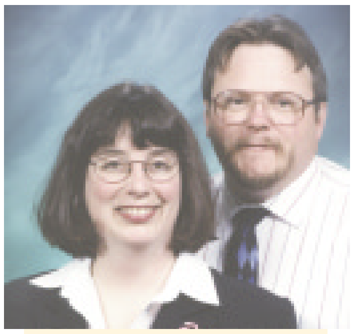
INTERVIEW

Fast Consulting

Susan Fowler and Victor Stanwick

Interview by Elizabeth Dykstra-Erickson



***Susan Fowler** and **Victor Stanwick** are co-authors of The GUI Style Guide (Academic Press) and The GUI Design Handbook (McGraw-Hill).*

Susan is currently working as a usability engineer at Telcordia, Inc., Piscataway, NJ, and has done technical writing, training, and screen design for major Wall Street, reinsurance, and telecommunications firms. She has taught technical writing, technical training, and usability live and online at Fairleigh Dickinson University in Teaneck, NJ, and at Polytechnic University in Brooklyn, NY.

Victor is a designer and technical communicator who consults for web-based mortgage, telecommunication, and accounting firms. He is also a cartoonist, blacksmith, and president of FAST Consulting, a metropolitan New York area web design, technical communication, and usability firm.

How did the "book thing" start? What inspired you to write such detailed and useful books?

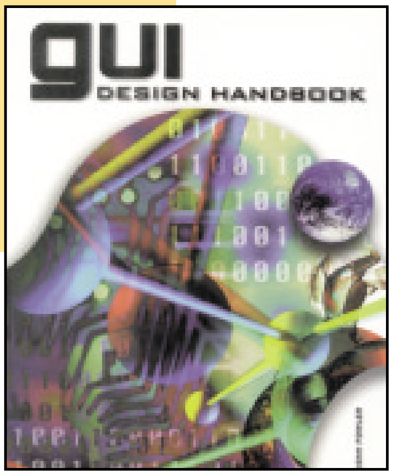
SUSAN: The first book, the *GUI Style Guide (Academic Press, ISBN 0122635906)*, was a thank-you to the programmers who taught me most of what I know about the software industry. They told me how things really worked. For example, when, in my first job, I would go to the programmers crying about losing three hours of work because of some weirdo system bug, they'd listen for awhile, and then say, "It's always better the second time you write it." (This is true, although irritating.)

So when I had the opportunity to return the favor, I did. I had noticed that, when developers did screens, they often made the same mistakes over and over, which they stopped making as soon as someone explained why they were mistakes. So Victor and I wrote down all the things that we had been able to tell them about visual design.

VICTOR: Here's a classic example—at a Wall Street software firm where Susan worked for a few years, the programmers always designed their charts with red lines and green lines (red indicates a falling price and green a rising price). Eight percent of all men are red-green color-blind; men are a majority on Wall Street; therefore, every twelfth person using the software could not understand the charts.

SUSAN: The problem isn't that programmers are insensitive to other people's needs—far from it. But basic visual design skills haven't been part of the standard computer science curriculum, not until recently, anyway.

How would you prefer that people use your books?



VICTOR: We designed the *GUI Design Handbook (McGraw Hill, 1998, ISBN 007059274 8)* to read like most developers' software documentation. Application programming interface (API) reference guides are set up alphabetically, with command names, classes, exceptions, samples, and so on, repeated until the writer runs out of commands. In the *GUI Design Handbook,* the structure is widget name, what it's used for, what it's not used for (the "counter-example," which is a valuable training tool), design rules and hints, usability tests, and see also's—in other words, what other widgets fulfill the same function.

SUSAN: Now, there is a particular reason we designed it that way: When the *GUI Style Guide* came out, I brought it to my office and showed it around, really proud of myself. Keep in mind that the *GUI Style Guide* is organized into standard text chapters on everything from window design to internationalization to multimedia. One of my friends, a programmer, thumbed through it, nodded a few times appreciatively, and then said, "This is great. Next time, though, can you write it so I don't have to read it?" In other words, no chapters, just quick reference.

So the answer to your question is that we expect programmers to look up a widget and find out how to use it and when to avoid it. But of course it doesn't really work that way.

Why not?

