For years and years, self-fashioned pundits, like, uh, me, have been nattering endlessly about usability, and how important it is to make software usable. Jakob Nielsen has a mathematical formula he'll reveal to you in exchange for $122 which you can use to calculate the value of usability. (If the expected value of usability is greater than $122, I guess you make a profit.)

I have a book you can buy for a lot less that tells you some of the principles of designing usable software, but there's no math involved, and you'll be out the price of the book.

In that book, on page 31, I showed an example from what was, at the time, the most popular software application on Earth, Napster. The main Napster window used buttons to flip between the five screens. Due to a principle in usability called affordance, instead of buttons, it really should have had tabs, which was the point I was trying to make.

And yet, Napster was the most popular software application on Earth.

In an early version of the manuscript, I actually wrote something like "this just goes to show you that usability ain't all that important," which was an odd thing to be writing in a book about usability. I was sort of relieved when the typesetter announced that I had to shorten that paragraph. So I deleted the sentence.

But there's a scary element of truth to it—scary to UI professionals, at least: an application that does something really great that people really want to do can be pathetically unusable, and it will still be a hit. And an application can be the easiest thing in the world to use, but if it doesn't do anything anybody wants, it will flop. UI consultants are constantly on the defensive, working up improbable ROI formulas about the return on investment clients will get from their $75,000 usability project, precisely because usability is perceived as "optional," and the scary thing is, in a lot of cases, it is. In a lot of cases. The CNN website has nothing to be gained from a usability consultant. I'll go out on a limb and say that there is not a single content-based website online that
Joel on Software - It's Not Just Usability

would gain even one dollar in revenue by improving usability, because content-based websites (by which I mean, websites that are not also applications) are already so damn usable.

Anyway.

My goal today is not to whine about how usability is not important... usability is important at the margins, and there a lots of examples where bad usability kills people in small planes, creates famine and pestilence, etc.

My goal today is to talk about the next level of software design issues, after you've got the UI right: designing the social interface.

I need to explain that, I guess.

Software in the 1980s, when usability was "invented," was all about computer-human interaction. A lot of software still is. But the Internet brings us a new kind of software: software that's about human-human interaction.

Discussion groups. Social networking. Online classifieds. Oh, and, uh, email. It's all software that mediates between people, not between the human and the computer.

When you're writing software that mediates between people, after you get the usability right, you have to get the social interface right. And the social interface is more important. The best UI in the world won't save software with an awkward social interface.

The best way to illustrate social interfaces is with a few examples of failures and successes.

Some Examples

First, a failing social interface. Every week I get an email from somebody I've never heard of asking me to become a part of his or her social network. I usually don't know the person, so I feel a little bit miffed and delete the message. Someone told me why this happens: one of those social networking software companies has a tool that goes through your email address book and sends email to everyone asking them to join in. Now, combine this with the feature that some email software saves the sender's address of every incoming message, and the feature that when you go to sign up for the Joel on Software email bulletin you get a confirmation message asking if you really want to join, and voila: all kinds of people who I don't know are running software that is inadvertently asking me to confirm that I'm their friend. Thank you for subscribing to my newsletter, but no, I'm not going to introduce you to Bill Gates. I currently have a policy of not joining any of these social networks, because they strike me as going strongly against the grain of
how human networks really work.

Now, let's look at a successful social interface. Many humans are less inhibited when they're typing than when they are speaking face-to-face. Teenagers are less shy. With cellphone text messages, they're more likely to ask each other out on dates. That genre of software was so successful socially that it's radically improving millions of people's love lives (or at least their social calendars). Even though text messaging has a ghastly user interface, it became extremely popular with the kids. The joke of it is that there's a much better user interface built into every cellphone for human to human communication: this clever thing called "phone calls." You dial a number after which everything you say can be heard by the other person, and vice versa. It's that simple. But it's not as popular in some circles as this awkward system where you break your thumbs typing huge strings of numbers just to say "damn you're hot," because that string of numbers gets you a date, and you would never have the guts to say "damn you're hot" using your larynx.

Another social software success is ebay. When I first heard about ebay, I said, "Nonsense! That will never work. Nobody's going to send money to some random person they encountered on the Internet in hopes that person will out of the goodness of their hearts actually ship them some merchandise." A lot of people thought this. We were all wrong. Wrong, wrong, wrong. Ebay made a big bet on the cultural anthropology of human beings and won. The great thing about ebay is that it was a huge success precisely because it seemed like a terrible idea at the time, and so nobody else tried it, until ebay locked in the network effects and first-mover advantage.

In addition to absolute success and failures in social software, there are also social software side effects. The way social software behaves determines a huge amount about the type of community that develops. Usenet clients have this big-R command which is used to reply to a message while quoting the original message with those elegant '>'s in the left column. And the early newsreaders were not threaded, so if you wanted to respond to someone's point coherently, you had to quote them using the big-R feature. This led to a particularly Usenet style of responding to an argument: the line-by-line nitpick. It's fun for the nitpicker but never worth reading. (By the way, the political bloggers, newcomers to the Internet, have reinvented this technique, thinking they were discovering something fun and new, and called it fisking, for reasons I won't go into. Don't worry, it's not dirty.) Even though human beings had been debating for centuries, a tiny feature of a software product produced a whole new style of debating.

