JUS JOURNAL OF USABILITY STUDIES

Vol. 3, Issue 3, May 2008, pp. 99-102

User Experience Design: The Evolution of a Multi-Disciplinary Approach

Deborah J. Mayhew, PhD
Deborah J. Mayhew & Associates
88 Panhandle Road
PO 248
West Tisbury, MA 02575
USA
drdeb@vineyard.net
http://drdeb.vineyard.net

An Early and Brief History of Software Engineering

The following sections present the history of software engineering in the 1960's through the 1980's.

The 60's and 70's

In the beginning (let's say in the late 1960's), there were programmers. Period.

There was little or no real division of labor or specialization in the software development industry. Programmers did business and functional analysis, they did project management, they did system architecture, they did user interface design, they coded, and they did software testing and quality control and user support. In some cases they even did sales.

There were also no standard or commercial software development methodologies in use. Each development project team leader (a programmer) in each company simply conducted their project in whatever way made sense to them, learning from their own experience.

I started my career in the software industry in 1978. I was in the middle of a PhD program in cognitive psychology at the time. I had picked up some programming skills as a research assistant in graduate school, and took a job one summer as a programmer in a software development consulting firm to make a little money to support myself while finishing my degree. I planned to leave the job at the end of the summer to return full time to my degree program, but became enamored of the software development business (not to mention actually earning a decent salary) and decided to stay on. I finished my PhD around a full-time job at the consulting firm.

The 80's

I remember the reaction among developers at the consulting company I worked at, back in the early 1980's, when the company executives first tried to implement a standard development methodology (they were trying to standardize their own internal development methodology in hopes of ultimately packaging it as a product to sell). Outrage! Project-leader programmers were appalled at the idea of not being able to manage their projects as they saw fit. They complained that having to follow a standard would squash their own creativity and ability to innovate, degrading their job satisfaction. Developers were similarly aghast, and for the same reasons, when the company started to move towards greater specialization and division of labor on development project teams.

One early change was to create a specialized role called a "business analyst," who was no longer a programmer but rather a business expert. Programmers would no longer do business analysis. This was the first responsibility taken away from programmers who were used to doing everything. They soon lost the rest of their client handling responsibilities, which were next assigned to "account managers," again, strictly a business role. Then another new specialized role emerged—a project manager—that similarly had little real technical role, and became primarily a management role. Again, programmers were disgruntled at the need to be "managed" and to lose yet more control over their projects.

Role specialization continued to expand. In the mid 80's an approach to system architecture that cleanly separated user interface code from application code emerged. This separation facilitated application maintenance and enhancement. Correspondingly, the programmer role began to split into "back end" engineers" and "front end" programmers—a further specialization which removed the user interface design role from at least one segment of the programmer role, but still left it in the hands of programmers. Specialized tools for architecting and building independent user interface code began to emerge, and correspondingly one segment of the programmer community began to specialize in these tools.

And ultimately, the final insult to the ever diminishing role of the programmer—the field of computer-human interaction, which had begun to emerge in the late 1970's, began to take hold. Programmers now had to take advice from "user interface designers" and "human factors" professionals, who were usually not software professionals at all. They tended to be people with backgrounds in psychology or human factors. Many, if not most, programmers (especially those who had come to specialize in user interface tools) were used to, and liked, doing user interface design and considered themselves perfectly competent in this area, and were not at all happy to lose control of it. Early usability professionals were usually considered unwelcome interlopers by developers, and were still not well understood—and thus not well supported—by management. It was a tough role to play in the 80's.

A Recent and Brief History of Software Usability Engineering

The following sections present the history of software usability engineering of the 1990's and 2000's.

The 90's

By the early 90's, development role specialization and standardized development methodologies had become common and accepted in the software development industry, especially by newcomers to the industry who had no history of broader roles and more freedom to manage themselves and their projects. On the other hand, being a usability professional in a software development organization was still not routinely accepted and continued to be a tough role to play, not particularly welcomed by developers and under-supported by upper management. And, while the software industry had by this time adopted the notion of standardized development methodologies, usability professionals had not yet found a formal place in these methodologies, and were much like programmers in the early 80's—generalists who did not specialize in any particular usability role (such as user interface designer, usability requirements analyst, usability tester, etc.) and did not have any structured, standard approach to integrating their expertise and work into software development teams and methodologies. History repeats itself.

Then came the dot com boom, starting in the mid to late 90's.

I remember taking on a long term consulting job with a dot com start up about this time. While I was nearly 50 with 25 years of work experience in the software industry, most of the employees in this start up were barely out of school and in their 20's. They were completely free of any negative feelings about their very specialized roles of back-end or front-end programmers with no other responsibilities, because it was all they had ever known. They had no issues at all with the notion that some other kind of specialist would be completely responsible for the user experience design that they would build.

This client company also had a new and emerging type of user experience professional in the mix—the graphic designer. Suddenly traditional usability professionals like myself found themselves in the same position as programmers had 20 years earlier: needing to share responsibility with a new and very different sort of user experience professional, for what they had been entirely responsible for in the past. Not long after that, yet another new specialized user experience role emerged—the information architect.

