JAMES RUDD AND SCOTT ISENSEE, IBM

Twenty-Two Tips for a Happier, Healthier Prototype

Three in A Box /Susan Todd

Never before has the human factors professional had the opportunity to take a lead role in software development. Enabled by a vigorous focus on software usability and the availability of a number of robust prototyping tools, corporate human factors groups are making significant positive contributions to software development through software prototyping. However, a successful prototyping effort requires more than a prototyping tool and a background in user interface design. It has been our experience that the success of a prototyping effort is dependent on many factors, some obvious, some not so obvious, and others learned only through postgraduate training at the School of Hard Knocks. Put your feet up, grab a cup of coffee, and let us share with you some of the lessons we've learned on the way to happier and healthier prototyping efforts. In no particular order:

Obtain upper-level development management support

Don't waste your time trying to convince the programmers that prototyping is a good idea. Don't waste your time convincing low-level programming managers that prototyping is a good thing. For that matter, don't even waste your time convincing your management that prototyping is a good thing. The people you have to convince are those that hold the purse strings for product development. Without their support, you will never be successful. The very best way to do this is to invite them into your lab, tell them about the benefits, relate success stories, and top the pitch off with a demo of your prototypes. We have found this to be the most effective way for communicating the power of prototyping and (as a very positive by-product) generating interesting, challenging work.

Throw away your prototype

That's right. Tear it into little pieces and start again. Don't expect it to be right the first time. You don't know your audience well enough. You do not fully know the limits of the prototyping tool. You will not know the application inside and out, and you don't know whether the developers can implement what you have prototyped. Prototyping is an iterative process, and you are going to learn as you go along. Leave enough time in your schedule to make radical changes based on the feedback you receive.

Make prototypes with high fidelity

We have found that high-fidelity prototypes are more successful at convincing management and development that a particular user interface approach will work. The higher the fidelity of the prototype the greater the perception that the design approach is feasible and the more likely it will be accepted. Don't be lazy. What's intuitive to you may not be to the developer. If you have the time (and we recommend that if you don't have the time, make the time), add minutiae to your prototype. Add error messages and help panels. Make sure that every path leads someplace. Not only does it improve the perception of feasibility, but it lets people know that you have studied the problem in depth and have considered each of the user interface issues.

Take every opportunity to show the prototype

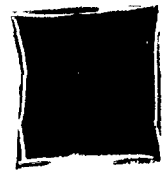
By showing the prototype to as many people as possible, you build a case for the benefits of prototyping and the feasibility of your approach. It gets people excited. No matter how busy you get and no matter how many deadlines are imminent, when asked to demo the prototype to somebody, do it. Not only do you understand the prototype better than anyone else, but you will be better able to answer any user interface question that may come up. If the prototype is good, it will only be a matter of time before Marketing will "relieve" you of the burden of demonstrating the prototype. Take advantage of the exposure. Fame is fleeting.

Don't waste your time prototyping add-ons

At some point in time, we all are assigned to point releases of products. By the time this occurs, the excitement of product development has worn off, and all major design changes have been decided upon. You are left with the crumbs and the "B" team. The name of the game is influence. If given the opportunity, move on and take an influential and lead role in the development of new products.

Start early

You cannot start early enough. We have found that an initial draft of the prototype should be available before the product objectives are published. Don't wait for the product requirements to come out. You should be defining the requirements. If you have some background in the application area, take your best shot at defining a set of requirements, prototype it, and then go to the customers to find out how close you are to providing what is really needed. The initial prototype serves as a straw man to get the customers thinking and talking about their requirements. They may tear it apart, but at least you know where they stand and what they want. We have found straw man prototype critiques to be a great mechanism for generating customer involvement and commitment in our development efforts.



Make the prototype the functional specification

When the prototype accurately represents what the customers are looking for, it provides a specification for the developers to code to. The old style of functional specification typically contains hundreds of pages of obtuse technical detail. It is impossible to fully understand and review. A prototype provides a "living spec" which the reviewer can see, touch, and play with. Prototyping can greatly increase efficiency of the development process by eliminating reviews of obtuse specifications only a small percentage of reviewers read and understand anyhow. If you are not able to get rid of the written specification, at a minimum, the prototype supplements the written word. At IBM, we use the saying: "A picture may be worth a thousand words, but a prototype is worth a thousand pictures."

Disseminate to all technical leaders and developers (including marketers, technical writers, planners, and others)

Everyone involved in development of the product has a different perspective and can provide valuable input. The product should reflect all of these viewpoints. You need only miss the mark in one area for a product to fail. Asking for their advice can yield a sense of commitment to the product, which fosters teamwork.

