Low *vs.* High-

An ongoing controversy exists in the prototyping community about how closely in form and function a user-interface prototype should represent the final product. This dispute is referred to as the "Low-versus High-Fidelity Prototyping Debate." In this article, we discuss arguments for and against low-and high-fidelity prototypes, guidelines for the use of rapid user-interface prototyping, and the implications for user-interface designers.

Jim Rudd
Ken Stern
Scott Isensee

-Fidelity Prototyping Debate

Although prototyping has been recognized as an efficient and effective means of developing user interfaces for some time [8] and has become an integral part of the development process in many organizations (e.g., [10,17]), the optimum methods of prototyping have not yet been agreed upon. We know that, if done systematically, prototyping provides the means to model software applications to support the evaluation of design alternatives early in the product development cycle. We understand that the use of iterative design promotes the refinement and optimization of interfaces through discussion, exploration, testing, and iterative revision. The experiences of many designers in developing and evaluating user-interface prototypes provide testimonials regarding the many applications and benefits of prototypes (e.g., [3, 5]).

The low-versus high-fidelity debate lies in the fidelity of prototype required to illustrate a concept, model design alternatives, or test an application. The debate rages to whether prototypes need to be complete, realistic, or reusable to be effective.

Types of Prototypes

Prototypes can be generally classified into two categories: low-fidelity and high-fidelity.

Jim Rudd
Ph.D.,CPE
Senior Scientist
IBM Corporation
6401 Morningsong Lane
Charlotte, NC
28269-0801
jrudd@vnet.ibm.com

Kenneth R. Stern
Ph.D., CPE
Advisory Scientist
IBM Corporation
8501 IBM Drive
MG22/103
Charlotte, NC 28262
kstern@vnet.ibm.com

Scott Isensee
CPE
User Interface
Architectural Team Leader
IBM Corporation
11400 Burnet Road
Austin, TX 75758
isensee@austin.ibm.com
http://home.aol.com/Isensee

Low-fidelity prototypes are generally limited function, limited interaction prototyping efforts. They are constructed to depict concepts, design alternatives, and screen layouts, rather than to model the user interaction with a system. Storyboard presentations and proof-of-concept prototypes fall into this category. In general, low-fidelity prototypes are constructed quickly and provide limited or no functionality. Low-fidelity prototypes demonstrate the general look and perhaps the feel of the interface; they are not intended to show in detail how the application operates. These prototypes are created to communicate, educate, and inform,but not to train, test, or serve as a basis from which to code.

Tullis [13] contends that the fidelity of a prototype is judged by how it appears to the person viewing it, and not by its similarity to the actual application. In other words, the degree to which the prototype accurately represents the appearance and interaction of the product is the determining factor in prototype fidelity, not the degree to which the code and other attributes invisible to the user are accurate.

The mind-set for low-fidelity prototypes is that prototyping is rapid, with the prototype code not reused once product coding is initiated. Heaton [4] believes that rapid prototyping should solve 80% of the major interface problems, with the speed of producing a prototype early (during the requirements-specification phase) outweighing the need to produce a final model.

Low-fidelity prototypes can consist of a series of static windows or menus that can be rapidly generated and displayed, either singly or in a storyboard presentation. Mark van Harmelen [14] classifies these noninteractive prototypes as scenario tools. Despite limited functionality, these scenarios can play an important role in visualizing the use of an interface. These low-fidelity prototypes are created to show visuals, including colors, icons, and the placement of controls. They will show design direction, but will not provide details such as navigation and interaction. Low-fidelity prototypes have very little functionality built in—users can see what the product is supposed to do, but the prototype may not respond to user input. Users do not exercise a low-fidelity prototype to get a first-hand feel for how it operates; rather, low-fidelity prototypes are demonstrated by someone skilled at operating the prototype. The presentation of a low-fidelity prototype is often carefully scripted to allow the presenter to tell a story about how the product will eventually operate. Low-fidelity prototypes are often used early in the design cycle to show general conceptual approaches without much investment in development. A low-fidelity prototype may be as simple as a paper-pencil mockup that shows general flow through the screens.