SUSAN: First of all, it seems that our audience is not primarily programmers, although we do spot the books on programmers' bookshelves when we visit other offices (that's a thrill). No, it seems that our audience is mostly usability engineers.

This disappointed us at first. Why aren't more programmers buying the book? It was designed for them! But in retrospect, we realized that our audience wasn't programmers or usability engineers—it's software interface designers. And who the interface designers are depends on when you ask.

Twenty years ago, programmers were the interface designers simply because they were the only ones who knew how to use the available tools. Then came GUIs and GUI programming tools such as Visual Basic. Technical writers started doing the interfaces themselves because it was easier to do the screens right than to try to document screens that didn't make sense. The human factors and industrial psychology people also converged on the same issue at the same time, reacting to the cries of horror and despair of software business owners being driven into bankruptcy by customer service calls.

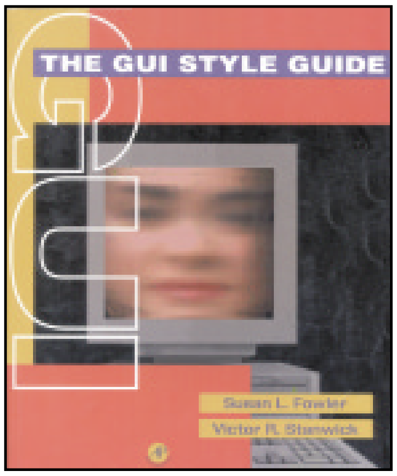
So, now usability engineers are the dominant group of interface designers. The fact that so many of them bought our books two and even five years ago indicates that they were prescient as well as smart.

A few years ago, there was a debate on UTEST, the online usability discussion group managed by Dr. Tharon Howard (tharon@hubcap.clemson.edu), about whether a real usability engineer "did" windows. Usability engineers were supposed to do usability testing and low-fidelity prototyping, not screen designs. But John S. Rhodes in his web article, "Rage Against the Machine" (webwork.com/moving/against.html) says that usability engineers should not do visual design, marketing, or programming. They should only test. What do you think about that?

SUSAN: I remember that debate. I think Rhodes makes a good point—that the usability engineers have to keep up their research skills. However, *somebody* has to do the windows and the navigation. And, to be honest, after you decide on a color scheme and the icons, window design is not a visually creative job. Instead, it's intellectually creative: "This user can't find an option that's right in front of her—what's going on? Is it a perceptual issue or an interaction problem? How do we change that?"

Keep in mind that, since the Information Revolution is only a few decades old, we're still in terra incognita. There are plenty of problems to solve, and the best way to solve any problem is to bring as many types of peoples and disciplines into the discussion as possible.

You know, a few years ago, there was a similar debate on UTEST, the online usability discussion group managed by Dr. Tharon Howard (tharon@hubcap.clemson.edu). I wondered at the time how it was going to turn out, and I think we know now. At the UPA 2000 conference, there was a workshop about writing requirements. Among the thirty attendees, there was absolutely no question that usability engineers did the window designs as well as the testing.



What would you like to see happen in the CHI community to increase awareness of principles and guidelines?

VICTOR: Nothing. I think there are too many guidelines and principles, half of them contradictory. What's the old saying? "It's great having standards—there are so many to choose from"?

But your books ARE guidelines—am I missing something here?

VICTOR: You're right, the books do contain guidelines. But those guidelines are only summaries of the research we did about each of the widgets—little tips of icebergs done up in paper. And although we're pretty sure that our guidelines are correct, we tried to include enough information that readers could draw their own conclusions. Guidelines without backup are often unconvincing, even when they're right.

Do you think that the platform-specific and technology-specific guidelines are too difficult to follow?

SUSAN: There's an interesting article by Henrik Thovtrup and Jakob Nielsen about the usability of standards online at www.useit.com/papers/standards.html. One key bit of information is that the examples were more influential than the actual specifications. In other words, get the pictures right.

What about enforcing guidelines? There doesn't seem to be any way to enforce a guideline, other than through cooperation in the development community and a sense of commitment to the user to not have to learn new ways of doing things every time they use a new piece of software.

SUSAN: There's actually an easy, time-honored method for enforcing guidelines and standards: hire an editor. Magazine and newspaper copy-editors make sure that every issue reads the same as every other. The pages always look the same and use the same colors. All abbreviations and titles are done the same. It's an honorable profession that requires a certain type of personality (obsessive-compulsive) and a good eye for detail. Why we ask developers to edit their own work when we can hire trained professionals, I don't understand.