Develop idealistic instead of realistic prototypes

Be idealistic in the prototyping which occurs early in the product development cycle. Add as many features to the prototype as you can. Even though you know some may not be feasible for the release, it is important to get these ideas in front of marketers, developers, and customers for a number of reasons. First, customers should determine what is important to be included in the first release. Development priorities on our products have often been modified by customer requests. Oftentimes the customers didn't realize they needed a new function until they saw it in the prototype. Second, the new features get the developers to stretch the envelope. If we can prototype it, they have a harder time arguing it is impossible to code. Third, it's good PR for the company. It convinces customers that your company is serious about producing best-of-breed software. Fourth, it builds up the reputation of human factors. We become the venerated advanced user interface technology people instead of the overpaid, prima donna traffic cops we are sometimes perceived as.

Use the best tools

The ideal tool is quick to learn, fast to prototype with, high in function, executes like lightning, and produces code to be used in the real product. Unfortunately, this ideal tool doesn't exist yet. You must pick a prototyping tool which has the functions most appropriate for your prototyping effort. The tool you select will have a significant impact on the success of your prototyping effort.

Grab a piece of the action

Take on responsibility for part of the project, if necessary. For example, on one of our projects, we volunteered to write the installation program and sample set because the development group did not have the staffing to meet our specifications. By taking on this responsibility we assured our company produced a better product; learned more about programming, our product, and the development process; got an appreciation of the problems developers face, and gained respect from the developers by showing we could work in their arena. It's difficult to be everything to everybody, and we are not advocating that position. However, sometimes volunteering for work that you normally do not do pays off in handsome ways.

The customer is king

Cash may be king to Donald Trump, but for a prototyping effort to succeed, customer involvement is essential. You can perform task analyses until you are blue in the face, but you will never know as much about the job as the customer who performs it day in and day out. We found repeatedly during development of our prototypes that customers made suggestions we never would have thought of. Customers are a powerful driving force in setting requirements. When they see a function they like in the prototype, they can demand it be included in the product.

Look outside the US

In many US companies, we are accustomed to developing products for the US market. We expect them to be used outside the US with little change other than language translation, However, the marketplace is becoming increasingly global. The world market is a market you can't ignore. Differences between countries such as laws, education levels, and customers influence the design of software. Talk to customers and members of your line of business in other countries.

Keep control of the prototype in your shop

Along the way, others outside your department may volunteer to do some of the prototyping for you. Adroit technical people know a good thing when they see it. You might be behind in your work, or you might be interested in improving your working relationship with those that volunteered. Sounds like a good idea. Doesn't it? If you are even slightly concerned about your place in the design process, our recommendation is an emphatic DON'T do it. By letting others from outside your department in on even a small portion of the prototyping work, you give up control of the user interface maybe not on this application but potentially on the next. Your contribution is no longer unique; constraints like cost, schedule, and coding complexity begin to subtly influence the design; you begin to slip into the old role of consultant rather than owner. In addition to this, you are doing the product a disservice since, most likely, the others working on the program such as developers or marketers will have limited user interface training. It is imperative that you keep control of the prototype in your shop.

Pay attention to aesthetics

In demoing to customers, we found it important that we use professional-looking artwork, good screen design, and pleasing color selections. This gave the customer an initial, favorable impression of the prototype, helped convince them to look at it seriously, and allowed us to move past the aesthetics to concentrate on the ease of use and function questions which are the bread and butter issues of our profession. If you have at your avail a graphics design department, use it even if it isn't staffed by computer-literate people. This resource will save you many hours and much heartache arguing over aesthetic issues.

Don't delegate the prototyping

It is tempting to define the requirements for the prototype and then get an intern or a programmer to implement it. We recommend against doing this if possible. This breaks the feedback loop that comes from getting an idea, implementing it, showing it to customers, and cycling through again. We quickly learned which ideas could and could not be implemented by our prototyping tool. We also got ideas by writing the prototype which we would not have otherwise. In addition to this, prototyping it ourselves forced us to become quite familiar with the nuances and intricacies of our corporate user interface design guidelines. Our experience with assigning prototyping work convinced us of the truth of the old programming adage that the most elegant designs are the product of a single mind.