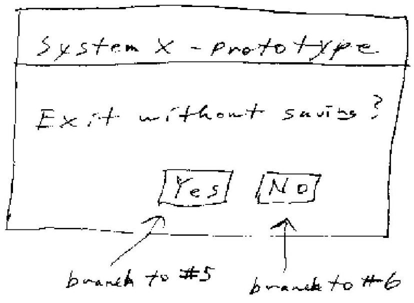
Low-fidelity prototypes generally require a facilitator, who knows the application thoroughly, to demonstrate or to test the application. Interactivity by the user is somewhat restricted. The user is dependent on the facilitator to respond to the user's commands to turn cards or advance screens to simulate the flow of the application.

In contrast, high-fidelity prototypes are fully interactive. Users can enter data in entry fields, respond to messages, select icons to open windows and, in general, interact with the user interface as though it were a real product. They are high-fidelity because they represent the core functionality of the product's user interface. High-fidelity prototypes are typically built with fourth-generation programming tools such as Smalltalk or Visual Basic, and can be programmed to simulate much of the function in the final product.

High-fidelity prototypes trade off speed for accuracy. They are not as quick and easy to create as low-fidelity prototypes, but they faithfully represent the interface to be implemented in the product. They can be made so realistic that the user can't tell them from the actual product.

However, in cases when time is a factor, it is still possible to develop what is called a "vertical prototype"—an interactive, high-fidelity prototype of only a subset of the product's available function. Another approach is to create "horizontal prototypes"—prototypes that contain high-level functionality, but do not contain the lower-level detail of the system (Floyd 1984 in [14]). These prototypes may be limited in scope, yet they can be quickly created and provide user-interface interactivity that may be essential for specific product design decisions.

