Languages for Developing User Interfaces

Brad Myers (editor), James and Bartlet Pubishers International, 1992.

Reviewing a computing book that was published before the web was born seems a little anachronistic! But, ten years ago, this book was about the future of languages for programming user interfaces and languages for the user to do their own programming. Today is that future.

And, well, looking back all the chapters in the book look rather simplistic to people used to .net or XML. Indeed lots of things have changed, but on closer reading almost all of the issues and concerns there were then are still with us. We haven't really moved on. Even though we don't talk about HyperCard now (and it *was* an elegant, ground-breaking system), we are in danger of forgetting too much of the early work and the general insights. Who remembers Garnet, Siri, the Alternate Reality Kit (ARK) or Turing?

Several things surprised me about the book. It is very difficult to do research in programming for user interfaces. Almost everybody has their own system, and as this book is an edited collection from 26 contributors, the diversity is very obvious. They know this much better than anyone else, and that is what they work on. It means that there is very little comparative work, where the merits of alternatives are considered. It also means there are barriers to joining in. By the time I understand, say, Smalltalk, why should I try out Garnet? It would take too much effort, and the results I could get would be behind what I can already do with Smalltalk. Worse, reading about interactive systems is hard work; the reader never *really* understands. But they want to: this book is like a carrot, dangling, inviting you to make the effort and get into some of these systems. I do hope these interesting systems still work somewhere! Today, we might expect such a book to have a fancy web site where we could go for demos, and we could understand how the claims in the book really work.

This problem of building a community of researchers has of course got worse. The systems we use now are much more complex, so it is harder to transfer allegiance and cooperate with others. There are also many commercial systems, and many platforms. Work done on PDAs is unlikely to be exciting for people hooked onto desktops, and *vice versa*.

I found the lack of theory interesting - theory, after all, is what bridges these differences. There is some maths in the book, but the maths was used to say mostly what the languages were about rather than about their semantics. The deepest take-home, in my opinion, is an observation of Randall Smith, David Ungar and Bay-Wei Chang (writing in chapter 5): there is a fundamental difference in kind between visual user interfaces and programming languages. Users will always have to make a jump from using to programming: the only way to avoid the jump (if you want users to have flexibility) is for programming to be done visually, as if the program objects are physical objects. Their argument for this claim is very good, and because of space, I shall leave it as a reason for you to get and read the book!

The lack of theory and the diversity of the contributions, even ten years ago when things were simpler, is exactly why we are making so little progress today. We are all doing different things and have no common experience or theory to draw on. And that's exactly why there is so much still relevant in the book. In short, anyone doing research in programming languages, user interfaces, or programming languages for user interfaces should be familiar with this book and its contents. Being a decade old, there is a clarity and the excitement of imminent progress that we often miss today, and the book has left a legacy we can still usefully build on.

Reviewed by Harold Thimbleby.