DevSource™ Sponsored by Microsoft
Microsoft Visual Studio.net

Towards Usability: An Interview With Theo Mandel

By Rebecca Rohan
October 25, 2004

Website: www.devsource.ziffdavis.com/

You've developed an application that can pick all the winning lottery tickets, make solar power 99.9% efficient, and spew just the right bon mots to propel your favorite candidates to power. The only hitch is that users find it such a pain that it's not worth the trouble to start it up.

Perhaps it's time to have a tete-a-tete with a usability expert.

Well, DevSource hit the jackpot for you with Theo Mandel, Ph.D., author of *The Elements of User Interface Design* (Wiley, 1997). Dr. Mandel earned his Ph.D. in cognitive psychology and quantitative psychology, and worked with IBM for 11 years as part of its Common User Access (CUA) group. CUA defined the user interface guidelines for IBM developers, and was used for OS/2 and Windows up to Windows 3.x. As a result, Dr. Mandel did extensive research into GUIs and OOUIs. For the past 10 years, Dr. Mandel has been a consultant in his own company, Interface Design and Development, where he's involved with design prototyping, reviews, and user evaluations.

DevSource: So where do you start?

Mandel: With general principles. There are three rules that help, regardless of the project:

The first Golden Rule is *Put The User In Control*. Let the user drive the application, instead of the developer defining how users have to use it. If the developer thinks the users have to do things in a certain order, for example, then users are like passengers on a train instead of drivers in cars. Let them get from point "A" to point "B" by different routes. Give users multiple ways to do things.

Let users customize applications and set up preferences. For example, Microsoft applications might not have Save As or Close in the menu bars, but users can add them, and that's really good. I can choose the kind of layout I want if they have multiple toolbars.

DevSource: I take it you would say people should be able to put just the tools they want on a toolbar or create custom toolbars that fit just the space they want, rather than having to crowd up the screen with whole toolbars for each tool they'd like to add?

Mandel: Right — pallets of toolbars. People should have the ability to personalize and customize things.

Be flexible. Let people use both the keyboard and the mouse, but don't assume they always use a mouse. If data input is intensive, allow keyboard use without having to go to a mouse. We assume they have a mouse, but don't assume an application should be optimized for the mouse, especially if the user is working with a text editor or Quicken, where you're typing in data.

If you're watching people do repetitive tasks, you see that they're going back and forth: keyboard, mouse, keyboard, mouse.... Wouldn't it be nice if they could just pick one and not have to go back and forth — if they were in a keyboard kind of mode?

DevSource: That would help a user stay in control.

Mandel: Be forgiving. Let the user do stupid things, then provide immediate reversibility. They can do something destructive, but the program should pop up a warning, such as, "Are you sure?" One warning is nice, but some take it to the extreme. "Are you really sure?" can be intrusive. "Put a check box in there that says, "I don't want to see that again." But you need an Undo.

The idea is to be helpful and give some idea when the user is doing something potentially destructive, but not to be too obtrusive about it. At the same time, let them undo something and let them get it back in a reasonable period of time.

DevSource: Do you see user control issues getting worse as the user gets to a higher level? For example, if the user is designing or editing a Web page, and the software leaves nonstandard source in large blocks of text with no white space or indentations, as if no human would want to manually edit their code after generating the main design? Like they've bought a house without keys to the tool shed?

Mandel: You always should try to give the user the best of both worlds. Try to do the grunt work for them. They don't have to understand HTML, but you have to give them the ability to go in and tweak that. Design for multiple levels of users: Support the novice user and the advanced.

At the same time, you don't want to force the more experienced user to go through the same steps as the novice user. You might supply a wizard, but don't force the more experienced user to go through it.

Mandel: The second Golden Rule, *Reduce User Memory Load*, is about grasping how humans think and learn and understand. People are good at recognizing things, but not remembering or recalling things. If someone asked you to name the states, it would be harder for you to answer that than if someone handed you a list and asked you to recognize them.

When you want to ask for information, if you know the valid sets of information, don't just put in a field, give a list, such as state fields (or any pool of information). If they want to type it in, fine, but don't *force* them to if they want to browse through a list or a hierarchy tree.

Provide good cues for people. When they're logging in, you should give a cue, and while they're logged in, you should give another cue —somewhere in the title bar or status bar — about where they are. We're not good with short term memory. The context of where we are is always good.

Real world metaphors help people transfer their learning to the computer. Things like Quicken are good for balancing a checkbook because the metaphor is a checkbook. They took a complicated real world tool and put a simple face on it with a paper and pencil checkbook.

DevSource: Some people find real-world metaphors, like a checkbook, babyish, and feel they just get in the way. They would prefer a straight spreadsheet-style page set up with the proper headings. What about making an alternative interface for them?

Mandel: That could be done, but I would assume that the bulk of the people using Quicken want the comfort of the metaphor. A developer still uses a checkbook. Just because he's using a spreadsheet, I don't think they would want a bare spreadsheet.

DevSource: Microsoft Money lets you uncheck "Show Transaction Forms," and get rid of some of the metaphor.