Low-Fidelity Prototypes

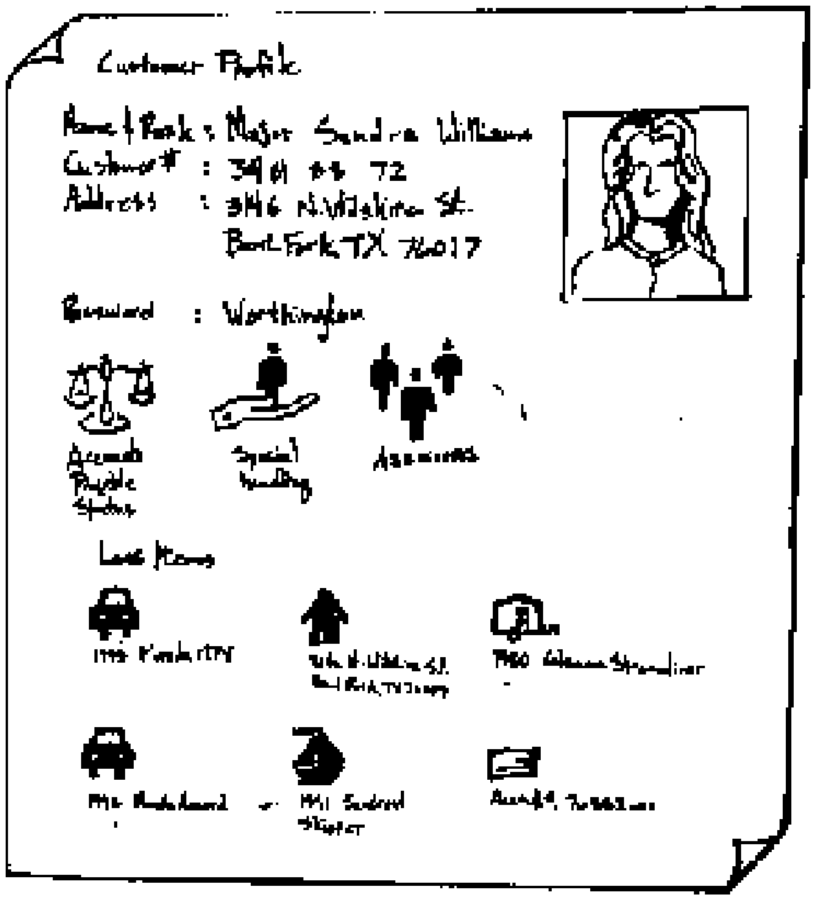


Advantages

Low-fidelity prototypes have great value in the early requirements gathering and analysis phase of product development. Customers and end users often don't know how to articulate their requirements. Verbalizing design requirements is often not objective, as design concepts may be biased by a customer's mental model of the system. In other words, customers may have a difficult time in separating what they want a system to do from how they want the tasks to be performed. A low-fidelity prototype can be the communication medium by which requirements can be articulated. A prototype can serve as the common language to which users and developers can relate. Low-fidelity prototypes are well suited for use in user interface design teams (UIDT) [5, 11], allowing developers to maintain a nearly focus on users.

Low-fidelity prototypes can be constructed early in the development cycle without a large investment in time and development dollars. Because they are constructed with paper and pencil or simple storyboard tools, low-fidelity prototypes require little or no programming skill on the part of the designer. Low-fidelity prototypes are easily portable—they can be presented on paper, view graphs, or white boards. A low-fidelity prototype can be used as a first step in proposing fundamental design approaches for the user interface. The prototype can be demonstrated to potential users to obtain feedback on how well the design meets their needs or which of several designs is most on target. This feedback can be used to iterate further the low-fidelity prototype or as requirements input for follow-on higher-fidelity prototyping.

Rettig [9] describes paper prototyping as the fastest of the rapid prototyping techniques. By sketching and designing ideas and concepts onto paper, interface designers are able to focus on aspects of design and not on the mechanics of a tool. Rettig used paper prototyping to design and sketch a prototype of an automated menu for a fast food restaurant. In a six-hour period,



developers designed an interface, built a paper model of it, tested it, and improved the initial design. The tools for building a paper prototype are simple, requiring paper, index cards, markers, adhesives, and other office supplies.

Ballantone and Lanzetta [1] used a low-fidelity prototyping approach to create and project non-programmable terminal user-interface panels onto an overhead screen for review and revision. Members of a user-interface design team addressed layout and terminology concerns with the panels. A walk through of the prototype was conducted, with the objective to gain consensus that the prototype was complete and "flowed" properly. Based on design change recommendations, the panels were iteratively modified and reevaluated until consensus among team members was reached.

Disadvantages

Low-fidelity prototypes represent broad brush approximations to the eventual design. Just because something can be represented in the prototype and artfully demonstrated to a set of users does not mean that the approach will be feasible in the product. Because low-fidelity prototypes are typically crude, and can provide little error checking, many important design decisions are often overlooked.

Low-fidelity prototypes may not provide a good forum for user evaluation. Because they are often demonstrated to, rather than exercised by, the user, it is more difficult to identify design inconsistencies and shortcomings. Nielson[7] compared the effectiveness of a high-fidelity interactive prototype with that of a low-fidelity static paper and pencil prototype for identifying shortcomings.Two groups of evaluators were asked to evaluate each of the two prototypes. There were 50 usability problems with each of the prototypes, 15 of which were labeled major. Nielson found that evaluators discovered significantly more problems with the high-fidelity prototype than with the low-fidelity prototype.

It is difficult for programmers to code to a low-fidelity prototype. Because the prototype is not fleshed out, the programmer is forced to make personal decisions about such details as

Table 1.

