Extending prototyping

by R. Van Buskirk
B. W. Moroney

Prototyping, a technique often employed by software developers, has been primarily used for usability studies. We discuss in this paper, through case studies, our experience in the use of prototyping during various phases of the product life cycle, including planning, testing, marketing, and field support.

A prototype can be thought of as a representation or mock-up of a proposed solution to a design problem, regardless of the medium.[1] The typical use of prototypes is for usability evaluations conducted in the design phase of a project.[2,3] For such an evaluation, the design team identifies tasks representative of those that the users of the product will perform, selects a set of users to participate in the usability experiment, and observes how well the participants perform these tasks. If the users have difficulty performing the tasks, changes are made to the design and additional usability experiments may be undertaken.

Prototyping is an excellent way of designing the user interface, including screen layouts, control labels, and other graphical user interface (GUI) characteristics. Usually the coding effort is small; that is, coding is "quick and dirty," and the code is eventually discarded. Because the cost of developing the prototype is low, there is little reluctance to modify it when usability testing requires such change. Prototyping work is best suited for developers skilled in higher-level-language quick-and-dirty programming, while developers focused on robustness, maintainability, and attention to detail excel on production-level code.

The prototype developed in the design and development phases of a project can range from a low-fidelity paper-and-pencil mock-up of the user interface to a highly functional, visually appealing, working prototype of the system. The two terms used in this context are "resolution," the level of detail implemented in the prototype, and "fidelity," the closeness to the target system.[1] The approach selected for prototyping depends on the stage of development, the scheduling constraints, and the degree of functionality required to evaluate the usability of the proposed solution. Often more than one prototype will be built during the life of the product.

When the "design window is narrow" and the design team must quickly evaluate several competing designs, a paper-and-pencil mock-up of the proposed solution can be quickly created and presented to selected users. These prototypes consist of screen layouts drawn on sheets of paper or created by using a presentation application such as Microsoft PowerPoint**. They are manually handled when presented to the user. For example, when the user presses a "button" on the paper mock-up, the evaluator presents the appropriate "dialog." Advantages of paper-and-pencil mock-ups include: (1) they can be created in a matter of minutes; (2) they can be modified on the fly, so a change may be as simple as erasing a label and writing a new one; (3) they are modular (when evaluating different Web page lay-

outs, for example, it is easy to run through a number of alternatives by rearranging the components); (4) they help the design team and the users focus on the higher level concepts of the solution, such as navigation or terminology, and ignore the less important aspects.

As the design evolves it is sometimes beneficial to move to a higher-fidelity prototype that incorporates the interactions among system components as well as the user-system interactions anticipated in the final product. There are many tools for developing such prototypes, including development environments such as Microsoft Visual Basic** and Borland Delphi**, which can be used for developing a range of implementations, from the initial prototype to the product-level implementation. For Web-based prototypes IBM WebSphere* Studio Homepage Builder, Microsoft FrontPage**, and Netscape Composer are inexpensive tools and relatively easy to use.

Changing from a low-fidelity prototype to a high-fidelity prototype need not require the finishing touches of the product "look and feel." In fact, in many cases it is beneficial to "keep it ugly" (a term our design teams use). Keeping it ugly helps maintain the focus on the usability of the proposed system. A prototype that looks too "polished" tends to discourage test participants from providing feedback about the solution being evaluated. When a prototype at this stage is shown to customers or colleagues, it is important to explain why the look and feel is still rough.

The design questions to be addressed will often dictate the type of prototype needed. Low-fidelity prototypes tend to be most useful when the function to be incorporated is limited (e.g., a simple wizard), while high-fidelity prototypes are useful when attempting to model more complex interactions among the system components. On another dimension, it is possible to create a *horizontal prototype* covering most of the user interface with little underlying functionality.[4] In contrast, a *vertical prototype* includes a narrow set of functions implemented in depth. In many design efforts it is beneficial to combine these approaches and create a high-level horizontal prototype with a limited number of vertical prototypes focused on the more challenging design issues.

Prototyping is not a panacea for dealing with design challenges, but rather one tool to be used in the design and development process. If the other steps associated with the User-Centered Design (UCD) process (task analysis, competitive analysis, design walkthroughs, beta testing, etc.) are not employed, prototyping may be of limited benefit.[5–7] For example, if a look-and-feel prototype is created prior to the completion of the task analysis, the design team may miss important functional requirements. When used in conjunction with UCD, prototyping provides a unique opportunity to "step into the future" and get a glimpse of how the product will perform, what it will look like, and how it will meet the customer's needs.

