Debunking the myths of UI design

e-mail it!

An expert takes on a range of beliefs that are now common in the software development community

Paul Smith (pwsmith@ca.ibm.com)
IBM, Toronto Software Lab
March 2001

Many powerful myths exist for software developers on the role of design; specifically the design of user interfaces. Drawing on 13 years of experience in the field of user interface design, Paul Smith describes and debunks these myths.

Search Advanced Help

Related sites: Go

Contents:

Introduction
Organization
Methodology
Expertise
Looking to the future<
Resources
About the author
Rate this article

Introduction

The software development industry is relatively young, rapidly evolving, and surprisingly little is automated. It is therefore an intensely human and social endeavor, having all the phenomena characteristic of any cultural activity -- communication issues, organizational issues, customs, values, fashions, and myths. It brings out the best and the worst in people. Personalities determine much of what happens. It is more like making movies than engineering cars.

Software development would benefit greatly from extensive study by sociologists, anthropologists, and clinical psychologists. As we await such analyses, let's document some beliefs embedded in the culture of software development, specifically about user interface design. This article identifies a series of cultural myths and presents realistic conclusions from my extensive experience in user interface design. I'll also address current trends in World Wide Web design.

Let's start with a fundamental myth. It's not necessarily unique to software development, but it's especially prevalent there.

MYTH: Design is a luxury.

REALITY: In his 1990 book, *The Design of Everyday Things* (see Resources), Donald Norman offers the traditional antidote for all those who underestimate the role of design in our lives. The issue is not whether design should happen -- it always happens. Everyone in product development does design work, whether they know it or not. Moreover, the design is the dimension of a product that customers see and feel; it is what satisfies or disappoints them.

In software development, design is widely misunderstood and undervalued. Often no explicit user interface design is done separately from the code. Iterative design then becomes recoding. This is a short-sighted strategy because it results in significantly more code being written in the long run. Because design is unavoidable, the real issue is whether it is left implicit in the software being developed, or made explicit and captured separately. The useful debate is about how to do design work well, and how to capture it in an optimal form for communicating to those who implement it.

An explicit user interface design can focus on how a product satisfies customer wants and needs rather than on how to build it. This can make implementation more difficult, but that is the price to be paid for focusing on the real goal of product development. An explicit design allows for early detection of implementation issues, as well as for placing the primary focus on satisfying users. Simultaneous design and implementation sometimes occurs on small projects. However, this approach is not scalable and requires some very special, multitalented people. Software development superheroes are in short supply.

Organization
Because almost all software development is a social activity, how groups of people are organized will necessarily have a profound effect on the user interfaces created by those groups.

MYTH: Developing a customer solution is just conceptually and logistically too big to tackle; people working at the solution level will get bogged down in unmanageable issues. You should develop the product as a set of components, and then organize your development team to mimic the product. (Of course, sometimes the components are developed independently and subsequently "integrated," producing the same kind of organization.)

REALITY: This is a kind of "divide and capitulate" strategy. Ultimately, the product design mirrors the company organization, and the product integration breaks down along the very component boundaries created within the organization. Often the user interface is expected to paper over most of the deep cracks after the product function has been independently developed.

Such an organization makes it difficult to create solution-wide strategies and plans. In addition, marketing's vision for the product is different from that of the coders, which is different from that of the technical writers, which is different from that of the service personnel, and so on. Similar functions are repeated here and there throughout the product, and often the user interfaces for the same function can be very different. Sometimes, there is no overall product evaluation because there are only component builds until just before the product ships.

The remedy is to assign lead designers the responsibility for producing a tightly-integrated solution, to do so right from the beginning of product development, and to give them the authority over the components to go along with the responsibility. Most of the steps needed to ensure an integrated customer solution will then be taken by the design team. The critical piece is to have people designing at the solution level.

MYTH: User interface design responsibility is best distributed within components. Each implementer designs the user interface for his module.

REALITY: If a significant proportion of coders expect to do user interface design work on their piece without overall design direction, this can result in users being confronted with a product user interface that was designed by a committee of hundreds! Solution-level user interface designers must lead teams of component developers. This role includes responsibility for overall user interface architecture, integration, and high-level design.