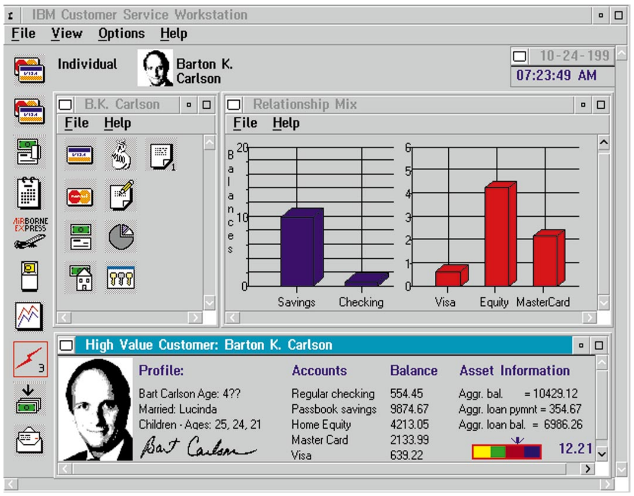
Relative effectiveness of low-vs.high-fidelity prototypes

This table summarizes the various advantages and disadvantages for conducting low-and high-fidelity prototyping efforts.

Type	Advantages	Disadvantages
Low-Fidelity Prototype	Lower development cost.	Limited error checking.
	Evaluate multiple design concepts.	Poor detailed specification to code to.
	Useful communication device.	Facilitator-driven.
	Address screen layout issues.	Limited utility after requirements established.
	Useful for identifying market requirements.	Limited usefulness for usability tests.
	Proof-of-concept.	Navigational and flow limitations.
High-Fidelity Prototype	Complete functionality.	More expensive to develop
	Fully interactive.	Time-consuming to create.
	User-driven.	Inefficient for proof-of-concept designs.
	Clearly defines navigational scheme.	Not effective for requirements gathering.
	Use for exploration and test.	
	Look and feel of final product.	
	Serves as a living specification.	
	Marketing and sales tool.	

user interaction and navigation, the content of error messages, the layout and architecture of help panels, and the design of little-used functions. This can be troublesome, as programmers with little user-interface design experience consistently make bad design decisions about what should and should not be done, even if they use an established design guide as a reference. An interesting study by Tetzlaff and Schwartz [12] bears this out. They found that programmers inexperienced at UI development often relied more on the pictures in design guidelines than on the supporting text in making design decisions. The inexperienced programmers also showed higher levels of frustration than experienced UI programmers when the design guide did not provide obvious and explicit rules for complex application-specific problems.

High-Fidelity Prototypes



Profile:
Bart Carlson Age: 4??
Married: Lucinda
Children - Ages: 25, 24, 21

Accounts	Balance
Regular checking	554.45
Passbook savings	9874.67
Home Equity	4213.05
Master Card	2133.99
Visa	639.22

Asset Information
Aggr. bal. = 10429.12
Aggr. loan pymnt = 354.67
Aggr. loan bal. = 6986.26

get a feel for how the product will operate and, therefore, can make informed recommendations about how to improve its user interface.

Usability testing can be conducted early in the design process with the prototype as a test vehicle. Realistic comparisons with competing products can be made via the prototype to ensure that the program is marketable and usable before committing the resources necessary to develop the product fully. A representative prototype can be available months before the product code, allowing usability testing,test case construction, help panel design, and documentation to be initiated much earlier in the development cycle.

