Designing for Designers: An Analysis of Design Practice in the Real World

Mary Beth Rosson
Susanne Maass
Wendy A. Kellogg

IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598

Abstract

Twenty-two designers were interviewed about their design of interactive systems. They were asked to select a recent project having a significant user interface component, and were probed about the general design process involved, how the design of the user interface fit into that process, and their personal strategies for exploring ideas. Analysis of their responses pointed to two models of the design process. The relationship of these models to the type of user testing done and the strategies used for generating ideas is discussed, especially with respect to the implications for developing tools to support design.

Résumé

Nous avons mené une enquête auprès de vingt-deux ingénieurs systèmes, à qui nous avons demandé d'analyser un projet de leur choix comportant une part importante d'interface utilisateur. Les questions portaient essentiellement sur la méthode générale de conception du système, l'intégration de l'interface utilisateur dans la conception générale du système, et sur leur stratégie personnelle de recherche d'idées. L'analyse de leurs réponses révèle leur préférence pour deux modèles de conception particuliers. Nous analysons la dépendance de ces modèles avec le type de tests à effectuer pour vérifier l'interface utilisateur, ainsi que les stratégies de recherche d'idées, spécialement dans le but de développer des outils de conception.

Keywords: Design practice, Tools for design, Usability engineering

Introduction

The design of interactive computing systems is a fascinating problem that has attracted considerable attention in recent years. The attention is due in part to a realization that the science of human-computer interaction is still in its infancy, and that we may never know enough about how users interact with computers to guarantee good system design. In addition, the design process itself is of interest, as another example of a complex, interesting human activity that might be supportable by advanced computing systems.

Research on interactive system design has taken two perspectives, recommendations regarding the *process* of design, and *tools* to support the process. Carroll, Thomas, and Malhotra [6] observed a real designer working on a problem, noting a number of contrasts in this process to the classic view of top-down hierarchical design. Gould and Lewis [10] proposed some basic principles to be followed in design; Boies, Gould, Levy, Richards and Schoonard [2] then analyzed a case study in which they applied these principles. Their work reflects a philosophy that has been widely promulgated: the design of interactive computing systems must proceed iteratively, and must involve early and sustained interaction with prospective users (see [1, 5, 9] for more discussion of *usability engineering*, as this design philosophy has come to be called.)

A complementary approach to design has focussed on the development of tools for designers to use. Given an iterative design process, we can attempt to provide tools that make this iteration occur as smoothly as possible. The proffered tools may simply represent a general programming environment [8, 18]; more recently, however, there has been great interest in the possibility of *user interface management systems*. These systems are intended to separate the development and control of the user interface to a system from that of its underlying functionality. The systems generally provide user interface development tools (e.g., menu layout, error and help management) as well as a runtime manager to handle the communication between user interface and application during actual system use (see, e.g., [4, 11, 15]).

The work we report here represents the beginning of an effort to combine these two perspectives on the design problem. We believe that good tools can have a great impact on the quality of interactive systems that are designed. But as for the case of interactive systems in general, we believe that the design of successful tools must begin with study of the target users and their tasks -- real designers and their design practices. The literature provides remarkably few examples of research in this vein (cf. [12, 13]). As our own starting point, we have interviewed a number of designers about their design experiences.

Interview Procedure

Our goal in this work was to gain as much information as possible about the design process, not to test formal hypotheses. We also wanted to get this information in the context of real world problems, making the introspective interview an attractive methodology.

In an effort to recruit as wide a range of designers as possible, we used two complementary procedures. Some designers were recruited via a notice on company electronic bulletin boards. The notice described our interest in design, focussing on interactive systems for unsophisticated users, and asked for short descriptions of the projects designers had worked on; this notice evoked 41 responses. We then selected projects that represented as much variety as possible with respect to application, operating environment, and user interface style. This method resulted in 12 designers. We identified 10 additional designers ourselves, both by requesting participation of designers visiting our research lab, and by following up on particular projects that were known to us. The final set included 17 designers who described designs done while working for IBM, and 5 from other organizations.

The interview was structured into four main sections. In the first we obtained information about the designer's *background* (job, length of design experience, operating environments and design projects). The next section asked about the *general design process* associated with the project they had selected (design goal, size and coordination of project, identifiable stages in the work, and key factors influencing the outcome). The third section focussed on the *user interface* component (where it fit into the process, what constraints were felt, what principles drove its design, to what extent consistency was an issue, what parts of the interface were most difficult to design or implement, what tools if any were employed, what kind of user testing was involved, and what problems were identified after completion of the system). The final section requested designers to introspect about *idea generation* (how they got ideas in the first place, how they refined these ideas, how they tracked idea status, and what might make idea generation easier or more satisfying).