Small changes in software can make big differences in the way that software supports, or fails to support, its social goals. Danah Boyd has a great critique of social software networks, Autistic Social Software, blasting the current generation of this software for forcing people to behave autistically:
Consider, for a moment, the recent surge of interest in articulated social networks such as Friendster, Tribe, LinkedIn, Orkut and the like. These technologies attempt to formalize how people should construct and manage their relationships. They assume that you can rate your friends. In some cases, they procedurally direct how people can engage with new people by giving you an absolute process through which you can contact others.

While this approach certainly has its merits because it is computationally possible, I'm terrified when people think that this models social life. It's so simplistic that people are forced to engage as though they have autism, as though they must interact procedurally. This approach certainly aids people who need that kind of systematization, but it is not a model that universally makes sense. Furthermore, what are the implications of having technology prescribe mechanistic engagement? Do we really want a social life that encourages autistic interactions?

When software implements social interfaces while disregarding cultural anthropology, it's creepy and awkward and doesn't really work.

**Designing Social Software**

Let me give you an example of social interface design.

Suppose your user does something they shouldn't have done.

Good usability design says that you should tell them what they did wrong, and tell them how to correct it. Usability consultants are marketing this under the brand name "Defensive Design."

When you're working on social software, this is too naive.

Maybe the thing that they did wrong was to post an advertisement for Viagra on a discussion group.

Now you tell them, "Sorry, Viagra is not a suitable topic. Your post has been rejected."

Guess what they'll do? They'll post an advertisement for V1agra. (Either that, or they'll launch into a long boring rant about censorship and the First Amendment.)

With social interface engineering, you have to look at sociology and anthropology. In societies, there are freeloaders, scammers, and other miscreants. In social software, there will be people who try to abuse the software for their own profit at the expense of the rest of the society. Unchecked, this leads to something economists call *the tragedy of the commons*. 
Whereas the goal of user interface design is to help the user succeed, the goal of social interface design is to help the society succeed, even if it means one user has to fail.

So a good social interface designer might say, let's not display an error message. Let's just pretend that the post about Viagra was accepted. Show it to the original poster, so he feels smug and moves on to the next inappropriate discussion group. But don't show it to anyone else.

Indeed one of the best ways to deflect attacks is to make it look like they're succeeding. It's the software equivalent of playing dead.

No, it doesn't work 100% of the time. It works 95% of the time, and it reduces the problems you'll have twenty-fold. Like everything else in sociology, it's a fuzzy heuristic. It kind of works a lot of the time, so it's worth doing, even if it's not foolproof. The Russian mafia with their phishing schemes will eventually work around it. The idiot Floridians in trailer parks trying to get rich quick will move on. 90% of the spam I get today is still so hopelessly naive about spam filters that it would even get caught by the pathetic junk filter built into Microsoft Outlook, and you've got to have really lame spam to get caught by that scrawny smattering of simplistic searchphrases.

Marketing Social Interfaces

A few months ago I realized that a common theme in the software we've built at Fog Creek is an almost obsessive attention to getting the social interfaces right. For example, FogBugz has lots of features, and even more non-features, designed to make bug tracking actually happen. Time and time again customers tell me that their old bug tracking software was never getting used, because it did not align with the way people wanted to work together, but when they rolled out FogBugz, people actually starting using it, and became addicted to it, and it changed the way they worked together. I know that FogBugz works because we have a very high upgrade rate when there's a new version, which tells me FogBugz is not just shelfware, and because even customers who buy huge blocks of licenses keep coming back for more user licenses as the product spreads around their organization and really gets used. This is something I'm really proud of. Software used in teams usually fails to take hold, because it requires everyone on the team to change the way they work simultaneously, something which anthropologists will tell you is vanishingly unlikely. For that reason FogBugz has lots of design decisions which make it useful even for a single person on a team, and lots of design features which encourage it to spread to other members of the team gradually until everyone is using it.

The discussion group software used on this site, which will soon be for sale as a feature of
FogBugz 4.0, is even more obsessed with getting the social interface aspects exactly right. There are dozens of features and non-features and design decisions which collectively lead to a very high level of interesting conversation with the best signal-to-noise ratio of any discussion group I've ever participated in. I wrote a lot about this in my article Building Communities with Software.

Since then, I've become even more devoted to the idea of the value of good social interface design: we bring in experts like Clay Shirky (a pioneer in the field), we do bold experiments on the poor citizens of the Joel on Software discussion group (many of which are so subtle as to be virtually unnoticeable, for example, the fact that we don't show you the post you're replying to while you type your reply in hopes of cutting down quoting, which makes it easier to read a thread), and we're investing heavily in advanced algorithms to reduce discussion group spam.

A New Field

Social interface design is still a field in its infancy. I'm not aware of any books on the subject; there are only a few people working in the research side of the field, and there's no organized science of social interface design. In the early days of usability design, software companies recruited ergonomics experts and human factors experts to help design usable products. Ergonomics experts knew a lot about the right height for a desk, but they didn't know how to design GUIs for file systems, so a new field arose. Eventually the new discipline of user interface design came into its own, and figured out the concepts like consistency, affordability, feedback, etc., which became the cornerstone of the science of UI design.

Over the next decade, I expect that software companies will hire people trained as anthropologists and ethnographers to work on social interface design. Instead of building usability labs, they'll go out into the field and write ethnographies. And hopefully, we'll figure out the new principles of social interface design. It's going to be fascinating... as fun as user interface design was in the 1980s... so stay tuned.

Discuss this topic on the Joel on Software Social Interface Design Forum.