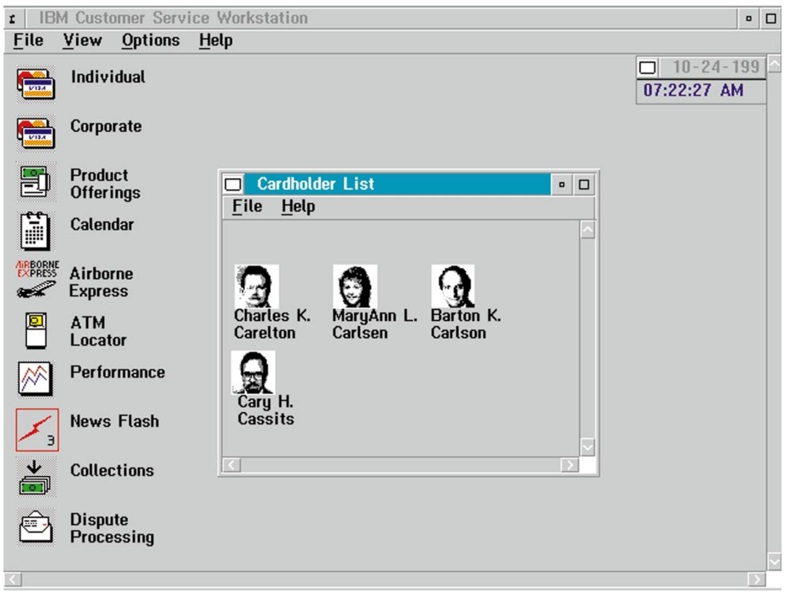
High-fidelity prototypes are a very good educational and productivity tool for programmers and information developers. The programmer can use the prototype as a living specification of the functional and operating requirements. Whenever the programmer needs design guidance, the prototype is fired up and the function in question is executed to determine its design. This can save substantial time over the typical development process where programmers sometimes make design decisions on the fly, which may require expensive rework to fix later. Information developers can generate more useful help panels and documentation earlier in the development process by running the prototype and identifying where users may have problems. In addition, the information developer will better understand the product because of increased exposure to the product's user interface.

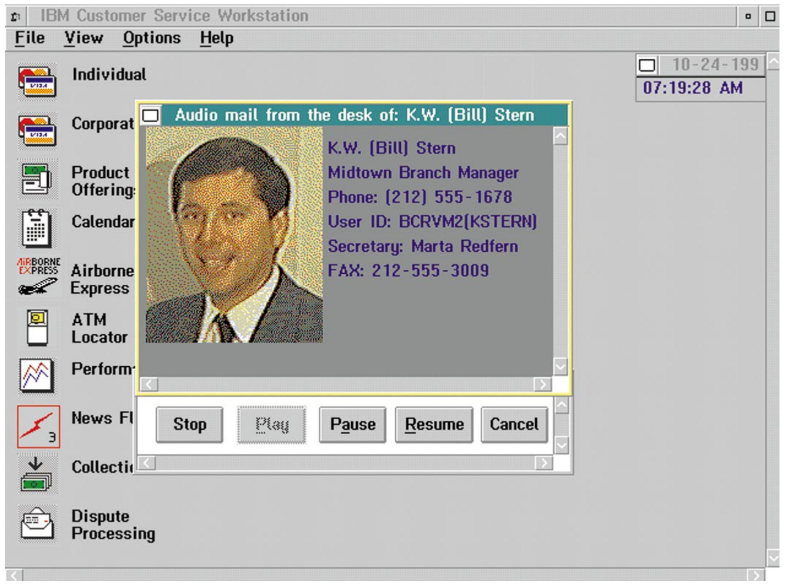
High-fidelity prototypes can make great

Advantages

Unlike low-fidelity prototypes, high-fidelity prototypes have complete functionality and are interactive. Whereas low-fidelity scenarios address the layout and terminology of applications (surface presentation), high-fidelity prototypes address the issues of navigation and flow and of matching the design and user models of a system [14].

High-fidelity prototypes are interactive and are used for exploration and test. The user can operate the prototype as if it were the final product. The user can select icons on the screen and expect windows to open or functions to be launched. Messages are delivered to the user at appropriate times. Data can be displayed in a real-time fashion and updated periodically. The user can take actions in response to data updates. Errors and deviations from the expected path will be flagged and identified to the user as if using the real product. The prototype will respond in a manner that represents the behavior of the eventual product. In general, the user can





K.W. (Bill) Stern
Midtown Branch Manager
Phone: (212) 555-1678
User ID: BCRVM2(KSTERN)
Secretary: Marta Redfern
FAX: 212-555-3009

