the Boundary Object is a useful concept to describe how cooperation occurs, one has to be cautious when relying on Boundary Objects as anything more than descriptive. There is a stark difference between saying that "organizations use boundary objects to cooperate" and saying, "lets use Boundary Objects to solve the schism between HCI and SE organizations". The prescriptive version has two problems, and they both point to the context of Boundary Objects, rather than the objects themselves, as the essential element of bridging between disciplines.

First, Boundary Objects as typically studied, are working arrangements that evolve out of their context of use rather than being engineered. For example: *"Objects become natural in a particular community of practice over a long period of time. It is not predetermined whether an object will ever become naturalize, or how long it will remain so",* [1], *p299.).* While recognizing the need for "Boundary Infrastructures"- i.e. collections of boundary objects whose use becomes institutionalized, Boundary Objects, for the most part, are not planned. I can muster anecdotes about both successes and failures for each of my own favorite Boundary Objects listed above. It is even more difficult to understand what characteristics of Boundary Objects predict their success. Are Boundary Objects most successful when they are small vs large, informal vs formal, inserted early vs late, high vs. low fidelity? Does it matter if management supports them or not? Are they best owned by the HCI or SE staff? (in practice artifacts are seldom shared in Industry). My experience gives me no answers to these questions, and without answers, we are limited in our "bridging" efforts to creating a random grab bag of objects that projects "might" want to try.

The second problem in attempting to use Boundary Objects as a prescription is that their success seems absolutely dependent on the attitudes of the participants- the human-context of the collaboration. While *"Boundary objects arise … from durable cooperation among communities of practice"* [1, p297, underlining mine], it is not at all clear that using even the best of Boundary objects improves cooperation unless participants' attitudes towards cooperation is already positive. In my experience, for example, while software prototypes can be an extremely useful object in "friendly" environments that are already collaborative, they usually fail in large, contentious projects that have not fostered a cooperative attitude. The results of Kazman, et al [2] would suggest that friction in Bridging between HCI and SE stems from attitudinal problems. And, while some preliminary evidence may suggest that training can reduce these [3], they are likely to remain a larger problem than that of finding appropriate Boundary Objects.

In summary, while boundary objects are an important vehicle for understanding how and to what extent SE and HCI disciplines cooperate, we

should be cautious when relying on them as a solution to bridging the fields.

[1] Bowker, G., and S.L. Star (1999), Sorting Things Out: Classification and its Consequences. MIT PRESS, 1999

[2] Kazman, R., Gunaratne, J. and Jerome, B.. Why Can't Software Engineers and HCI Practitioners Work Together? In J. Jacko and C. Stephanidis (Eds.) Human-Computer Interaction: Theory and Practice, Lawrence Erlbaum, Mahway, NJ, 2003.

[3] Milewski, A.E. Software Engineers and HCI Practitioners Learning to Work Together: A Preliminary Look at Expectations. Paper Presented at the 17th Conference on Software Engineering Education and Training (CSEE&T 2004), Norfolk, Virginia (USA), March 1-3, 2004

[4] Star, S.L., and J.R. Griesemer (1989), "Institutional Ecology, 'Translations', and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology 1907-39", Social Studies of Science, Vol. 19.

[5] Walenstein, A. Finding Boundary Objects in SE and HCI: An Approach Through Engineering-oriented Design Theories, Workshop position paper accepted at *ICSE'03 - International Conference on Software Engineering, May 3-11, 2003. Workshop on* Bridging the Gaps Between Software Engineering and Human-Computer Interaction