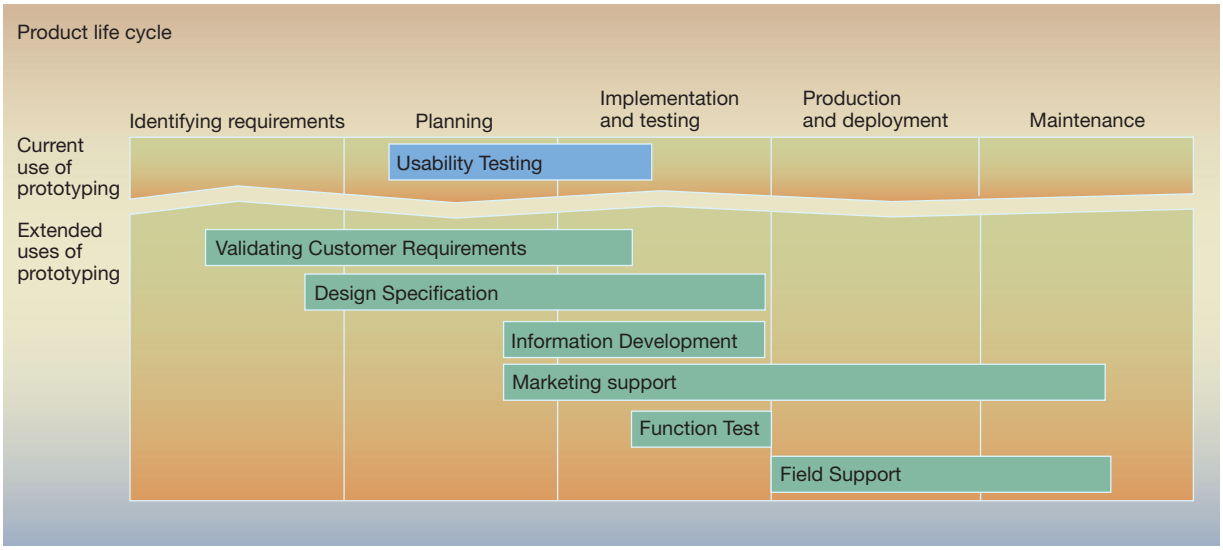
Prototyping has been covered in depth in the technical literature (see for example References 1 and 8). Houde and Hill, in particular, have defined a three-dimensional space for discussing prototypes in which the coordinates are *role*, *look and feel*, and *implementation*.[1] Prototypes high on the role scale focus on how the product will fit into the users' lives. Prototypes high on the look-and-feel dimension examine questions related to the sensory experience, whereas implementation prototypes are primarily targeted to questions regarding implementation methods and techniques.

Houde and Hill also discuss how prototyping relates to several distinct audiences: end users, design teams, and the supporting organizations (managers, business clients, etc.). They point out that designing interactive systems requires multidisciplinary teams, in which people with various skills collaborate. For example, a project might require the skills of programmers, industrial designers, usability experts, and project managers. Their collaboration is essential because they may have different expectations of the task at hand. Schrage points out that organizations develop their own "prototyping cultures" that may cause them to consider only certain kinds of prototypes.[8]

In many cases prototypes became not only a communication tool, but they also became a productivity tool used by various stakeholders inside and outside the development organization. Schrage draws an analogy between prototyping and a common language used by all stakeholders (lingua franca) by stating,[8] "Within some innovation environments, prototypes effectively become the *media franca* of the organization: the essential medium for information, interaction, integration, and collaboration." For more about usability testing, we refer interested readers to Tognazzini[2] and Lewis and Rieman.[3]

Just using a prototype for usability testing can be a boon to a project. It can help shrink schedules by

Figure 1 Beyond usability testing



reducing rework, and it can also help produce a more usable product. Moreover, as the use of prototyping has become more widespread, other less traditional uses of prototyping have appeared. In the next section we describe how prototyping is being used beyond usability testing, and in particular during planning, testing, marketing, and field support. Then we discuss three experience-based cases that illustrate some of these uses. In the last section we distill our experience into a set of guidelines for developers involved in a prototyping effort.

Beyond usability testing

As shown in Figure 1, prototyping can be extended beyond usability testing to other phases of the product life cycle. The x-axis shows the various phases of the product life cycle. These coarsely defined phases are only intended as major milestones; the discussion in this section should also apply to alternate ways of characterizing the product life cycle. Along the y-axis a number of extended uses of prototyping are illustrated. Information development, for example, is the activity that produces the documentation and help information for the product. As Figure 1 shows, the use of a prototype helps information developers to get an early start on learning how the product works. Similarly, the prototype could help the team responsible for function test get an early start into the test schedule. The proposed extended uses of prototyping are discussed in the following sections.