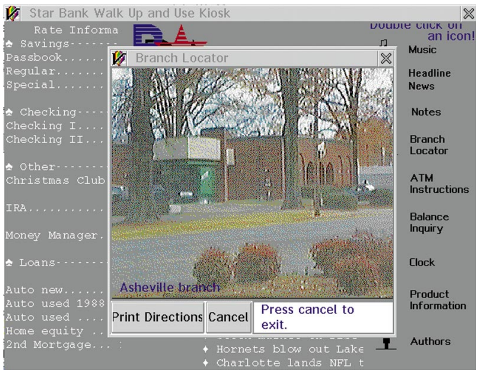
marketing and sales tools. Customer input can be solicited at customer sites and trade shows and this feed backused to refine the prototype.

High-fidelity prototypes can be used very effectively to encourage customer buy-in. Because the prototype is more fully functional than a low-fidelity prototype, it provides a better basis for thorough evaluation by end users. End-user feedback can be used to fine-tune the prototype. Because changes can be made rapidly, end-user feedback can quickly be incorporated into the prototype. This rapid turnaround fosters greater end-user acceptance, as users can immediately see their design recommendations put into place. End users become constructive, contributing members, rather than "evaluators" of the design team when they see immediate turnaround on their input. A high-fidelity prototyping effort implemented properly can provide this buy-in.

Disadvantages

There are, however, some serious shortcomings to high-fidelity prototyping. Generally, high-fidelity prototypes are more expensive and time consuming to construct than low-fidelity prototypes. Because high-fidelity prototypes represent function that will appear in the final product, prototype development becomes a development effort in itself, sometimes requiring many weeks of programming support. Although high-level languages and screen builders exist to make this process easier, high-fidelity prototyping still requires a substantial programming effort. This places constraints on who does interface design. It is difficult to find people who are both good interface designers and skilled programmers.

When customers see a high-fidelity prototype, it often appears that the product is ready. If the prototype is much better than the product that they are using currently, they may demand it immediately. It is hard for customers to understand that substantial effort may be required to write the base code and perform the testing required to turn a prototype into a commercial product. In addition, not all the function demonstrated in a prototype may be incorporated into the product. For reasons such as cost, schedule,



and perceived customer interest, function may be dropped from the time the customer sees the prototype to the time the product is released.

Often, funds are not committed to develop a high-fidelity prototype. Given a choice, developers often would opt to not construct any prototype, but to test the interface once it is fully coded, during the test phase. Although happening with less frequency in recent years, there are many development groups that perceive extensive prototyping as an unnecessary added expense and a duplication of effort of a programmer's job. They fear that a high-fidelity prototype may result in an impact to product development costs and schedules, the measures to which most development efforts are rated.

Unlike low-fidelity prototypes, the high-fidelity variety is not good for investigating more than two or three design approaches to the interface. High-fidelity prototypes are simply too expensive to construct and often the detail provided by such a prototype is extraneous to the more important issues at hand. When we are investigating various design approaches, we do not care about the wording of message boxes or the content of help panels. Rather, we are trying to determine if a particular approach is appropriate for the intended set of users. High-fidelity prototyping is not a good tool for identifying conceptual approaches unless the alternatives have been narrowed down to two or three, or perhaps even one, depending upon budget and schedule.

A disadvantage of presenting alternative design options in a prototype is that design decisions are often made too quickly. Generally, products are tested extensively before being released to the field. Testing ensures that the product meets performance, service, and usability guidelines. Given the speed in which rapid prototypes can be generated, there may be the tendency to make rash decisions regarding design options without ensuring that the proper validations are conducted.

There is a time and place in the development process for both low- and high-fidelity prototypes. The following sections describe when low- and high-fidelity prototypes are most useful.

Tips for Using Low-Fidelity Prototypes

Use low-fidelity prototyping, and lots of it, when the team is trying to identify market and user requirements. The low-fidelity prototype can be used to generate ideas quickly and cost effectively about how the product might work. Low-fidelity prototypes are useful in providing a broad brush user-interface design [16], where alternative designs can be quickly generated and evaluated during the requirements-gathering stage.