But I realize my views are unpopular (no one has ever agreed with me, anyway). So if we can't have editors, then I want more research and better dissemination of the existing research. One of the reasons I like both Jared Spool (User Interface Engineering, UIE) and Jakob Nielsen (Nielsen Norman Group) is that they share what they know. I realize that people in the field criticize one or the other, saying that their methodologies are wrong or incomplete, but I don't care. At least they tell us what happened.

Spool's and Nielsen's results are surprising—counter-intuitive, even disheartening at times. For example, Nielsen wrote that people have stopped updating their browsers and computers as quickly as they used to—they really don't care about getting the latest version of Flash or QuickTime, not if they end up having to upgrade memory, free up disk space, get scolded by their IT departments about getting out of sync with the rest of the corporation, and so on. What this means is that, as designers, we will probably have to keep writing for Netscape version 4 and Internet Explorer version 4 until 2003 (see www.useit.com/alertbox/990418.html).

VICTOR: Here's another example of a surprise: Spool gave a talk last spring about the results of UIE's study on Web site access speed. People thought that a fast site, About.com, was slow and a slow site, Amazon.com, was fast. The difference, they discovered, was that people managed to buy things on Amazon, but often failed at About.com. This accurate perception of usefulness translated into an inaccurate perception of download speed.

Are you suggesting that practitioners ought to publish their designs and their rationales *and that researchers ought to include pointers in their publications to practitioners, so that we know what the design implications of some very esoteric research actually are?*

VICTOR: Yes, exactly. Thank you.

Sharing results is clearly important. Why don't you think it's done?

SUSAN: A few reasons. The first is that the last thing most professionals want to do after a hard day's work is go home and write some more. The second, though, is that a lot of the information is seen as proprietary. Companies don't want to give away their trade secrets, but many won't take the time to figure out what's a secret and what isn't, thereby effectively creating a chill on publishing. For example, details of an interface for an as-yet-unreleased product might indicate the company's future plans and therefore could be seen as a trade secret. Once the product is released, however, it's no longer a secret. But instead of giving employees a clear set of guidelines and a reasonable review plan, company lawyers too often imply that everything is a secret and if an employee wants to publish, it's up to him or her to make the case otherwise. Most people don't have the energy for that. So, in the current atmosphere, academics help the field immensely by doing the broad-based studies. For example, Ben Schneiderman and his colleagues and students at University of Maryland do wonderful work.

But academics (all of us, really) still think too narrowly. We eat, breathe, and live surrounded by software. Almost everything we do is touched by software—our cars run on software, our phone systems are all software, there are chips in our microwaves and our watches, many of us spend hours and hours in front of computers. The air is so full of electronic buzz that I'm surprised that the birds don't just fall out of the sky.

VICTOR: Despite all that, who pays attention to software interfaces? Where is the Jane Jacobs (*The Death and Life of Great American Cities*) who will criticize our structures? Where is the Hunter S. Thompson who will savage our games and entertainments? Where are the Siskels and Eberts who will give our projects thumbs up or thumbs down? We criticize each other's work, that's true, but why hasn't anyone else got into the act? It still seems to me that people think software is "out there," "nothing to do with me," "too technical," and worst of all, "bor-r-r-ing."

SUSAN: Of course, we may just be looking in the wrong places. They may be out there already, doing their own webzines or "blog" sites (www.blogger.com/about.pyra). The people with the really new ideas often self-publish years before the mainstream is ready to hear from them.

Don Norman once told an audience that the only way you can become a good designer is through experience. What do you think about that?

VICTOR: If that were really true, there would be no point in having schools. (He was a teacher for a long time, wasn't he?) But he has a point. Many times when I design a new web page, I find out that a design rule I believe in with my whole heart and soul isn't right: I try something and it JUST DOESN'T WORK. I hate that.