MYTH: User interface guidelines solve the problem of distributed design.

REALITY: The best way to get a good user interface is to get good designers to design it, not to write guidelines so that bad designers can design it. Let's assume that the user interface guidelines are in fact read. Interpretation is always a problem. The need for interpretation can be avoided if guidelines are restricted to low-level design choices, but that still leaves the most important design decisions to be made by unskilled people. Pointing these people to a model user interface design is helpful. The primary responsibility of user interface designers is not to improve the way coders do design work; design work needs to be done by specialists in human-computer interaction (HCI).

MYTH: Teamwork.

REALITY: As long as team members have different goals, teamwork does not happen. If a coder's objective is to ship on time and a designer's objective is to satisfy customers, real teamwork is virtually impossible. Often all disciplines involved in creating product externals are not accepted as equal partners with the coders. Team members must be put on an equal footing. Each discipline needs its own design specialists, including someone to do the internal design of the code. Each discipline needs to put trust in the expertise of the others. Each discipline needs user-centered design processes. Each discipline needs prototyping tools and methods.

Most importantly, each product team needs to be led and managed by the person with overall responsibility for the product -- both development and marketing. It's important to focus lobbying efforts on the executive with overall product responsibility. Management decisions need to be influenced by user feedback data, but don't fall into the trap of wasting time trying to influence the wrong managers -- managers whose performance is not measured on the basis of the success of the product in the marketplace.

MYTH: Human-computer interaction experts should carry out both design and feedback activities.

REALITY: HCI organizations generally have individuals with skills in both user feedback and user interface design. Because there is never enough time, they inevitably have to trade off one for the other (not to mention the conflict of interest in evaluating your own designs). Usability specialists have typically conducted tests and made design change recommendations. Usability work needs to evolve into two separate specialties -- user feedback and user interface design.

MYTH: Software companies generally assign usability a high priority.

REALITY: A common reason for ignoring usability is the cult of the "killer app," one of the worst things ever to happen in software development. The implication is that if you can just find the right function, user-centered design and all those other "frills" are unnecessary. However, for every killer application, there are thousands of merely useful applications. Consider the parable of the stock market application: If you build software that tells you tomorrow's stock prices, people will put up with a very bad user interface, but if you build something that tells you yesterday's stock prices, it had better be usable.

Many people in the software industry still don't understand usability or user-centered design, so they prefer to work on projects where usability is not critical to success. In such projects, usability can be managed as a laundry list of low-cost, cross-component consistency items (for example, all dialog windows should have a gray background, place the Help button to the right of all other buttons, etc.).

MYTH: Industry pressure requires a full-court press on the current release.

REALITY: Key people need to be dedicated to release N+1. It is short-sighted not to have the lead product designers laying the groundwork for the next release while the coders are working out the implementation of the current release. As soon as the coders working on the current release are set loose on the follow-on release, the opportunity to set solution-level user interface direction starts to vanish.

Methodology
In an industry where there is so little consensus on methodology, the methods people choose often derive as much from preconceptions or wishful thinking as from proven effectiveness.

MYTH: Successful software companies develop technology, package it, and market it. The packaging is essentially assembling available parts into a customer solution.

REALITY: Products need to be designed from the outside-in rather than from the inside-out. Some familiar consequences of inside-out software development are: feasibility cannot be judged before coding begins; the size of the coding effort cannot be judged before coding begins; the size of the design effort is hidden in the coding effort; and the ship date is chosen before the product is designed. Take the approach of designing the user experience rather than designing the user interface after all the function is set.

MYTH: Usability and functionality have separate requirements.

REALITY: This is a false distinction. *All* requirements support the user's tasks. The distinction perpetuates the tendency to develop technology, and then treat the user interface as a layer between the user and the underlying technology. It also allows usability requirements to be treated separately (and assigned less importance) than functional ones.

MYTH: Users should design user interfaces. Designers should avoid making decisions on behalf of the user --

users know best what they need. The best thing we could do is give them a blank sheet of paper and get out of their way.

REALITY: Occasionally user interface developers get lucky when choosing users to take the lead in design. However, most users are good at reacting to product designs and notoriously bad at identifying the sources of their reactions. Participatory design is a good thing, but is sometimes treated as a panacea leading to abdication of responsibility. The responsibility of user interface designers is to design, and then get user feedback on their design ideas.