Use low-fidelity prototypes to provide a common language among development and support groups. A prototype can provide customers and developers with an understanding of the application that cannot be obtained by reading the functional specifications and can serve as an educational aid in understanding how an application works [1].

Use low-fidelity prototypes to investigate early concepts about what function the product might have and how it might be presented to the user. It can allow the team to get an early understanding of approaches to screen design and navigation. The prototype can allow the team to iterate through a number of alternatives without a great investment in time and money. The prototype can provide a degree of comfort regarding the suitability of design approaches early in the development cycle.

Remember that the goal of low-level prototyping is to evaluate design alternatives and depict concepts. Do not get bogged down in the details. Evaluate broad-brushed approaches to particular user-interface design approaches.

Constrain the list of tasks to the set of most common tasks that the user performs. Build the prototype around these tasks and don't worry about the other, less important, tasks the user performs. Structure the use of the prototype (demos, focus groups, interviews) around these common tasks. Thus, the prototype is vertically, rather than horizontally, integrated.

Use the prototype as a straw man to elicit customer input during requirements gathering. Users know the tasks that software should help them perform, but they may not know how to express these requirements in ways that are useful for interface design. A low-fidelity prototype gives them some idea of what is possible, providing a starting point for discussion and a target for criticism.

There are places in the design process where low-fidelity prototyping clearly does not belong.

Do not use low-fidelity prototyping anytime after the product requirements have been decided upon and coding has started. At this point in time, the prototype is too vague and too incomplete to give the programmers the guidance they need in developing the product.

Be careful in the use of low-fidelity prototypes as a vehicle for testing user-interface issues pertinent to the product under development. A low-fidelity prototype may be suitable for conducting qualitative evaluations, but will not have the detail necessary to allow the design team to make quantitative decisions with high levels of confidence about user-interface nuances.

Tips for Using High-Fidelity Prototypes

Use high-fidelity prototyping to create a living specification for programmers and information developers. High-fidelity prototyping should be done in conjunction with the development of a written specification. Screen captures of the prototype, as well as descriptions of its use, can be included in the written specification. Give programmers a working version of the prototype to refer to when the written specification does not provide the clarity or the detail they need. Give the information developers a copy of the prototype to help them develop the users' manuals and help panels.

Use high-fidelity prototypes at trade shows to show the public how the product will operate prior to the code being fully developed. This gives the development and marketing team a leg up on getting the word out about future releases.

Use high-fidelity prototypes for testing user-interface issues.Because high-fidelity prototypes model the application and have error and help information built in, the actions of the user will more closely resemble the actions of the eventual users of the product. Because high-fidelity prototypes can be available months before coding is completed, the opportunity exists to affect and test changes before the design has been frozen. It is possible to instrument a prototype to automate some of the data collection for user testing. Add code hooks to collect mouse and key stroke time and error data. These data can be used in fine-grained statistical analyzes.

There are phases in the development

Table 2.

Prototyping Considerations

This table summarizes some of the key points to consider when deciding whether a low or high-fidelity prototyping effort would be most appropriate for your design and development needs. Bullets indicate suitable choices.

- **Cost constraints?**
- **Define market requirements?**
- **Schedule constraints?**
- **Screen layout?**
- **Navigation and flow?**
- **Proof-of-concept?**
- **Communications medium?**
- **Training overview?**
- **Training tool?**
- **Usability test?**
- **Basis for coding?**
- **User driven?**
- **Facilitator driven?**
- **Data collection?**
- **Look-and-feel of product?**

process where high-fidelity prototyping clearly does not fit.

Do not develop three or more high-fidelity prototypes during requirements gathering. It is a waste of resource. It is too early in the game to be making huge investments in programming services when the market requirements are not clearly understood.