Validating customer requirements

A prototype can validate customer requirements early in the product cycle. Although a functional specification captures the specific items the requirements consist of, it sometimes fails to convey how the user experiences the product. "Walking through" a prototype with a user enables the design team to verify that they have captured the customer requirements. It is far better to hear, "That's not what I wanted," when only 5 percent of the coding is done, than after 100 percent of the coding has been completed.

The use of prototypes for validating customer requirements has its own pitfalls. A prototype that includes placeholders for components not yet implemented may give the appearance of performance that is faster or more accurate than the eventual product. For example, a speech recognition application may use computationally demanding algorithms to parse an audio stream into phonemes and a language model to assemble those into words. Because it is not performing any real computation, an early prototype of the system may appear faster than the real product would. Non-technical people may have trouble recognizing the difference between a prototype and the real product. When they see what appears to be a fully functioning version of the program, they may question why the contract calls for six more months of development. In such instances it is important to help the customer understand the assump-

tions of the demo and thus set appropriate expectations for the session.

Design specification

A prototype can serve as a "living" design specification. The intrinsic nature of a prototype forces the designer to specify exactly how the application works. Even after the design team agrees on a design specification, it is not uncommon, when the prototype is presented to them, for half the team to conclude that the behavior of the prototype is in error ("It's not supposed to perform the action directly. It's supposed to launch the properties dialog first!") and the rest of the team to disagree ("No, it's working exactly as we discussed. What are you talking about?"). Because the prototype operates under many of the constraints of real code, it forces the prototype developer to consider low-level details that often are left out in high-level design discussions. The prototype enforces the design specification and ensures that all have a common perspective.

When members of the development team are dispersed over several countries, a prototype can serve as a valuable communication tool. Understanding the implications of a design specification can be difficult even when all team members share a common language, as shown in the preceding example. This challenge might be amplified when members of the development team represent many languages and cultures. In this case a prototype can be a powerful tool in support of a written specification.

Because a prototype cannot replace the design specification, it is probably best to consider it a complement to the specification. A prototype is excellent at illustrating the dynamic behavior of the product that would otherwise require the reading and understanding of lengthy descriptions.

When prototypes are used, it is important to consider the prototyping culture within the organization.[8] Because prototypes are relatively easy to produce, it is possible to prototype designs that are nearly impossible to implement. Whenever the person creating the prototype is not the same person implementing the final product, control issues may arise. The specification can serve as a middle ground. The GUI designer owns the prototype, the specification is generated from the prototype (Schrage calls this "prototype-driven specification"), and the resulting specification is owned by the implementer. This ensures that the person responsible for building the product has control over it.

Information development

Ideally a design team includes a member whose main responsibility is information development for the product. In practice, however, this may not always be possible. If the information developer does not participate in the product design phase, then he or she can use the prototype for learning about the product and thus get an early start in producing the required documents (a.k.a. information deliverables). Completion of the documentation can have a positive ripple effect on activities dependent on it—translations go out earlier and documentation-based training starts earlier.

One potential drawback of this approach lies in the "fluid" nature of the prototype and the likelihood it will continue to change. When the prototype undergoes repeated changes, the information developer works with a moving target. Any advantages gained in an early start are canceled by the amount of rework needed.

Marketing support

Another non-traditional use of prototypes is in support of product demonstrations for marketing ("a picture is worth a thousand words"). Complex specifications or extensive verbal descriptions of a product cannot compete with prototypes in succinctness or effectiveness. Although most usability testing prototypes are not in condition to be shown to a customer, additional effort in cleaning up the prototype could produce an extremely effective sales tool.

Although product-level code can often be used for demonstrations, prototypes are available much earlier in the product life cycle. In addition, because the prototype is typically written in a higher-level language (Microsoft Visual Basic or Borland Delphi) and has significantly fewer lines of code, it tends to be more stable than alpha- or beta-level code (the alpha test takes place in the laboratory; the beta test that follows takes place in a user environment).

A prototype that starts as a faithful representation of the design could, over time, diverge from the final product, thus causing difficulties for the marketing team. In addition, because of the normal pressures that sales teams work under, and their separation from the design team, there can be a

temptation to oversell a prototype. If expectations are not appropriately set, the customer might erroneously expect product features not available in the final product. Also, because a prototype rarely performs any real processing, it may appear to perform better than the real product. The customer has to understand that the prototype is tentative, and its features and performance may differ from those of the final product. Another danger involved with early demonstrations is the possibility of the design falling into the hands of a competitor (the prototype has "escaped").