Participatory design helps resolve design issues. There is no substitute for the insight gained by watching someone struggle through an important task with a product you designed or built. But there also needs to be separate evaluation activities that are comprehensive and quantitative, and which provide opportunities for users to disconfirm the biases of user interface designers. Designers who ask a few users for their opinions assume that users are always dissatisfied when confronted with bad software, and always know the sources of their dissatisfaction. (It is a simple matter of asking them.) Final evaluation of the design should be done by specialists in user feedback.

MYTH: Design teams should include a customer. This is a shortcut to getting user input, and it allows you to skip the messy business of usability testing.

REALITY: A single customer cannot represent the target population for the software you're developing. More importantly, including a customer on the design team doesn't replace the need for user feedback. The risks of being misled by this form of participatory design outweigh the benefits.

Another potentially dangerous shortcut is the expert user interface review by an HCI professional (after the interface has been designed and implemented). This person's opinion may be better than a coder's, but it is still just an opinion until confirmed by feedback from representative users.

MYTH: The first user interface deliverable is a working prototype.

REALITY: Many developers start design work by building one finished dialog, and then evolving the rest of a working prototype according to the assumptions underlying that dialog. There are interim stages to user interface design. Design needs to take place in stages appropriate for user feedback: first metaphor, then user model, then high-level design, then low-level design. High-level design concentrates on breadth as opposed to depth. It illustrates a system's usefulness and "conceptual consistency." You produce high-quality interface designs by starting at the high level and progressively refining the design to a level detailed enough for coders to implement. Make sure the product will work as a system before designing the details.

MYTH: Eliminate alternatives ASAP. Designers should not waste time exploring alternative approaches, but should stick with one unless it collapses under its own weight -- and only then explore a new approach.

REALITY: All too often, design alternatives are quickly rejected out-of-hand, usually for reasons of technical complexity or coding effort. In addition, designers sometimes tend to go down the first "design path" they see, taking each fork in the road as it arises without consulting users. It is important to dedicate a period of time to quickly generating and exploring a number of high-level design alternatives. Then, select the one with the most promise for supporting the critical user tasks (not just the one that's easiest to implement).

MYTH: Prototypes replace the need for specs. They capture all the information that's needed to code the final product.

REALITY: Prototypes are excellent for illustrating the appearance of user interface and for demonstrating some of its intended behaviors. However, each coder takes different things away from a prototype, depending on how he explores it. To avoid this, some aspects of the behavior of the prototype have to be documented. The need is for prototype-driven specs rather than spec-driven prototypes. There is also great benefit in explicitly identifying design alternatives, design questions, design objectives, and design rationale. These need to be communicated to all those working on the user interface to ensure that everyone is on the same wavelength.

MYTH: Innovative design is better design. Users prefer software that has creative new methods of *presentation.*

REALITY: People have not digested many of the user interface innovations they have already seen and touched. The best policy is to plagiarize whenever possible (as long as you stay out of jail) and innovate when necessary. In other words, follow corporate or industry standards and take advantage of others' innovations.

MYTH: User interface designers know how to use information about the user's environment and tasks.

REALITY: In many software development projects, lots of user input gets collected and lots of user interface design takes place. The problem is that the former often does not have much influence on the latter. They are parallel universes. We know how to use feedback from late in the development cycle when users are reacting to a working prototype. However, early input on user tasks and preferences has much less impact on design work because we don't know how to use it. If someone claims to have solved this problem, be very skeptical.

Expertise
Because user interface design is not an established and recognized discipline, the vacuum is filled by all sorts of people with very different skills.

MYTH: Good user interface developers can both design and code the user interface.