SUSAN: Here's another way in which he's right. A few years ago, researchers looked at the effect of expertise on short-term memory by showing a mid-play chessboard to two sets of chess-players and asking them to reconstruct the positions of the pieces from memory. The novice players needed to check the board seven times on average, but the master players checked the board only four times. What made the master players "smarter" than the novice players wasn't that they had better memories but that they could chunk the board—in other words, they were able to recognize a pattern in the way the pieces were set up. They remembered the entire pattern, not the individual pieces. (Interestingly, when the pieces were set up randomly, the master players did worse than the novices did.)

This is why Christopher Alexander's patterns (*A Pattern Language,* Oxford University Press, 1977, ISBN 0195019199) have become so interesting to both software and usability engineers. One of his ideas is that anyone with enough experience can recognize and describe patterns accurately enough to make them accessible to non-experts.

VICTOR: One of my favorite Alexandrian patterns is "desire lines." These are the lines in the grass that people make walking between buildings on university or business campuses. Smart architects wait for those lines to appear and then pave them.

Another type of pattern is land surveyor marks, which are the lines, crosses, and arrows you sometimes see cut into the sidewalks. At a glance, any other land surveyor can read a dozen things about the area—which way is north, where the property lines are, whether the property is being sold or subdivided, and so on.

Alexander's work in architecture has inspired further reflection on patterns as well as anti-patterns in diverse fields (a particularly accessible volume is Brown et al.'s AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis (Wiley Computer Publishing, 1998, ISBN 0-471-19713-0*). Can you give us an interface design pattern example?*

SUSAN: In interface design, one common pattern is the "too many" problem. In very large databases, there has to be some way to scope the view *before* the users open a list. Otherwise, the list can take forever to load. Developers and usability engineers have come up with a variety of solutions—for example, load a page's worth of the list, then wait for the user to ask for the next page; make the user fill in search fields first, before he or she can do anything else; put up a message that says, "This request will load the entire database. Are you sure you want to do this?"; and so on.

Any experienced engineer, when faced with this problem, automatically offers one of those solutions. You can recognize a master engineer, however, because she'll propose two or three alternatives, not just the one she used in her last project.

For more interface-design patterns, try starting with the pattern gallery managed by Sally A. Fincher at www.cs.ukc.ac.uk/people/staff/saf/patterns/gallery.html.

What books do you keep handy on your bookshelf?

VICTOR: Jared Spool's *Web Site Usability* (Morgan Kaufmann Publishers, 1999, ISBN 1-55860569X), Jakob Nielsen's *Designing Web Usability: The Practice of Simplicity* (New Riders Publishing, 1999, ISBN 156205810X), *Dreamweaver® 3 Bible* by Joseph W. Lowery and Paul Madar (IDG Books Worldwide, 2000, ISBN 0764534580), and the *CorelDraw* users guides.

I also like both of Scott McCloud's books, *Understanding Comics* (Kitchen Sink Press, 1994, ISBN 006097625X) and *Reinventing Comics* (Harper Perennial Library, 2000, ISBN 0060953500).

SUSAN: Within arm's reach, I have *Microsoft Windows User Experience* (Microsoft Press, 1999, ISBN 0735605661), *Java Look and Feel Design Guidelines*, (Addison-Wesley Publishing Co., 1999, ISBN 0201615851), Carl Zetie's *Practical User Interface Design: Making GUIs Work* (McGraw-Hill, 1995, ISBN 0077091671), Jeff Johnson's *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers* (Morgan Kaufmann Publishers, 2000, ISBN 1558605827), and my annotated copy of the *GUI Design Handbook*.

However, for the really sticky problems, I also keep books by Eric von Hippel and R. Meredith Belbin nearby. Together, they explain why certain businesses and teams succeed and why others, alas, fail. Their books are based on years of research, and like Spool and Nielsen, the results are often surprising.

Von Hippel, professor at MIT's Sloan School of Management discovered that many lucrative products aren't invented by the companies who sell them but rather by their customers, and that the most successful of those companies are the ones with the most open interfaces. He also discovered which customers are most likely to innovate (the "lead users," who recognize and solve a problem earlier than other organizations) and how to find them. You can find out more in *The Sources of Innovation* (Oxford University Press, 1988, ISBN 0-19-509422-0) and *Breakthrough Products and Services with Lead User Research* (Lead User Concepts, Cambridge, MA, 1998, with co-authors Joan Churchill and Mary Sonnack).

