Simplifying Complexity, Part 2: Cognitive Load and Mental Models

In the last article, we looked at the lure of simple software and the reality that software is complex, and often for good reasons. In this article, we will begin our look at some of the techniques that user interface designers can use to deal with complexity. We'll start by looking at the big picture, at some high-level design approaches. Then, in Parts 3 and 4 we will turn to some specific design techniques.

Your Deal with the User

The previous article mentioned Adobe Photoshop as an example of complex software where Adobe has entered into a kind of implied deal with the user: if you invest the time to master Photoshop, you'll be rewarded by being able to use one of the most powerful image processing tools available. Judging by the success of Photoshop over the years, many users think that's a fair trade.

You are always entering into a deal like this with your users. They are investing time and (often) money in your product with the expectation of some payoff. Make sure this deal is worth it to them. Photoshop users are willing to invest a lot of time in the product; they watch online videos, ask (and answer) questions on message boards, buy Photoshop books, and attend seminars. But the threshold for the effort the user is willing to commit is often much lower. Have you ever abandoned filling out an online form if you got one too many error messages (wrong format for dates, phone numbers, or zip codes, for example)? Downloaded free software then uninstalled it because you couldn't figure out how to use it after a few minutes of poking around?

We only have so much attention we're willing to devote to a task. If you're playing in the Super Bowl, teaching your child to read, or perhaps learning Photoshop to accomplish something important, you're probably willing to devote a lot of attention. If you're responding to an unsolicited user survey or trying out an app you downloaded on a whim, you probably aren't willing to devote as much attention.

This is another way of saying that, depending on the situation, users are willing to accept some complexity. One of your first tasks as a designer should be to understand the environment in which your product will be used. How much complexity does the business case demand? How much complexity will users tolerate and what motivates them? Knowing the answers to these questions gives you the framework to best apply the techniques we are going to discuss in the rest of this series.

Minimizing Cognitive Load

Whatever level of effort a user is willing to devote, you want to help them by minimizing the demands placed on them. Simplify what you can so that the user can focus their energy on what remains complex.

This is called minimizing cognitive load -- "the total amount of mental effort being used in the working memory."[1]

Here's an example. If you are a United States resident you are of course used to driving on the right-hand side of the road. If you vacation in the United Kingdom, you will need to drive on the left-hand side. This requires a lot of mental effort, because you need to behave counter to a well-established habit and the steering wheel is also on the “wrong” side of the car. In other words, there's a substantial cognitive load in overcoming your past significant training. Now assume that, in addition to driving on "wrong" side of the road, you have to do so in a vehicle with a manual transmission, and you are used to driving an automatic. The cognitive load is now even higher, and successful completion of the task becomes even more difficult.

One way for software designers to minimize cognitive load is to be consistent. If something works one way or means one thing in one place, then it should work the same way or mean the same thing everyplace.

- Following established user interface guidelines and standards is a fundamental way of being consistent. If your buttons and drop downs and windows work like everyone else's, users don't need to figure out the basics of your interface and can focus on other issues.
- Consistency is also important within your application. Inconsistent applications can frustrate users. For example, in Quicken you can right-click on an account transaction in the transaction list to get a pop-up menu, but no such menu exists for an account in the account list (you have to click an edit button, which in turn differs from the gear icons and right arrow icons used elsewhere to invoke a pop-up menu).
- It's also important to be consistent with established ways of accomplishing tasks. For example, online shopping sites have shopping carts, as we all know how shopping carts work. So that concept is one less thing the user needs to figure out.

Mental Models

A shopping cart is a good example of what is called a mental model. Mental models are important. When we you first encountered an online shopping cart, you probably had a pretty good idea of what to expect -- it was a place to put your items before you actually purchased them. You understood that it was like a shopping cart in the physical world. That understanding gave you a good starting point for learning how online shopping carts work.

UI designers frequently rely on mental models because they leverage knowledge users already have. Photoshop, for example, has layers. A Photoshop image can consist of a stack of layers, each containing different components and edits to the image. The visible result is the view from the top layer down, like

[1] https://en.wikipedia.org/wiki/Cognitive_load, 29 April 2015

a stack of transparencies. Layers are a useful tool for organizing edits, and the concept is easy to understand because of the straightforward analogy to the physical world.

However, Adobe has complicated the layer concept by introducing variations on the layer -- adjustment layers, layer effects, layer masks, blend modes, and so on. Here, the mental model breaks down, because these things don't have corresponding analogies in the real world. But at least users understand what the basic function of a layer as they tackle the cognitive load of learning what these additional variants do.

As another example, Apple normally does a good job leveraging mental models. However, iOS 7 dealt with photos in a somewhat non-intuitive way. Photos appeared in the Camera Roll as well as the Photo Stream, and it was not immediately clear how they differed (the Photostream syncs with iCloud). There was no mental model to anchor our understanding of how they differed regarding where photos went after you took them. It's a rare Apple misstep. David Lieb, writing on TechCrunch in 2013, said, "iCloud is rife with cognitive overhead — it only backs up your most recent photos, it works on certain select apps but not others, you have to create an icloud.com email account for it to sync your mail and notes but not everything else. Oh, and it works on new iPhone and iPads and Macs running OS X v10.7.4 or later, but not your PC or Android tablet."[2]

Summary

You don't have to look far to find examples of the consequences of failing to minimizing cognitive load and leverage mental models. In 2011, Microsoft encountered plenty of controversy when they released Windows 8, which featured a brand new Metro interface. The Metro interface was optimized for tablets, so Microsoft had a good business reason for going in that direction. The problem was that, in doing so, they introduced so many inconsistencies and conflicts with their users' existing mental model of Windows that David Pogue, writing in the *New York Times*, went so far as to say that the learning curve "resembles Mount Everest." [3] Many users did not find Windows 8 to be simple.

Parts 3 and 4 will look at some powerful design techniques that can easily be used for keeping the complex simple.

Dr. Craig Rosenberg is an entrepreneur, human factors engineer, computer scientist, and expert witness. You can learn more about Craig and his expert witness consulting business at www.userinterfaceexpertwitness.com

[2] http://techcrunch.com/2013/04/20/cognitive-overhead/

[3] http://www.nytimes.com/2012/10/25/technology/personaltech/microsofts-windows-revamped-and-split-in-2.html

Craig Rosenberg, Ph.D.
www.userinterfaceexpertwitness.com
craig@globaltechnica.com
206-552-9898