Function test

With automated testing becoming an increasingly important quality assurance tool, prototypes provide the opportunity to shorten the product life cycle. Instead of waiting for the final product to be released, the function test team can use the prototype to create test cases. Although the size of screens and the position of buttons may change, the tab stops and keyboard shortcuts can be set up to match that of the final product. This can save a significant amount of time during the testing phase when schedules are typically tight.

The incremental cost of adding in the appropriate keyboard functionality of the product (typically not required in most usability testing) has to be weighed against its benefits. Another consideration: the extra work of defining the action of the tab-stops, keyboard accelerators, and mnemonics can force the designers to consider earlier important accessibility features of their product.

Field support

Field support teams can make use of a prototype in two ways. First, they can use it to train and familiarize themselves with the product at a much earlier stage. Otherwise, they may have just a week to learn about the product before it is shipped. Second, the prototype can be made operational in cases when it is impractical to install the actual product. For example, the product may require special hardware, significant system resources, special connections, or more physical space than is available to the support team. Having a prototype operational while providing remote support for a customer can be very valuable.

The downside of using a prototype in this manner, of course, is that it must be considerably higher fidelity than typically required for usability testing. If the support staff is going to walk remote customers through tasks, the menus and labels have to be correctly reproduced. Unfortunately, the incremental effort needed to upgrade the prototype to this level can sometimes be greater than the benefit it provides.

Case studies

The following case studies, which illustrate how prototypes add value throughout the entire product life cycle, are based on our experience within the IBM Printing System Division. The first case study, which involves the development of a printer console, started with the use of a prototype for a usability study. The prototype developed for this purpose was later used for marketing and field support.

The value of the prototype as a training tool came to light during an episode involving a change of staff on the design team.

The second study involves a custom printing solution (a job-ticketing application) for a large customer. Although the main objective of the prototyping effort was usability testing, the prototype was also used to validate customer requirements. The last case study involves the development of a new industrial printer that was based on new vendor-supplied technology. The prototype served to validate the design specifications and was the focal point of a project that involved a multidisciplinary team from two different companies. The extended use of prototypes in these projects was not preplanned, but rather developed naturally as members of marketing, development, test, and field support teams recognized the benefits they could derive from the available prototypes.

In all of these case studies there are two types of teams. The main design team is a comprehensive multidisciplinary team that includes information developers, programmers, graphic artists, project managers, testers, usability engineers, vendors, customers, and field support personnel. Representatives from all these specialties are needed in order to en-

Figure 2 IBM Infoprint® 4100 Advanced Function Printing System

sure their input into the design at an early stage. For working sessions, however, the entire group is too large for active participation. Thus, a subset of this group, referred to as the "core" group, conducts working sessions and performs the detailed tasks. The resulting plans and designs are then presented to the main team for review. This mode of operation allows the work to proceed at the speed of a small group, backed by the thoroughness and skills of a large multidisciplinary team.

