17. Prototyping

Contents

Introduction

Prototypes are experimental and incomplete designs which are cheaply and fast developed. Prototyping is an integral part of user-centred design and the usability engineering lifecycle because it enables designers to try out their ideas with users and to gather feedback.

The main purpose of prototyping is to involve the users in testing design ideas and get their feedback in the early stage of development, thus to reduce the time and cost. It provides an efficient and effective way to refine and optimise interfaces through discussion, exploration, testing and iterative revision. Early evaluation can be based on faster and cheaper prototypes before the start of a full-scale implementation. The prototypes can be changed many times until a better understanding of the user interface design has been achieved with the joint efforts of both the designers and the users.

Prototyping can be divided into low-fidelity prototyping and high-fidelity prototyping.

Low-fidelity prototypes are quickly constructed to depict concepts, design alternatives, and screen layouts, rather than to model the user interaction with a system. Low-fidelity prototypes provide limited or no functionality. They are intended to demonstrate the general look and the feel of the interface, but not the detail how the application operates.

High-fidelity prototypes are fully interactive, simulating much of the functionality in the final product. Users can operate on the prototype, or even perform some real tasks with it.

Low Fidelity Prototyping

Sketches and paper prototypes

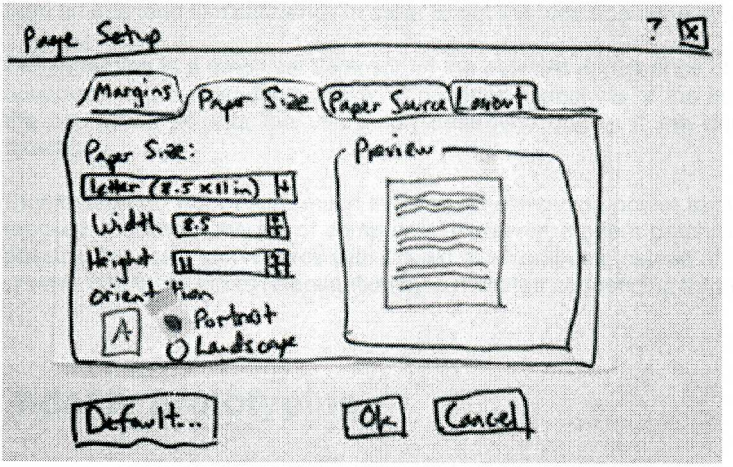
Sketching techniques, a kind of visual brainstorming, can be useful for exploring all kinds of design ideas. After producing initial sketches the best ideas can be further developed by constructing cardboard representations of the design, which can be evaluated with users. This can then be followed by developing scenarios, software or video prototypes

The type of mock-up depends on how advanced the idea is. It may be quicker and cheaper to use paper-and-pencil forms at early stages, whereas computer-based prototypes may be important in later stages for exploring and demonstrating interaction and design consistency.

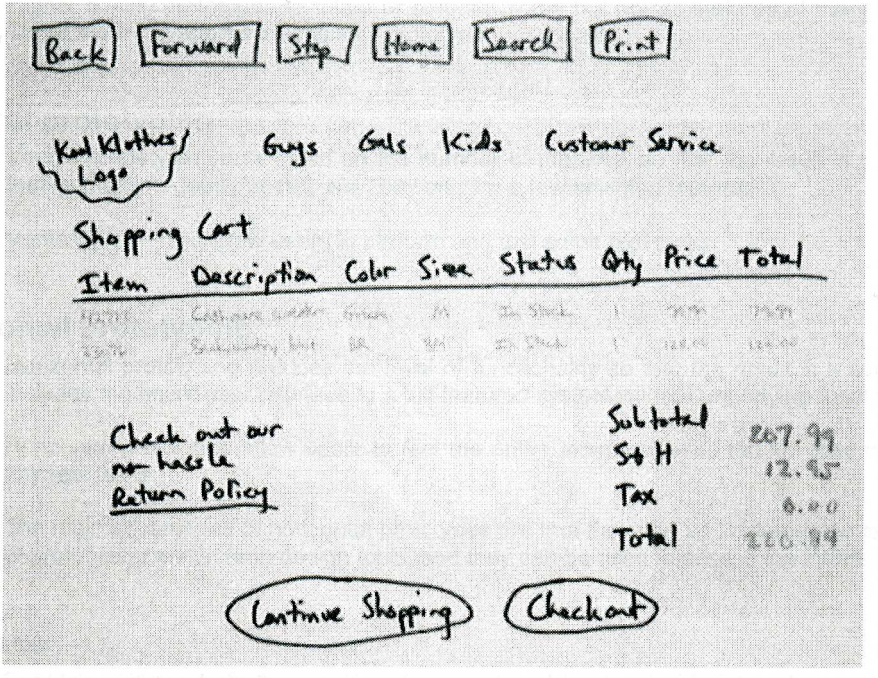
Paper prototyping is a method of usability testing that is useful for Web sites, Web applications, and conventional software. Here's how it works: You first decide on the tasks that you'd like the user to accomplish. Next, you make screen shots and/or hand-sketched drafts of the windows, menus, dialog boxes, pages, popup messages, etc. that are needed to perform those tasks. Then you conduct a usability test by having one or two developers play the role of "computer," manipulating the pieces of paper to simulate how the interface would behave. Users are given realistic tasks to perform by interacting directly with the prototype -- they "click" by touching the prototype buttons or links and "type" by writing their data in the prototype's edit fields. (Using transparency or removable tape prevents the prototype from being written on directly.)

