Notes on Prototyping
By David Walker, Ph.D.

Why Use Prototyping?

Prototyping is a development methodology in which a model is quickly constructed to test or illustrate design features and ideas, in order to gather user feedback. Subsequent models are created by refining earlier versions, with the aim of convergence on the desired end product. This design methodology is sometimes known as evolutionary design or incremental development. Different versions of prototyping are used in software design, system development, and electronics engineering.

This document deals with prototyping as a methodology for developing web sites with a minimal need for custom programming. (Software prototyping is a more complex area, and the methodologies currently used are much less straightforwardly applicable.)

Prototyping has been widely used for at least 30-35 years. It gradually replaced the older ‘waterfall’ model. Like a series of waterfalls, this older method assumed a number of separate stages, each self-contained and finally flowing into the next stage. In this scenario users would first clearly state all of their requirements for the system, and then leave it to the systems analyst to design the complete system. Next the programmer would create the system code. Finally the testers would verify the correctness of the system and it would be presented to the users. Theoretically this is a tidy model (somewhat simplified here), but in the real world none of these stages were self-contained and often users were presented with systems that had little to do with their real requirements. Details were often forgotten at the outset, misunderstandings often arose during design, compromises were made during programming, and false assumptions were used during testing. Clearly, such a one-way process was less than ideal.

The waterfall model taught us that:

- there is rarely enough information available at the outset to design a non-trivial application completely
- few (if any) individuals have a broad enough outlook and knowledge base to design a system with all features required by all users
- new ideas and enhancements occur constantly during development
- missing components are often only obvious after the system reaches a high level of completeness
- requirements can change during the course of system development
- people make mistakes
What is Prototyping?

Prototyping addresses these issues with an iterative or spiral process, in which a simpler model is continually refined toward the desired end. Basically, prototyping consists of a series of phases in which a model is discussed and refined by the stakeholders, and then implemented by the developers. Then the next phase follows the same pattern, but with a slightly more detailed model. A prototype starts simply and grows in complexity. All stakeholders – managers, users, developers, researchers, etc. – are included from the start and at all levels of development. The aim is to work together to realize the common objective and minimize misunderstanding and omissions. The methodology highlights flexibility to accommodate change right up until the product is finalized.

For many people, the idea that prototyping is quicker seems counter-intuitive, since so much time is spent discussing and reworking earlier versions. The secret is that the hard work is put in up front in prototyping, whereas waterfall methods invariably take longer than estimated (not uncommonly 3-4 times longer!). Also, since versions are evolutionary, each is built upon the strengths of the last one, and you rarely need to start again from scratch.

Steps in prototyping:

1) decide on the goal of the project and its major components or categories
2) choose one or two features to begin with (usually in the main category or home page)
3) create a preliminary design on paper
4) discuss the design with stakeholders with the aim of improving it
5) repeat steps 3 & 4 to generate a simple design that all can agree on
6) implement the design on the computer
7) repeat the consultation / refinement process until the features are ‘good enough’ to continue
8) add another feature and repeat the prototyping process of consultation and refinement

There are several important things to watch for during the process:

- be sure to hear from all stakeholders
- schedule regular meetings from the outset, to keep the project on time and avoid costly delays
- clarify who is able to approve a prototype and allow the team to move to the next step (this is typically one person, with perhaps a delegate or alternate)
- look for improvements and don’t see them as criticism
- don’t be afraid to make mistakes or say ‘foolish’ things – brainstorm
- don’t be afraid to change earlier decisions if new ideas warrant it
- don’t get bogged down on details – they may have to be changed later anyway
- be sure to keep to your timeline
- choose the appropriate step size for each version (discussed below)
- expect the early stages to take longer; the first few features will take the longest
- be sure to do the most important features early on, so that if you run out of time you still have a worthwhile system
- but don’t do the most important feature first as you are learning the method!
- work together on the first few features to familiarize the whole team with the look and feel
- if you delegate parts later be sure to compare them so that from the user’s point of view they are consistent
- try to design for your least sophisticated user
- don’t have people test the parts that they have coded

Some tips:

- designing on paper at first speeds up the project and keeps the team from being ‘mesmerized’ by the computer screen
- paper designs bridge the gap between those comfortable with technology and those less comfortable or knowledgeable
- if this is your first project, begin with an easy feature
- having a good paper layout makes screen design much simpler
- spend more time designing and less time coding
- precise details such as sizes of buttons or placement of graphics are often best left until later when more features are added and the impact of the details can be assessed
- use the experts you have – for example, not everyone has the design sense of a trained graphic designer
- don’t have separate teams working independently on different sections, as these rarely mesh into a coherent single project
- invite people who are not tech-savvy and don’t know the project to help with testing to get a real world viewpoint

Choosing an appropriate step size between versions is important to keep the project manageable. Too large a step (i.e. too many new features or very difficult ones) may result in frustration and non-productive meetings. Steps that are too small waste time and resources, which remain idle waiting for approval or critique. While step size changes in every project, it should be based on the time between project meetings, the personnel available, the skills available, and the maturity of the project. In the later stages, more can be done in a step because earlier steps can be used as frameworks and models.
Pros and Cons of Prototyping

Benefits of prototyping include:

- systems which are finished more quickly
- less maintenance required
- greater client satisfaction
- more collaboration for stakeholders

In a well-run prototyping project, misunderstandings between clients and developers (and within the development team) are often identified as the prototypes are demonstrated, rather than at the end of development. Missing features can be identified early on. Difficulties in use or confusing features can be identified and fixed quickly. Developers may identify incomplete or even conflicting requirements as they implement them, and users have the opportunity to clarify them before time is wasted coding incorrect requirements. If necessary, a demonstrable prototype, though not fully functional, can be shown to management as proof of concept or to justify further funding.

    Even in the worst case, the prototype can provide the basis for a specification for a more complete, complex system.

Nothing is perfect, and especially when starting with prototyping there are some pitfalls you must avoid:

- more involvement is required of stakeholders – more meetings, more thought, more effort
- managers or funding agencies may not realize that early prototypes are mostly ‘smoke and mirrors’ and may think that the project is more complete than it is
- developers may forget some of the ‘quick and dirty’ compromises made in early versions and these can create problems later on
- testers may not use the system as real users would, thus failing to expose problems that should be fixed before the system goes live
- documentation needs to be constantly updated as the system changes during each prototype phase

Finally, as with any system, there will be a need for ongoing maintenance. Prototyped systems have been demonstrated to have lower maintenance costs, but there is always a need for software upgrades, bug fixes, and unexpected glitches, not to mention new features that may be added in the future.