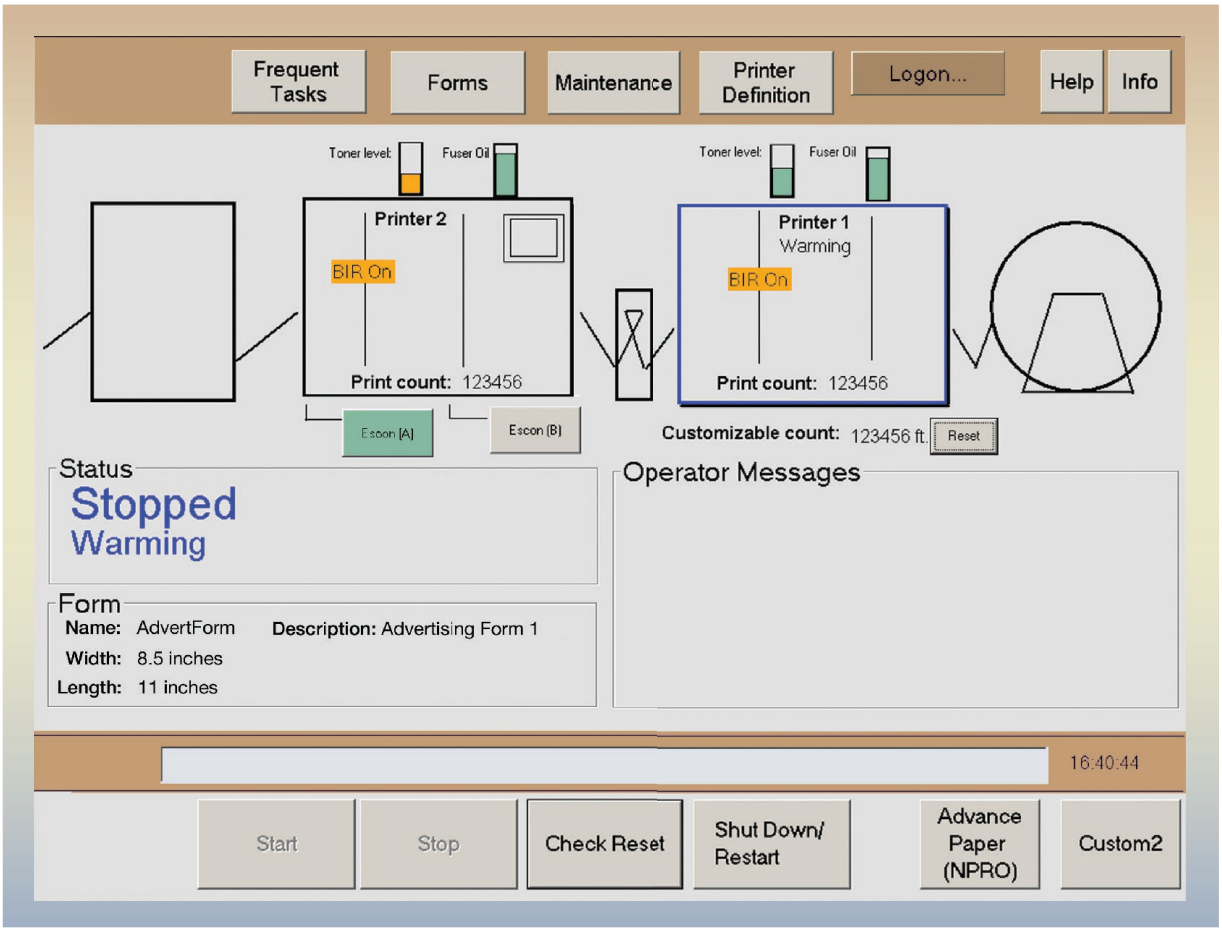
Operator console for IBM Infoprint 4100. Figure 2 shows the IBM Infoprint* 4100 Advanced Function Printing System, an industrial printer that prints up to 1000 pages per minute. The operator console, which had to be redesigned, includes a fairly complex GUI application and a 15-inch touch screen panel display. Although the new console interface inherited some functions from its predecessor, it was being completely redesigned, and in many ways it was a new application.

The multidisciplinary team consisted of developers, usability engineers, customer support personnel, testers, information developers, and graphic artists. Although several of the team members had worked together in the past, most of the group had not. The core design team, which varied in size from two to four, consisted of developers and usability engineers. During the design sessions, whiteboards were initially used for illustrating the design. Then components of the prototype were developed using Microsoft Visual Basic. Typically the group would meet and discuss the design; following the meeting the changes to the design would be prototyped; in subsequent meetings the changes would be reviewed. As the design progressed and further changes involved gradually less work, some of these were made extemporaneously during the meeting.

After the prototype was complete, the first round of usability testing began. Three print operators were selected from within the company, and guided through a number of basic printing tasks to be performed on the prototype. When problem areas were uncovered, these were noted. A second round of usability testing followed, which involved several customers from different market segments. In between visits to customer sites and on completing the tests, the team returned to the laboratory, discussed the findings, and made appropriate design changes.

The value of the prototype came to light during an episode involving a change of staff on the design team. During this design phase one of the information developers was replaced by another. The new team member needed to train quickly because the design had stabilized to the point where the writing of the documentation was about to start. The pro-