Belbin, who has been running experiments on teams for 30 years in England and Australia, discovered that teams are successful when they have the right mix of *team* roles—the right mix of *job* roles isn't enough. He discovered the team roles by setting up business games for executives taking his workshops and seeing which combinations of people did best. The categories he discovered are plant (creative, unorthodox), resource investigator (extroverted, sociable, a fixer), coordinator (mature, confident, a natural chairperson), shaper (thrives on pressure, confident, challenging), monitor evaluator (sober, strategic, sees all options), teamworker (cooperative, reduces friction), implementer (good at follow-through), completer (conscientious, anxious), specialist (the engineer type).

When a team is overloaded in one area or another, nothing gets done. In one hilarious experiment, his institute created a team composed only of shapers and set them to work in competition with four or five other teams. They came in last. Each member of the team thought his ideas were best and none would compromise with any of the others to solve the problem.

So if you've ever had trouble working in a team, you'll want to read his books. I recommend *Management Teams* (Butterworth Heinemann, 1981, ISBN 0750626763) and *The Coming Shape of Organization* (Butterworth Heinemann, 1996, ISBN 0750639504). Video Arts also offers two videotapes based on his work, *Building the Perfect Team and Selecting the Perfect Team*. (The Video Arts phone number is 1-800-553-0091; www.video-arts.com.)

If you've picked the right mentors, you'll know it because they'll make you furious and jealous and stretch you farther than you could ever stretch yourself.

Do you have any UI heroes? Villains?

SUSAN: No villains. Heroes, yes. Besides the people I've already mentioned, I have many heroes who aren't as well known.

For example?

SUSAN: Donna Timpone and Gary Haughney, principals of UserEdge, Inc., in New Jersey. Donna has been teaching usability all over North America for many years and has helped many technical communicators move into usability. Even better, UserEdge finds jobs for them through their very ethical and professional placement firm (see www.useredge.com).

Another hero of mine is Chauncey Wilson, who shares everything he knows with anyone who asks for help. He's a mainstay of the UTEST list, and if you're not on UTEST, you can access his usability bibliography at world.std.com/~uieweb/biblio.htm. I also have a lot of respect for the people I work with at Telcordia Technologies. If I mention only one of them—Merrill Klier—it's just to give you a flavor of what I mean.

Merrill seems to know instinctively what will interest users (although it's probably not instinct but years of experience). At a workshop a few months ago, I watched her wake up twenty jet-lagged, slightly grumpy users with an average of 25 years experience each by saying, "You have a new employee. He's 20 years old, right off the street, doesn't know your business. So he comes in and what does he see? What kind of GUI is it? Is it a list like the ones you use? Or is it a map? " She hit on what was bothering them: That the experienced people like themselves are retiring and they're being replaced by smart kids whose idea of a computer interface is Nintendo and the web. Because she had the courage to challenge them, we came away with dozens of new ideas and, more importantly, got confirmation for a visual approach that everyone had been uncertain about before.

VICTOR: I would add Valerie Fenster, UserlWare (www. userware.com), and Sylvia Woodard, Two Twelve Associates (www.twotwelve.com/) in New York City. They led the design process for Citibank's ADA automatic teller machine, which is probably the most elegant ATM in the U.S. ("ADA" is the Americans with Disabilities Act). The ADA ATM had to be usable by blind and partially sighted people, and what made this tricky is that Citibank has only touch-screens—there are no buttons to stick Braille labels on. How they came up with the right design is a great story. You can read about it in ID magazine, July/August 1994, pp. 69-71.

Here's my last question. What advice do you have for beginners?

SUSAN: Pick the right mentors and the right problems. If you've picked the right mentors, you'll know it because they'll make you furious and jealous and stretch you farther than you could ever stretch yourself.

VICTOR: As for the right problems, Shneiderman said it best in your interview with him last year: It's important to have a problem that really matters, not a problem you might come up with on your own. Pick something that you're passionate about, and find someone who is willing to pay you to design it. Rule of thumb: If no one will pay for it, then it's not an important problem.

✳ ✳ ✳ ✳ ✳ ✳ ✳

Victor Stanwick *and* **Susan Fowler** *are currently working on their next book, tentatively titled* Web Application Design Handbook: How to Design Applications for the Web. *To see—and to add to—the work in progress, go to their web site, www.fast-consulting.com/web-book.htm.*