Evaluating Software Architectures for Usability

Len Bass and Bonnie E. John

Carnegie Mellon University
Pittsburgh, Pa 15213
ljb@sei.cmu.edu, bej@cs.cmu.edu

For the last twenty years, techniques to design software architectures for interactive systems that support usability have been a concern of both researchers and practitioners. Recently, in the context of performing architecture evaluations, we were reminded that the techniques developed thus far are of limited utility when evaluating the usability of a system based on its architecture. Techniques for supporting usability have historically focussed on selecting the correct overall system structure. Proponents of these techniques argue that their structure retains the modifiability needed during an iterative design process while still providing the required support for performance and other functionality.
We are taking a different approach. We are preparing a collection of connections between specific aspects of usability (such as the ability for a user to "undo" or "cancel") and their implications for software architecture. Our vision sees designers using this collection both to generate solutions to those aspects of usability they have chosen to include and to evaluate their system designs for specific aspects of usability. Our contribution is a specific coupling between aspects of usability and their corresponding architecture. We do not attempt to designate one software architecture to satisfy all aspects of usability. Details can be found in [1].

References

1. Bass, L., John, B. E. and Kates, J. Achieving Usability Through Software Architectural Means, CMU/SEI-2001-TR-005, April 2001, Carnegie Mellon University, Pittsburgh, Pa.

Acknowledgments

This work is supported by the U.S. Department of Defense.

Discussion

F. Jambon: Do you think that tools can verify some of your checklist items automatically?
L. Bass: Yes, some of them can be calculated automatically, but this is not the purpose of this work.

M. Reed Little and L. Nigay (Eds.): EHCI 2001, LNCS 2254, pp. 37–38, 2001.

J. Coutaz: Are your patterns expressed at the conceptual level or implementation level?
L. Bass: It depends on the scenario; for instance, cancel pattern is pretty much implementational.

J. Röth: What (scenario, checklist, etc.) was the most useful to the software engineers?
L. Bass: Found that checklist was most useful part of work for software engineers. Did I forget that? Have I thought about it?

D. Salber: I like the idea of being proactive versus testing after the fact. But actually you should do both. What support does your approach offer?
L. Bass: Approach intended to be complementary to existing software practice.

D. Salber: Then what support for coordinating the two phases?
L. Bass: Try to have usability engineers plus software engineers talk at beginning of V-cycle not just at the end (what they do actually).