Figure 3 The operator console "keep-it-ugly" prototype



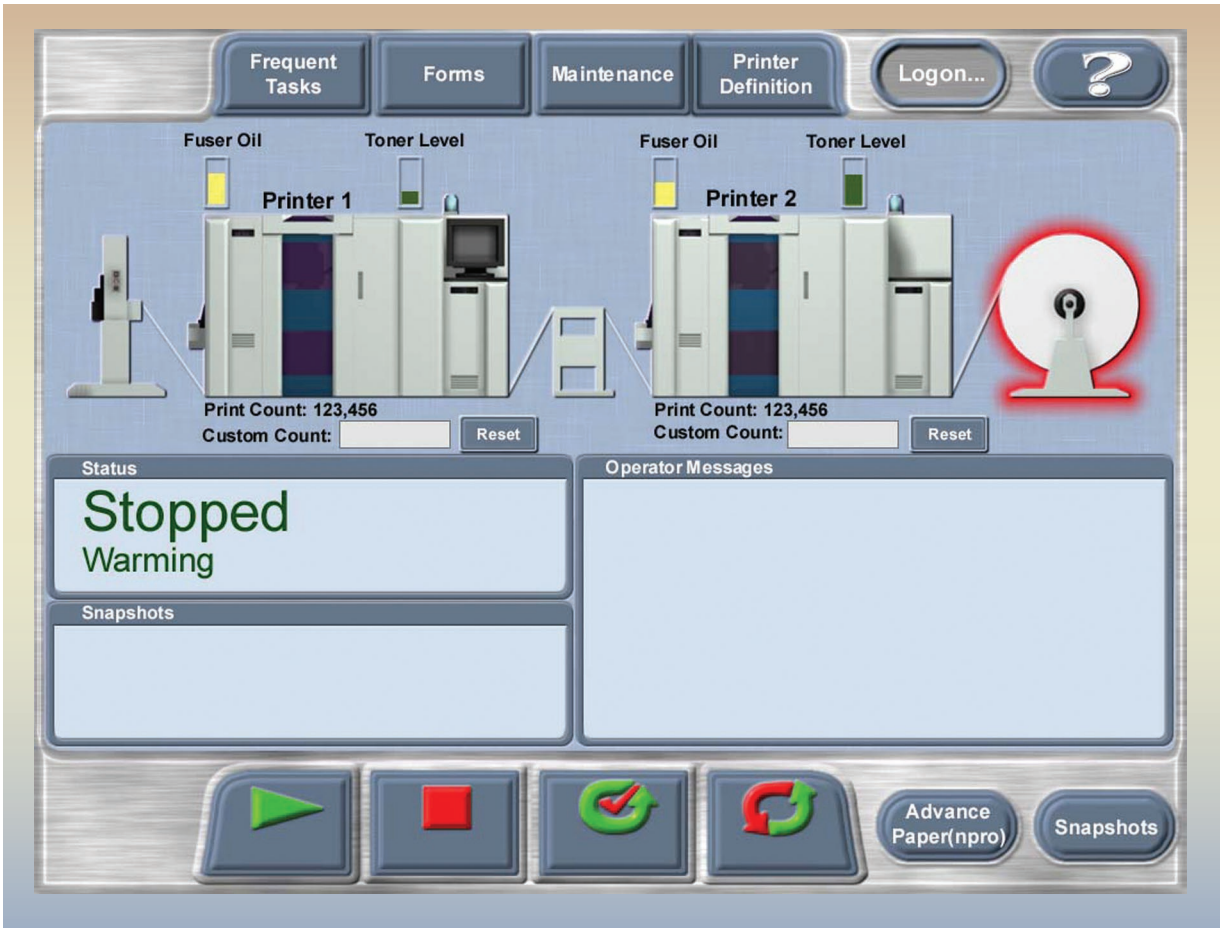
totype proved invaluable in helping the new information developer carry on the work.

Concurrently with this work, another ongoing project had the goal of developing a scaled-down version of the interface for a different class of printer. Within two weeks the team was able to create a new prototype and test a derivative interface. A field support manager suggested that the prototype be made available to the field Customer Engineers (CEs) responsible for installing and repairing the printers so that they could become familiar with the interface before the product was released. A version of the prototype for the laptops used by CEs was created and shipped together with the associated training manuals.

Toward the end of the project, before the implementation of the product was complete, the marketing staff requested help to set up a product demonstration at a trade show. Because the product-level code was not available, the prototype was upgraded and enabled for the demonstration. The GUI was upgraded from the keep-it-ugly stage, shown in Figure 3, to the more polished look shown in Figure 4. The prototype was also included in a product demonstration CD-ROM to be used by marketing staff in early product demonstrations.

Near the end of the development cycle, a substantial transaction involving a sale of several of these printers to a corporate customer was being negotiated. The customer was familiar with the previous

Figure 4 The operator console marketing prototype



version of the operator console and raised questions regarding its usability. It became essential to convincingly demonstrate to the customer that the new printer (still under development) had excellent usability properties. A meeting was held in which the prototype of the new operator console was demonstrated, and it met the customer's usability requirements. Moreover, the session helped validate requirements, and some additional suggestions made at the time were later incorporated into the product.

Because the printer is bulky and costly, it is not practical to provide a printer to all the field support personnel. An operating prototype, however, helps the CE about to go out on a customer call. The CE can use it as a learning tool—a quick interactive session works as a refresher course. In addition, the field support team recognized the value of using the prototype when responding to telephone calls from customers—they use it to talk customers through the interactive behavior of the GUI application. In fact, it was discovered that an unofficial version of the prototype was operating and in use by the field support team even before its release by the development group.

The impact of this prototyping work was long-lasting throughout the organization. After the value of using prototypes in various phases of the product life cycle was demonstrated, these uses became embedded in the standard operating procedures of the various departments.

