#12 - The art of UI prototyping

By Scott Berkun, November 2000

Even the brightest people make mistakes. This is especially true for teams of people. Somehow, as a project moves forward, small assumptions and well-intentioned but poor decisions accumulate, turning hours of work into a lousy user experience. The smart teams eliminate their mistakes before they ship by using a technique called UI prototyping. Combined with usability studies, prototypes keep teams headed in the right direction.

Why Prototype?

Prototyping is a means of exploring ideas before you invest in them. All experienced craftspeople and engineers create prototypes of their work before they build anything: Architects create models out of paper or cardboard, or with virtual reality tools. Aeronautic engineers use wind tunnels. Bridge builders create stress models. Software and Web designers create mock-ups of how users will interact with their designs.

The best reason to prototype is to save time and resources. The value of the prototype is that it is a facade—like a Hollywood set, where only the front of the building is constructed. Relative to the real product, prototypes are easy and inexpensive to create. So, for a minimal investment, you can find usability and design problems and adjust your UI before you invest heavily in the final design and technologies.

On examining the needs of your particular project, you might come up with reasons for creating a prototype other than saving money. Is the goal to explore a new interface model? Make modifications to one part of the existing design? Investigate a new technology? It's important to be clear about why you're building what you're building before you start. If you begin with clear goals, you can be direct and effective in your efforts. The reasons for creating prototypes fall into three basic categories:

1. **Proof of concept.** Among some teams there are disagreements about the future direction of a project. You can use a prototype to prove that an idea or new approach has merit or value. A prototype can help illustrate that an idea works, express its qualities in a visual and interactive way, and/or motivate team members to think about the problem from another perspective.
2. **Design exploration.** If you design interactive things, the only way to explore how something will be used is to create a mock-up and interact with it. Sometimes the mock-up is tied to a usability study, where parts of the prototype can be evaluated in a structured way. Sometimes it's just a way to clearly express to a developer how something should work or look. In many cases, a designer might simply be experimenting, in an effort to get a sense for what approach might work. Anyone on the team can use prototypes to explore design issues, although designers are generally the most skilled. Design explorations should be directed at trying to solve specific problems that you've recognized in your product.
3. **Technical exploration.** Developers investigating implementation approaches to a problem often try out different coding techniques to see if they work well. Using HTML, Jscript, SQL, DHTML, Win32, or specific coding approaches within each technology have different tradeoffs. Sometimes a prototype represents an exploration into what technology will work well to support a certain UI or web feature.

Sometimes prototypes are created for a combination of these reasons. If a team plans well enough, they can allot time for a developer and a designer or project manager to work together on a prototype. In such cases, it's

extremely important to clearly define the goals and the contributions each team member is expected to make. You want everyone to be clear on what the goals are, what's at stake, and what the potential outcome will be.

Who Is Involved?

Prototyping can be done informally by anyone, regardless of their background or role in the project. It's easy to make a simple but effective prototype by taking a bitmap from Adobe Photoshop, putting it into the Microsoft® FrontPage® Web site creation and management tool, and adding active buttons or links. These lightweight prototypes only go so far, and can become unwieldy for complex interactions.

For more formal prototypes by larger teams, a developer or project manager will often collaborate with a designer and a usability engineer. Together they'll generate ideas, build a prototype that represents the best ideas, and then go into the lab to see how effective it is in solving user problems. Developers might get involved if they can spare the time, or if their technical expertise is needed. Often the designer or project manager will do most of the scripting or coding to build the prototype.

When Do You Build a Prototype?

Depending on the scope of the prototype and the level of detail required, prototypes can be built at any time during the project. Most often they are created early in the project, during the planning and specification phase, before developers write any production code. That's when the need for exploration is greatest, and when the time investment needed is most viable. If developers instead of program managers or designers are prototyping, scheduling time becomes even more important because you need to make sure the work invested in the prototype is accounted for in the development schedule. Planning for any UI release should include some level of prototyping.