A facilitator (usually someone trained in usability) conducts the session while other members of the development team observe and take notes. The "computer" does not explain how the interface is supposed to work, but merely simulates what the interface would do. In this manner,

you can identify which parts of the interface are self-explanatory and which parts are confusing. Because the prototype is all on paper, you can modify it very easily to fix the problems you find.



Paper prototype of a File Setup dialogue (from IBM Developerworks)



Paper prototype of a web shopping cart – user 'input' is on removable tape. (from IBM Developerworks)

Storyboarding

Storyboards originate from the film industry, where a series of panels roughly depicts snapshots from an intended film sequence in order to get the idea about the eventual scene.

Storyboarding is a graphical depiction of the outward appearance of the intended system without accompanying system functionality. It provides snapshots of the interface at particular points in the interaction so that the users can determine quickly if the design is heading in the right direction.

Storyboards do not require much in terms of computing power to construct, in fact, they can be mocked up without the aid of computers. However, modern graphical drawing packages make it possible to create storyboards with the aid of a computer instead of by hand. It is also possible to provide crude but effective animation by automated sequencing through a series of snapshots

High-fidelity prototyping

Computer-based simulation

Higher fidelity prototypes simulate or animate some but not all features of the intended system. There are three approaches to limit prototype functionality.

Vertical prototyping

Vertical prototyping cuts down on the number of features, so that the result is a narrow system that includes in-depth functionality, buy only for a few selected features.

Vertical prototypes allow users to perform and test some real tasks.

Horizontal prototyping

Horizontal prototyping reduces the level of functionality so that the result is a surface layer that includes the entire user interface to a full-featured system without underlying functionality.

Horizontal prototypes allow users to feel the entire interface, even though they can not perform any real tasks.

The main advantages of horizontal prototypes are that they can be implemented fast with the use of prototyping and screen design tools, and they can be used to assess the interface as a whole.

Scenario

Scenario reduces both the number of features and the level of functionality. It can simulate the user interface as long as the user follows a previously planned path, i.e., a user can use a specific set of computer facilities to achieve a specific outcome under specified circumstances.

Scenarios can be easy and cheap to build, and to be used during early evaluation of a user interface design to get user feedback without the expense of constructing a running prototype. It can also be used for user testing if they are developed with slightly more detail than a pure narrative.

Wizard of Oz

This is a method of testing a system that does not exist. It allows designers to test ideas without implementation a system.

The Wizard of Oz technique works as follows: the user interacts with a screen, but instead of a piece of software responding to the user's requests, a developer (the wizard) is sitting at another screen (generally in another room) simulating the system's intelligence and interacting with the user. The wizard may simulate all or part of the system function. When setting up a Wizard of Oz simulation, experience with previously implemented systems is helpful in order to place realistic bounds on the wizard's "abilities"

Chauffeured

In Chauffeured Prototyping, a third party walks the user through the prototype. This may be used with vertical prototyping, when only a select few paths have been implemented.

Rapid Prototyping

In rapid prototyping interactive prototypes are developed which can be quickly replaced or changed in line with design feedback. This feedback may be derived from colleagues or users as they work with the prototype to accomplish set tasks.