Job-ticketing application. The goal of this project was to produce an Adobe Acrobat** plug-in that enabled users to specify the paper on which documents should be printed, the type of stapling to be used for the printed documents, whether covers should be added, and so on. In the trade this is known as "ticketing" the print job. The job-ticketing application was being developed as a custom solution involving a single IBM customer. The customer was very knowledgeable about the ticketing process and had many ideas about the design of the application. The multidisciplinary design team for this project included an architect and subject matter expert from the customer's organization, and development and usability personnel from IBM.

First, the requirements had to be converted into something the customer could visualize. The prototype enabled the design team to validate that their interpretation of the customer requirements was correct. During early design sessions with the customer, the prototype was one mechanism that helped the parties communicate.

It was common practice for the design team to list aspects of the design on a whiteboard in one meeting and then overnight convert these into a working prototype. The next day the design team would walk through various scenarios using the prototype, new ideas would be drawn up, and the cycle would repeat. The process of collecting design ideas and converting them into something that could be perceived and experienced helped ensure the product would meet customer requirements. The prototype developed was a highly functional prototype created with Microsoft Visual Basic.

Next, usability tests were undertaken. It helped that the customer had a sophisticated attitude toward usability testing and provided the design team with resources, including access to approximately 40 users at 12 locations around the United States. The design team installed the prototype at customer locations and had the target audience use the prototype as if it were the final product. Although not fully functional, the prototype was integrated into the customer workflow. When jobs arrived through the existing workflow, the prototype was used to set up the print job. This approach enabled the design team to evaluate the performance of the prototype with a wide variety of job types and thus ensure that the proposed solution would integrate with the existing system.

The prototype also enabled the design team to demonstrate the proposed product to two other target audiences, the customer sponsors and field managers as well as the technical community at large. This opportunity arose at an annual technical convention.

Industrial printer with vendor-supplied technology. This project involved the development of a new industrial printer based on new vendor-supplied technology. The IBM team had little experience with the new technology. The vendor had a great deal of expertise with the new technology, but not much experience with print technology. The main challenge was to help the two teams communicate so that everyone's assumptions were apparent.

The multidisciplinary design team consisted of a project manager, the vendor team, application developers, print quality experts, graphic designers, usability engineers, and information developers. The core design team consisted of the usability engineers and the application developers. The teams included U.S. as well as non-U.S. participants.

The project manager initiated a number of meetings with the vendor in which a prototype for the operator console was used to communicate the team's expertise in the areas of the market segment, data streams, and controller design. This prototype was derived from that described in "Operating console for IBM Infoprint 4100" and adapted to the new technology. Working with the prototype forced the design team to focus on how the new technology could fit into the existing architecture. The conversations that ensued helped validate the assumptions and put the design on a solid basis.

A project involving a multicultural development team usually requires working through language and cultural barriers. In this project the use of a prototype was very helpful because it allowed the participants to have a common visual representation of many of the concepts under discussion.

Following the meeting, a low-fidelity paper prototype of the printer hardware was developed and used in conjunction with the operator console prototype. Even though there were still many gaps in the design, it was possible to put together some basic functions, which enabled the team to start looking at a number of scenarios. These prototypes allowed us to perform a rough walk-through of user tasks much earlier than typically possible in the hardware design cycle (this took place even before the first hard-

ware prototype was operational). The team was able to determine early the location of covers, consoles, and view ports, a task that would have been costlier if performed later in the development cycle. Also, the walk-through using a rough paper prototype generated numerous discussions as it was discovered that the participants had different assumptions on how parts of the design would function. Discussions on how the user would use different printer functions may not have occurred if the design team had not walked through the prototype. Before the prototyping sessions the team members had different understandings of the design. Using the prototype, they were able to compare their individual visions and thus reconcile them into a common shared understanding.

Make the prototype disposable– do not use it in product-level code.

Conclusion

In this paper we discussed various ways in which prototyping can be extended beyond usability testing. The three case studies presented do not exhaust the possible opportunities for the use of prototyping within the product life cycle. Rather, they were meant to provide a sampling of such opportunities derived from our own experience. Although our experience comes from the printing industry, the extended use of prototyping is clearly applicable to many other areas.

Guidelines for extended use of prototyping

These guidelines are useful for those considering the extended use of prototyping:

Polished is not always best. If you are using a GUI prototype to get feedback on the design, then making it look too polished might reduce the number and types of comments that you receive. Your audience may assume the design is complete and might not offer creative wide-ranging suggestions.

Use low-fidelity paper-and-pencil prototypes for non-interactive applications. Using a paper-and-pencil prototype is often satisfactory when the interactive aspect of the system is secondary (e.g., a wizard). On the other hand, if there is a GUI involving menus, dialogs, and other dynamic behavior, then a higher-fidelity prototype, possibly created with a prototyping tool, is most likely the right solution.