Late in a project, smaller prototypes can help resolve tough design and technical issues. A quick HTML mock-up of how a dialog box or Web page should behave can help answer a developer's question or give other teammates a feel for what the desired experience should be. In some cases, a strong program manager or designer can implement the behavior in Microsoft JScript® development software and approximate much of the programming logic that developers will need to think through.

The time it takes to create a prototype can vary tremendously, depending on the scope and precision of what the end result needs to look like. An informal prototype could mean a few hours of work by one person; a more organized effort can involve weeks of effort by an entire team.

How Far Should You Go?

In your prototype, build only as much of the design as you need. It's okay to have buttons that don't work, or text that never updates. As long as you can experience the interactions you want to explore, it's fine to use smoke and mirrors for the rest. Here are a few reasons why you should focus your efforts carefully:

- **Cost of building the prototype.** You want to minimize the cost involved in building the prototype. The challenge with prototyping is recognizing the minimal investment needed to effectively answer your questions about the design. This is where usability studies are critical, because they clearly identify the parts of your UI that need the most work. Even without usability studies, you should clearly define what user problems you're trying to solve, or what tasks you're trying to improve, with the design in your prototype.
- **Limited lifetime.** Prototypes should have clearly defined lifetimes. Is the end goal a presentation at a team meeting? An

executive review meeting? A spec review? A usability study? Convincing yourself, with your devil's advocate hat on, that the design solves a user problem? Once the needs for these specific objectives are met, the prototype should be set aside. The basic mindset is that the code or bitmaps generated in a prototype will be left behind. There might be exceptions where code or visuals live on in the product, but the expectation should be that this won't be the case.

- **Risk of overwhelming the team.** Showing prototypes to developers and teammates can be tricky. An overly complex or elaborate prototype, sporting amazing visuals and animation, can overwhelm people. You should always have a sense for how far to go and how much of what you're creating in the prototype you want to be taken seriously.

Determining the Scope of Your Prototype

As you determine where to focus your prototyping efforts, here are some things to consider:

- **Customer needs.** If you start with an understanding of the key problems or needs of your users (perhaps something your usability engineer has provided for you), then you have an idea about which parts warrant the most exploration.
- **Usability study tasks.** If you are creating the prototype for a usability study, discuss with the usability engineer what specific tasks will be part of the study, and design around those elements.
- **Team input.** Talk with key developers on your team as the ideas in your prototype are coming together. Get a basic sense from them on what's reasonable, what's possible, and what is beyond consideration for the next release. In some cases, you might deliberately go beyond what they say is possible for one aspect of the design if it's a key point and you think the team needs to be challenged. However, you don't want to do this with every aspect of your prototype. There is a fine line between pushing the limits and overwhelming your team. If you only want to inspire the team by showing them a vision for several versions out, then go for it. However, if you're looking to define specific changes for the next release, then focus your efforts on those changes. Make sure you call out the specific changes in a modular way to show developers a path for building your designs.
- **Breadth vs. depth.** For larger prototypes, there is the additional consideration of breadth versus depth. Do you make each feature in the design work just a little bit, or do you pick one feature and prototype almost all of its pieces and options? If you're not careful, you'll try to do both at the same time and end up with a large, unwieldy prototype that is hard to modify and difficult to throw away.
- **Wireframe vs. Visual design** - Depending on your audience, consider what level of visual design quality you need your prototype to have. Sketches or box drawings may suffice for folks you work with often, that you do not need to impress, or who can understand the difference between the prototype, and what it represents. If you are presenting to less experienced clients, certain executives, or more technical audiences, a more robust and aesthetically invested prototype might be appropriate. An additional consideration is what questions you have for the design itself - if you hope to learn about the impact of your aesthetic and layout choices on usability, and you are planning for a usability test, you need to make that additional investment.
- **ROI: Return on Investment** - Prototypes allow for various forms of evaluation (aesthetic, business, technical, usability). The