REALITY: An analogy would be to ask an HCI expert to design the user interface and the internals, and then go ahead and code the product. User interface coders implement designs; their skill is designing and coding the internals. Kapor's (1996) ideal of a software designer as distinct from software coder is not the current practice. Typically, user interface design is done by a coder on a part-time basis. This practice is attributed by Winograd (1996) to the immaturity of software development. In building construction, the division of labor between architect and contractor evolved as the industry matured. Occasionally, an exceptional software developer can both design and code the user interface, but the majority do not have the critical skills for user interface design. These include:

- The ability to apply the principles of navigation, selection, direct manipulation, consistency, and standard interaction styles
- The ability to apply user input to design (wants and needs, tasks and scenarios, competitor information, feedback on user interface designs)
- A grasp of high-level design (metaphors, user models, systems design, usefulness and conceptual consistency, task flow)
- An understanding of user interface paradigms (form-based, menu-based, application-oriented graphical user interfaces, multiple-document interfaces, object-oriented interfaces, compound-document interfaces),and knowledge of what makes a good hybrid

Of course, a coder needs to appreciate design (just as a designer needs to appreciate implementation constraints). Each has their own expertise. Communicating their designs is at least half the job for user interface designers. This task continues right up to the product ship date, in order to avoid the "last person to touch the code is the user interface designer" phenomenon. Communication between practitioners of these two disciplines is a fascinating subject worthy of further analysis.

MYTH: User interface design is a veneer added to the product. It is cosmetic -- fit and finish.

REALITY: Those who subscribe to this myth usually have in mind the role of visual design and information design in multimedia titles and in Web pages. There, the main tasks of users have been browsing and reading (Weed, 1996). Productivity software needs more interaction than visuals because users are much more active with other tasks such as creating and editing. User interfaces require the design of information (terminology, content of online help), of visuals (splash screens, icons, layout), and of interaction (relationships between user behavior and program behavior). The mistake is to believe any one of these skills is the whole story.

MYTH: User feedback speaks for itself; it does not require interpretation. If one or two users dislike a feature of the design, change it. If one or two users dislike the new design, change it again.

REALITY: Just as there is user interface design expertise, there is expertise at collection of data. This expertise involves knowing when to collect data, how much to collect, when to stop, what to measure, how to interpret it, how to communicate it, and whom to communicate it to.

Looking to the future

The revolution that is the World Wide Web has created opportunities to do user interface design work differently and better than in the past. However, beware the zeal of revolutionaries. They are in the mythmaking business.

MYTH: The Web eliminates the need for interaction design or user feedback skills.

REALITY: The Web is recapitulating the history of graphical user interface design, only this time it is likely to be worse because the implementation hurdle is lower and the cycle time is shorter. This means the integration problems will be worse. There are many people designing system components (pages/links) that must be combined to support users' tasks.

Schedule pressure has increased, and the belief is that Web pages are so easy to build and online feedback so easy to collect that we all might as well skip design and feedback during development. Clearly, there is a need for streamlined development methods for the Web context, but all the realities described above still apply.

In addition, during the HTML-only period, there was an illusion that standards had evolved to make Web user interfaces more consistent. However, we're already seeing that as the limits on interactivity disappear, so does the illusion of standards and consistency.

There is still a publishing mindset with respect to the Web, with heavy reliance on graphic and information design. As it is used more and more for productivity as well as publishing information, we'll require much more interaction design, but recognizing that need will likely follow a familiar period of user frustration.

Resources

- Read The Design of Everyday Things by Donald A. Norman, Doubleday, March 1990. (ISBN:0385267746)
- For more on HCI, check out the Human-Computer Interaction Resource Network.
- The Association for Computing Machinery offers an online version of *Interactions*, its bimonthly magazine for designers of interactive products.
- Usable Web is a collection of links relating to human factors, user interface issues, and usable design specific to the World Wide Web.
- Visit the IBM Ease of Use site for the latest in design guidelines, from designing for the Web to out-of-box-experience. Also, find out how you can participate in IBM's annual worldwide conference on ease-of-use: "make it easy 2001."

About the author

Paul Smith has been a human-computer interaction professional at the IBM Toronto Software Lab for the past 13 years. In that time, he has served as manager of a UI design department and, for the last four years, as the user-centered design lead for IBM's WebSphere Commerce Suite. Paul earned his Ph.D. in Cognitive Psychology from the University of Toronto in 1991. He can be reached at pwsmith@ca.ibm.com.

e-mail it!

What do you think of this article?

○ Killer! (5) ○ Good stuff (4) ○ So-so; not bad (3) ○ Needs work (2) ○ Lame! (1)

Comments?

Submit feedback