Choose depth or breadth, as appropriate. Sometimes the prototype has to present an encompassing view of the system (horizontal prototype). Other times the prototype has to probe in depth a limited set of functions (vertical prototype). Make the prototype broad enough to cover the aspects of the system that you are concerned about, but ignore the less relevant items.

Make the prototype disposable (do not use it in product-level code). When you are working under a tight schedule, it is often tempting to include parts of a working prototype in the product-level code. In most cases this is not a good idea. When a prototype is created, the main goal is to create the illusion of a working product as quickly as possible. Thus, the implementation often does not have the required attributes of efficiency, maintainability, and reliability.

Set the expectations of the audience appropriately. Before presenting a prototype, make sure the audience understands what a prototype is, why it was created, and how the design is likely to change. If the prototype is high on the implementation scale, let the audience know that the product may have a different user interface. A prototype high on the role scale may have a rough appearance. If the prototype is high on the look-and-feel scale, let the audience know that they should expect limited function.

Use the prototype to determine the product specifications. This has been suggested by Schrage,[8] and our experience confirms it. Premature specifications may lead to rigid suboptimal designs. Instead, allow several iterations of prototyping to drive the specification. After the prototype is stable, then the specification can become firm.

Manage access to the prototype. Considering how quickly prototypes can be modified, it is important to ensure that colleagues outside the design team (e.g., in marketing, field support) understand the extent to which the prototype is likely to change. In some instances distribution of the prototype should be limited until its design stabilizes.

Label sales prototypes as such. If a prototype is polished enough to be used for sales demonstrations to customer staff and executives, make sure that it is clearly labeled as a prototype. A fully functional prototype may give the erroneous impression that the product is close to release. Place PROTOTYPE labels at various locations to ensure that everyone understands what they are being shown.

Plan frequent prototyping cycles. Plan frequent iterations of your prototype, as short as hours or days if possible. If the cycle is a week or more, your prototyping medium is not flexible enough.

Acknowledgments

We thank the anonymous reviewers for comments that enabled us to substantially improve the paper. We also acknowledge the creativity and the efforts of the design teams that participated in the work described here.

*Trademark or registered trademark of the International Business Machines Corporation.

**Trademark or registered trademark of Microsoft Corporation, Borland Software Corporation, or Adobe Systems Incorporated.

Cited references and notes

1. S. Houde and C. Hill, "What do prototypes prototype?" in *Handbook of Human-Computer Interaction*, M. G. Helander, T. K. Landauer, and P. Prabhu, Editors, Elsevier Science, Amsterdam, The Netherlands (1997), pp. 367–381.
2. B. Tognazzini, *TOG on Interface*, Addison-Wesley, Reading, MA (1992).
3. C. Lewis and J. Rieman, *Task-Centered User Interface Design* (1995); available at *http://www.hcibib.org/tcuid/index.html*.
4. J. Nielsen, *Usability Engineering*, Academic Press, San Diego, CA (1993).
5. K. Vredenburg, "Building ease of use into the IBM User Experience," *IBM Systems Journal* **42,** No. 4, 517–531 (2003, this issue).
6. K. Vredenburg, S. Isensee, and C. Righi, *User-Centered Design: An Integrated Approach*, Prentice Hall, New York (2002).
7. K. Vredenburg, Editor, *International Journal of Human-Computer Interaction* **14**, Special Issue: Designing the Total User Experience at IBM (2002), pp. 275–558.
8. M. Schrage, "Cultures of Prototyping," in *Bringing Design to Software*, T. Winograd, Editor, Addison-Wesley, Reading, MA (1996), pp. 191–205.

Accepted for publication May 9, 2003.

Ron Van Buskirk *IBM Printing Systems Division, 6300 Diagonal Hwy, Boulder, CO 80301 (ezprint@us.ibm.com).* Ron Van Buskirk is a User-Centered Design lead in the IBM Printing Systems Division. He received an M.S. degree in industrial engineering from North Carolina State University in 1994. He is inventor or co-inventor of 30 user interface patents. His interests include prototyping, accessibility, speech recognition, and user interface personality.

Brian Moroney *IBM Printing Systems Division, 6300 Diagonal Hwy, Boulder, CO 80301 (bmoroney@us.ibm.com).* Dr. Moroney joined IBM in 1999, after receiving his Ph.D. degree in psychology (human factors) from the University of Cincinnati. He worked as a User-Centered Design lead on multiple projects and now manages the User Interface Design and Development department in Boulder, Colorado. His interests include usability, user interface development, user interface consistency, and prototyping.