Development of a high-fidelity prototype requires programming skills on the part of the interface designer and may be time consuming and expensive. If your skills, schedule, and/or budget don't allow for this, a high-fidelity prototype may not be an option. In this case, consider developing vertical prototypes to design and test a subset of available function.

Summary and Conclusions

In many ways, the debate over the relative value of low-versus high-fidelity prototyping is moot. This article has argued that both low-and high-fidelity prototypes have a place in the design process. It is hoped that the tips provided here will guide the user-interface designer in their prototyping efforts.

References

[1] Bellantone, C.E. and Lanzetta, T.M. Works as Advertised: Observations and Benefits of Prototyping. Technical Report TR-36.0005. IBM Corporation, Southbury, CT, 1992.

[2] Dumas, J.S. and Redish, J.C. A Practical Guide to Usability Testing. Norwood, NJ: Ablex Publishing Company, 1993.

[3] Greitzer, F.L., Wunderlich, D., and Weinberg, M. Hypermedia-based rapid interface prototyping. Journal of the Society for Information Display, 1, 1 (1993): 111-119.

[4] Heaton, N. What's wrong with the user interface: How rapid prototyping can help. In IEE Colloquium on Software Prototyping and Evolutionary Digest London, IEE (1992), Digest No. 202, Part 7, pp. 1-5,

[5] Kinoe Y. and Horikawa, Y. Eliciting requirements for a new product's user interface design: The customer prototype express.. Technical Report TR58-0963. IBM Corporation: Yamato, Japan, 1991.

[6] Melkus, L.A., and Torres, R. Guidelines for the use of a prototype in user interface design. In Proceedings of the Human Factors Society32nd Annual Meeting (Santa Monica, CA, 1988, Human Factors Society), pp.370-374.

[7] Nielson, J. Paper versus computer implementations as mockup scenarios for heuristic evaluation. In Proceedings of the Third International Conferences on Human-Computer Interaction, 1990, pp. 1-8.

[8] Pfauth, M., Hammer, A., and Fissel J. Software prototyping as a human factors tool. In Proceedings of the Human Factor Society 29th Annual Meeting (Santa Monica, CA, 1985, Human Factors Society), pp. 467-469.

[9] Rettig, M. Prototyping for tiny fingers. Communications of the ACM, 37, 4 (1994): 21-27.

[10] Rudd, J.R. and Isensee, S.H. 22 tips for a happier, healthier prototype. ACM Interactions, 1, 1 (1994): 35-40.

[11] Telek, M.J. and Schell, D.A. The user interface design team(UIDT) process. In Proceedings of the 1992 GSD Technical Symposium (IBM Corporation, Bethesda, MD 1992), pp. 115-125.

[12] Tetzlaff, L. and Schwartz, D. The use of guidelines in interface design.Human Factors in Computer Systems (CHI '91), (1991), pp. 329-333.

[13] Tullis, T.S. High-fidelity prototyping throughout the design process.In Proceedings of the Human Factors Society 34th Annual Meeting (Santa Monica, CA, Human Factors Society 1990), p. 266.

[14] van Harmelen, M. Exploratory user interface design using scenarios and prototypes. In Sutcliffe A. and Macaulay L. (Eds.). People and Computers V: Proceedings of the Fifth Conference of the British Computer Society (Cambridge, Cambridge University Press 1989), pp. 191-202.

[15] Virzi, R.A. What can you learn from a low-fidelity prototype? In Proceedings of the Human Factors Society 33rd Annual Meeting (Santa Monica,CA, Human Factors Society, 1989), pp. 224-228.

[16] Virzi, R.A. Low-fidelity prototyping. In Proceedings of the Human Factors Society 33rd Annual Meeting (Santa Monica, CA, Human Factors Society 1990), p. 265.

[17] Windsor, P. and Storrs, G. (1992). Prototyping user interfaces. InIEE Colloquium on Software Prototyping and Evolutionary Development, Part 4. (pp. 1-14). London: IEE.