Various techniques may be used for rapid development

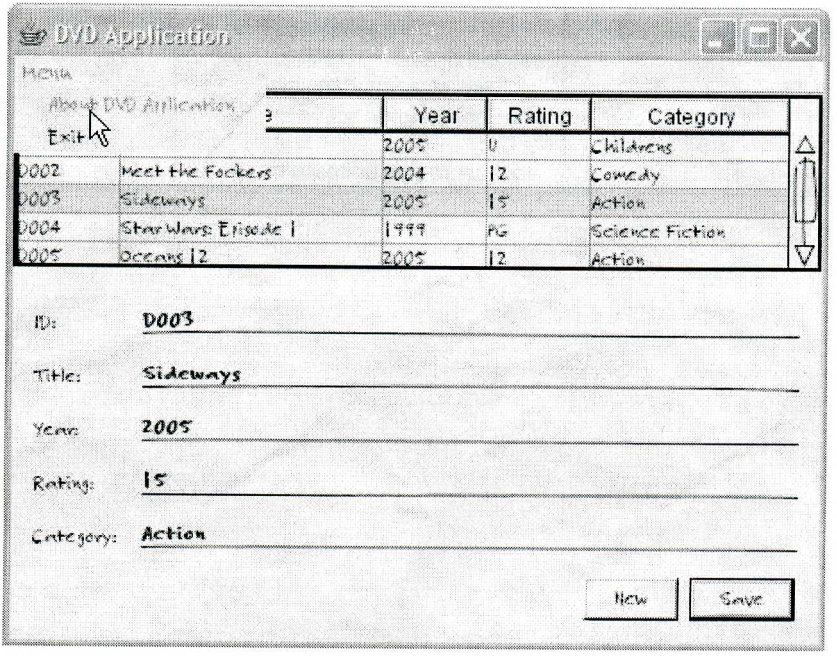
- Dynamic high-level language development
- Database programming

Visual programming is an inherent part of most prototype development systems

Problem – false expectations

Because High Fidelity prototypes look real, this can lead customers or management to believe that the product is nearly finished. Imagine this situation: a manager tells you to build a prototype to test the look and feel of a new application. You work for a week straight, day and night, producing code, which will work for the prototype, but will not be suitable for the finished product, to make the Big Demo. After all of your work, the meeting is a huge success and everyone loves your prototype. Then someone asks you how long until the application is ready to be deployed. You respond with a reasonably worked estimate of six to nine months. The feeling in the room quickly changes - no one understands why its going to take nine months to build something that they already see in front of them. No one is happy anymore.

Some developers design their High Fidelity prototypes so that it is clear visually that this is a prototype, not a finished application. One way of doing this, which is used by many developers, is shown in the figure below. This shows a working GUI form created in Java. Java allows the 'look-and-feel' of an application to be altered, and the one shown here uses a special look-and-feel called 'Napkin', which is meant to look like the form has been drawn roughly on a napkin!



Comparison of low-fidelity and high fidelity prototyping

Type	Advantages	Disadvantages
Low-Fidelity	• less time & lower cost • evaluate multiple design concepts • useful communication device • address screen layout issues	• limited usefulness for usability tests • navigational and flow limitations • facilitator-driven • poor detailed specification
High Fidelity	• partial/complete functionality interactive • user-driven • clearly defines navigational scheme • use for exploration and test • marketing and sales tool	• time-consuming to create • inefficient for proof-of-concept designs • managements may think it is real

Prototyping in the software process

Evolutionary prototyping

This is an approach to system development where an initial prototype is produced and refined through a number of stages to the final system. It is used for systems where the specification cannot be developed in advance and is based on techniques which allow rapid system iterations. Advantages include accelerated delivery of the system and user engagement with the system.

Throw-away prototyping

A prototype which is usually a practical implementation of the system is produced to help discover requirements problems and then discarded. The techniques of Rapid Prototyping can be used. The system is then developed using some other development process. Customers and end-users should resist the temptation to turn the throw-away prototyping into a delivered system that is put into use.

There is an important difference between the objectives of evolutionary and throw-away programming:

- The objective of evolutionary prototyping is to deliver a working system to end-users.
- The objective of throw-away prototyping is to validate or derive the system requirements.

Incremental development

The system is developed and delivered in increments after establishing an overall architecture. The requirements and specifications for each increment may be developed. Users may experiment with delivered increments while others are being developed and so these can serve as a form of prototype system