Become multidisciplinary

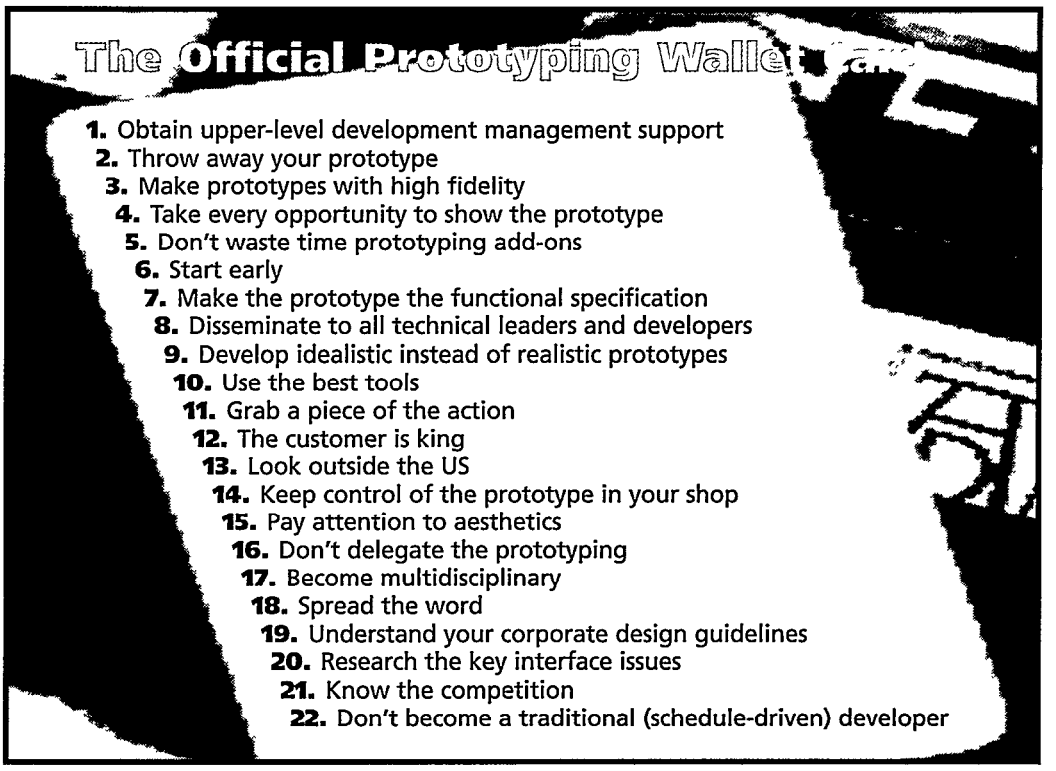
We are doing human factors work on programming tools for the banking industry. We work very hard trying to keep up with recent developments in all three domains (software usability, programming, and banking) and believe it has contributed positively to our prototyping efforts.

Spread the word

Human factors is changing. The profession is moving from hardware to software, from testing to prototyping, from support to ownership. As you learn new things, it is important to pass this knowledge along to others through journal articles, conference presentations, technical interchange groups, and in-house publications.

Understand your corporate design guidelines

It isn't sufficient just to read your design guides. They are incomplete, imprecise, and represent a single time slice picture of a moving target. At IBM, our corporate design guide is called "Common User Access (CUA)." To stay current with CUA, departments keep up with the electronic bulletin boards, send a representative to CUA meetings, and talk to the people responsible for CUA. Know what the rules are today and what they will be by the time your product hits the market. Be informed enough to challenge the rules when you don't think they're right.



Research the key interface issues

Prototypes allow us to do our testing up front, before the program is written. Rather than just looking for problems, we can design and evaluate multiple alternatives. We see early prototyping efforts as a vehicle for performing advanced-technology EUI studies. For many, this will be a very rewarding side-effect of early prototyping efforts. In Fact, these early prototyping efforts can often identify patent idea

Know the competition

You can't compete in the high-jump until you know the height of the bar. No matter how brilliant and creative you are, there are important lessons you can learn from the competition. Get to know the competition by purchasing and experimenting with their software, and visiting trade show exhibits.

Don't become a traditional (schedule-driven) developer

You know, developers, those nasty little trolls who hide in dark corners throwing out curses like "cost" and "schedule." As human factors prototypes become a key part of the development process, we come under some of the same pressures. Be careful not to use cost and schedule as excuses for not producing the highest-quality prototype possible. Wear a clove of garlic around your neck and (if you are familiar with the Motorola approach to quality) inscribe it with mysterious Greek incantations like "six sigma." **H**

James Rudd joined IBM as a human factors scientist in 1988. He designs, prototypes, and develops application software for the banking industry.
email: jrudd@cltvm3.iinus1.ibm.com.

Scott Isensee is an advisory human factors engineer at IBM in Austin, Texas. He is designing a desktop to be used through the UNIX world.
email: isenesee@austin.ibm.com.