higher fidelity the prototype is, the greater accuracy your evaluation will have. The most robust prototypes require no explanation - you just point people at it, and let them experience it for themselves. The more you have to explain ("oh, that wouldn't do that in the real version", "the style will be more techno") the less robust it is, and the less accurate the evaluations are likely to be. Of course, you pay a price for the investments you make. Deciding how robust is enough, and how accurate an evaluation you need to make is a judgement call, much like the decision making processes of actual web sites or software products. Your goal is to invest as much as necessary to obtain the information and effects you want, but no more.

Making Prototypes Flexible

One way to focus your prototyping resources is to concentrate on smart design. You can create more effective prototypes by allowing one piece of prototype code to exercise many different ideas. Instead of having five different prototypes, consider making one prototype that has the options to switch the different attributes of the prototype.

Should the toolbar be located on the left or on the top? Should we show 10 items on the home page or 20? A good prototype has some sort of built-in options panel that allows you to change the parameters of how the prototype looks or works. Keep these option panels hidden in your prototype—you don't want a usability participant accidentally finding them during a test.

The challenge is to keep the prototype simple, but still useful enough that you can show it to a teammate, walk through some of the different options you're thinking about, and get feedback on them.

How Do Beta Releases Differ from Prototypes?

Beta releases don't qualify as prototypes, because they are complete engineering efforts. If you find a critical mistake in a feature of a beta release, you are unlikely to throw it away, even if that might be in the best interest of the product. The developers, testers, and designers have already invested their time, and the pressure to live with bad decisions is very high. Betas certainly do help in finding bugs and defects, but they are rarely useful in making controlled studies of the value of specific design directions.

Tools and Technologies

There are several different tools and technologies you can use for creating prototypes, each of which has its advantages and disadvantages. Consider the type of design work you're trying to prototype and the goals of your prototyping effort as you decide which tool or technology is right for you.

- **Paper -** For usability studies or quick reviews, paper is often the fastest way to prototype a design idea. Using Photoshop, mspaint, or any tool you are comfortable with, produce screens that express the design, and print them out on paper. If you make enough screens, you can simulate walkthroughs, allowing test users to make choices and experience the design. However, for prototypes of moderate complexity, generating paper prototypes can be cumbersome. Highly interactive things like games or chat rooms can not be simulated well on paper. Also, the more elaborate the tasks, the more pages you might need to have handy.
- **Microsoft Visual Basic—.** This is the fastest technology for creating Windows-style UI prototypes. The Web browser object makes it easy to integrate HTML UI with your standard Windows-style components. While it's true that an experienced C/C++ developer might be able to generate UI faster in C/C++, this creates the temptation to reuse code from the UI prototype

in your production code. It takes discipline to recognize that the goals of a quick and dirty UI prototype are highly divergent from high-quality engineering. Make sure you know what kind of code you're writing, and that your team or manager understands what will be discarded.

- **Macromedia Director or Flash.** This is one of the most popular UI prototyping tools among designers. It is most useful for multimedia or non-standard GUI designs, or for prototypes that require significant animation. It's high flexibility makes it cumbersome for Windows-style UI compared to Visual Basic. However, a proficient Director user can generate Windows or Web UI without difficultly.
- **HTML.** Dreamweaver, FrontPage and other HTML editors allow for fast creation of simple prototypes. To express an idea, you can often create bitmaps that illustrate a sequence of user interaction, and place them into FrontPage. Then you can create link areas to connect the pages, and simulate how you can interact with the design. JScript and DHTML take things to another level, allowing for very sophisticated logic and interaction. If you are using HTML to prototype your Web site, the warning just described for C/C++ applies here as well—don't fall into the trap of confusing quick prototype code with production-quality engineering.

Scott's first book, the art of project management, will be published by O'Reilly in April of 2005.

RSS Feed