Mandel: One of the problems with a metaphor is that, when it gets stretched (as with a checkbook and stocks), it loses the effectiveness of the metaphor. Metaphors are good for the tasks involved.

When we look at toolbars for tasks that are very generic, common actions are pretty understandable. When you try to come up with icons for concepts, that's when you get bizarre icons and nobody has any idea what they mean.

There's a whole area with icons (and user memory load): just putting a picture there doesn't mean people are going to understand it. Half the development team couldn't remember what the icon meant to themselves, so how can they expect their end users to do it?

Another way to reduce memory load is progressive disclosure — giving people information when they need it, when they want it — not just throwing information at them and letting them wade through it to get what they need. In Microsoft Word, 60% of the people might not need 40% of the features.

Try to give users only what they need right now. Give them only the appropriate context, instead of throwing gobs of information at them all the time. Instead, give them specific help, then also give them more details and related topics and let them choose where they want to go. The tendency is to put more stuff than is needed on main pages. Sometimes it's political — different groups want theirs stuff out there — put it on a secondary screen where you can get to it only when they want it.

In the Microsoft Word Options Dialog tabs, when you click on the top row, it becomes the bottom row, so it feels like everything is changing. It loses all consistency in terms of memory.

DevSource: Speaking of Options, I've noticed programs with options scattered throughout menus. Would it reduce the user memory load to place all options on one menu, in an option tree, if necessary? I've seen this done somewhere, but my user memory is taxed at the moment....

Mandel: Whatever options and preferences and settings the user has available to them, it should be in one area so they can do all these things. I hate having to look all over the application for things that I would call settings. They should be accessible from one menu and not necessarily from one dialog.

DevSource: Where should this be?

Mandel: On the Tools menu bar. And there could be one options dialog, but that options dialog can present a lot of information. It can be navigating between options; it can be done in a more user friendly fashion.

DevSource: I've seen a left window pane with—

Mandel: They have help topics organized that way. It would be organized in a common way, the way help screens are laid out: There would be an Explorer-style interface with a left vertical area with hierarchy where contents are displayed in the right area. You could use that to display your options. That would avoid having so many tabs to navigate with.

DevSource: Do you prefer tabs, if done more sensibly?

Mandel: Tabs are good if there are not too many. If you have two or three or four tabs. If you have eleven groups of tabs, it starts to fall apart. You can have nested tabs. The highest level can have three or four. It goes back to cognitive abilities. People can't handle more than five to seven items in a grouping, such as having more than seven tabs. With more than seven tabs, you might think of having a list of things on the left.

Mandel: A lot of people latch on to consistency. They believe that, if it's consistent, they'll have something usable. But if it wasn't right to start with, it will be consistently wrong throughout the whole application. Some inconsistencies are a problem; you can see that different people or different groups within the company did it. You can see this across the major Office suites. [Get it right, then] make sure you do it consistently. That's why the third Golden Rule is *make the UI consistent*.

Users have expectations about what's going to happen, too. Keep the results of interaction the same. If you click on one thing and one action happens, then you click somewhere else that looks just the same, and something different happens, that violates user expectations. A lot of the time that happens because the

application is not consistent in the use of buttons or tabs or links or whatever. You expect the button to do one thing, expect the link to go to another page, expect the tab to do another... but it is not always real clear what was supposed to happen.

Interaction architects focus on all the interaction aspects of an application. I just completed a review of a Web site where, in the main navigation area, a click opened a PDF file. The more you do that to your users, the more tentative they become. They are fearful about clicking on anything.

DevSource: Can we talk a little bit about on-screen Help? I've seen a dramatic decline in the amount of local Help data in some applications. Some programs have employed nonstandard methods such as Web-based help, Help searches that lead users to articles online (when their users may be offline or the company server may be down), help that asks users to answer market research questions when they need to get their problems solved, help that comes in PDF files when users may not have Acrobat Reader or may simply find PDF files inconvenient or confusing, and more. What's going on with application Help, and what can developers do to better aid their users with questions about their products?

Mandel: The best thing is to provide the best context help to support the user before they even get to ask for help. Back to the second Golden Rule, about relieving users of memory load: indicate where they are, where they've been, where they're going, and what modes they're in. If you do, that will reduce the need for them to request help. That's a high level goal.

The next question is where do they go and how do they get there. That's field-level help, possibly through a right click or the F1 key. If the application doesn't provide field-level help, then screen-level help — same thing as field-level, except for the screen; then above and beyond: table of contents, search capability — how it's created or how it's stored is not as important as that they get the information.

If I'm on an airplane and need help, I want help. I shouldn't have to be connected to get help. There's a distinction between getting help and doing research. Just providing a PDF file is not acceptable. Nor is interactive as easy to use as Help should be. It's just not user friendly to dump a 50-page user guide on the user when they have a simple question to ask.

Help should be local, but you can say, "To learn more," and then offer Web content. You're being a good citizen and, hopefully driving traffic to your Web site.

You can learn more by checking out resources at Dr. Mandel's site at www.theomandel.com.