My book, *The Usability Engineering Lifecycle* (Morgan Kaufmann Publishers, 1999, http://drdeb.vineyard.net/index2.php?loc=7&nloc=1), was intended to bring the practice of usability engineering more in line with software engineering practices at the time, by outlining a structured and engineering-like methodology for accomplishing good interface design that could be integrated into the underlying software development methodology. Such a standardized methodology was new to the field at that time, and again, represented a change comparable to the change programmers went through 20 years earlier. The Lifecycle approach was also intended to more clearly establish the role of user experience professionals in the development project team. We needed to catch up with the practices in the industry we were operating within.

I remember that initially, learning how to divide up responsibility for user experience design with graphic designers and information architects was an interesting challenge. It was clear to me from the get-go what a powerful combination our separate skill sets could be, but I often found that usability principles came into conflict with graphic design and branding principles, and had to work out ways to arrive at optimal compromises. That entailed them learning my principles and me learning theirs, at least to some extent. I felt I had a good sense of the boundaries between our skill sets and responsibilities at that point, and it was more than clear that we were much more effective as a user experience design team than any of us would have been alone.

The 2000's

Recently, I have discovered a new emerging type of user experience specialist: the "persuasion architect." This specialist has a marketing and sales background, and focuses on aspects of a Web site user experience design that contribute to "conversions," that is, to the number or percentage of site visitors that ultimately contribute directly to the business goals of the site, such as buying a product, signing up for a newsletter, registering, using the site for support, etc. The design aspects that contribute to converting visitors into customers are quite different than aspects that contribute to making task completion easy and fast, making a site visually appealing, or architecting the site information or functionality in the most natural way.

I have a new business partner (Todd Follansbee, Web Marketing Resources, www.webmarketingresources.net) who is a leading persuasion architect. I have started working with him on a new tool he is developing that will allow a quick, objective, and quantitative assessment of the user experience design of a Web site, particularly of an e-commerce Web site. This tool allows reviewers to score a Web site on how successfully it currently incorporates some fundamental usability principles, content principles, graphic design principles, and persuasion architecture principles. I applied a beta version of this tool (called "The Dudley," soon to be made available through a Web site) to my own company Web site. While my site scored high on usability, it scored quite low on persuasion architecture. This is just not part of my skill set, in the same way that graphic design is not my skill set, which is also reflected in my Web site design.

The persuasion architecture part of The Dudley tool measures whether a site adheres to the following standard marketing concepts and principles:

- Provides a clear “value proposition” on the home page (clarifies the offerings of the site).
- Focuses on product or service benefits rather than features (customer focused language).
- Effectively addresses issues of credibility and trust.
- Effectively addresses issues of privacy.
- Uses visitor language (avoids corporate or industry jargon).
- Provides all information visitors will need, just in time to decide to take each step along the conversion path.
- Presents salient and clear “calls to action” (provides a clear path to the conversion point).

These standard marketing concepts and principles are not usually found in the repertoire of other types of Web site user experience professionals such as usability engineers and graphic designers. I redesigned my site to achieve The Dudley’s persuasion principles that my site had failed on, and it is much better for it. It is also doing significantly better in its search engine rankings (showing up on page 1 or 2, instead of page 1-10, of a Google search result list on key search phrases).

Going Forward

Those of us in the Web user experience profession need to partner with this new kind of specialist. Easy task completion (traditional usability) is not enough in the Web world. Appealing visual site design is not enough. A site visitor needs to not only be attracted to a site and able to figure out how to buy (or register, sign up, etc.)—they need in addition to be able to tell quickly that a site will meet their needs, and they need to want to buy from this site, as opposed to a competitor’s site. This is a key aspect of overall Web site success.

So yet again, we find ourselves in a position in which we need to embrace and adopt new specializations within the user experience profession. At this point, a multi-disciplinary team approach to Web user experience design is the key to overall success within our profession. Just as programmers had to do 20 years ago, user experience professionals need to embrace other specializations within our field and learn how to work with them most effectively. Part of the key to this is defining and adopting a development methodology that clearly incorporates all these skill sets in a balanced way. This may be the main challenge before us going forward. Programmers have gone through a lot of changes and had to adapt in many ways since the 60’s. But they are still considered indispensable to software development. While the usability profession has made a lot of progress and inroads in the industry over the past 25 years, we still are not, by and large, considered integral and indispensable. By aligning ourselves with other user experience specialists of the types described above, and jointly inserting our methods into the standard and underlying software development methodologies in our organizations, we have a better chance of achieving this strategic goal.

About the Author

Dr. Deborah J. Mayhew
An internationally recognized author, teacher, speaker and consultant on software user interface design and usability engineering. Her most recent book is *The Usability Engineering Lifecycle*. Dr. Mayhew holds a B.A. in Psychology from Brown University, an M.A. in Experimental Psychology from the University of Denver, and a Ph.D. in Cognitive Psychology from Tufts University.