Over half of the interviews (13) were conducted over the phone. Regardless of interview medium, the questioning was always conducted by a single experimenter, with one or more others present to follow up on answers that seemed unclear, or on comments that seemed particularly interesting. Interviews were recorded and later transcribed with assistance from notes taken during discussion. An interview typically took 1.5 hours, but some took as much as 2.5 hours.

The Design Population

Participants' current positions ranged from programmers, software developers, and technical staff, to researchers and university professors. Experience in design varied from 5 to 26 years.

Of the projects described, 13 were designed for mainframe or mid-sized timesharing environments and 9 designed to run in an intelligent workstation environment. The projects represented a wide variety of applications, including *office support* (general personal services, electronic mail, pop-up typewriter), various types of *tracking systems* (site performance, facilities and budget planning, inventory control, education management, manufacturing control), *information or function access* (on-line information retrieval, multi-application interface), *personal computer services* (disk maintenance, error diagnostics), an online *tutorial*, a *graphics* system, and *software development support* (UIMS, program library management, visual programming aids, code-generation support). The size of the projects ranged from individuals working alone to a group of as many as 12-14 during system implementation; project length varied from a few months to as much as 8 to 10 years. Projects also varied with respect to their business goals, with five scheduled for release as external products, two as internal products, eight as local Information Systems (I/S) support tools, two as research versions of future external products, and five as research projects with no commercial plans.

Interview Findings

The interview methodology produced a large amount of qualitative data, not only about the process of design, but also designers' beliefs about how user interfaces ought to be designed. In this brief report, we will summarize salient features of the design processes described to us, and their implications for how we might best support the process; another paper [14] describes some of our findings regarding designers' beliefs about user interface design.

The General Design Process

Of particular interest to us were designers' comments regarding design iteration and user testing; these are important tenets of the emerging philosophy of usability engineering, and we wanted to understand their role in these actual design accounts.

Iteration in design.

Our participants were almost evenly split in the process they followed during design and development. Ten of them described an *incremental development* model, in which design and implementation of the system occurred simultaneously in a highly iterative fashion. The other twelve described a *phased development* model, in which there was a design phase followed by an implementation phase, with some sort of evaluation marking the point between design and implementation. However, within this group of twelve, three designers noted that the design phase itself had been iterative, in the sense that prototypes had been generated, tested and revised; the others indicated that the more traditional "design on paper" approach had been used.

As one might expect, design model employed was very much a function of the business status of the project: all of the projects scheduled for external or internal release as products were associated with the more tightly controlled phased development model, whereas all of the research-oriented projects followed the incremental development model. Designers working on I/S support tools were split, with three choosing the incremental model and five the phased model.

Another factor associated with design model was the environment in which the work was done. The hardware component seemed relatively unimportant, with mainframe and intelligent workstation projects falling into both model categories. However, projects undergoing incremental development tended to use interpretive languages: seven of the 10 cases used an interpretive language; for the three other examples, one used an incrementally compiled language, and the other two relied on special tools for iterating on just the user interface. Notably, in the three cases of design iteration within the phased model, the iteration was done in a different, interpretive language than the final implementation. In contrast, for the 10 cases of non-iterative phased development, only two projects chose to use an interpretive language.

Our analysis of these design descriptions suggests that there are two ways to think about prototyping as part of an iterative design process. In the incremental model, the prototype *is* the system, and the iteration that takes place ultimately *evolves* into the final system. At early points in development, any given function may be only partially implemented, while the designer explores additional function in parallel. In the phased model, iteration is limited to the initial design phase, and the prototype is a *simulation* of function that will not be implemented until later.

These two characterizations of prototyping in design (see also [3]), are of course idealizations. They do however allow us to understand some of the tradeoffs inherent in the two approaches. The evolutionary approach demands a situation permitting flexibility in the design result; the final design will be known only at the end of the process. It has the advantage that all design is done in the context of the target environment, so that system constraints are felt and dealt with all along the way. It also seems more efficient, in that the final system is being built throughout the process. In contrast, the simulation approach lends itself to a situations needing tighter controls, because at some point the design is accepted and then implemented. Because the prototyping is a simulation, some system constraints may not be felt until the implementation phase, and this may induce compromises in the original design. On the other hand, in cases where the target environment is complex, an independent simulation tool might support more freedom in initial idea exploration. Further, because of the simulation status of the prototype, it may be possible to examine a greater breadth of function earlier in the process, and non-optimal ideas may be easier to discard.

User testing.

Most designers described some sort of user contact in their projects, varying from active user involvement in the generation of design requirements to rather belated field tests once a system had been completed. In general, though, most testing was informal (with no special attempt to select representative users and representative tasks) and occurred relatively late in the development cycle. Six designers made an effort to talk to users in advance, some having at least one end-user on their design teams. Interestingly, all of these projects were I/S support projects; perhaps because the target audience in such cases is a well-defined and accessible body, early interaction with users is a more natural part of the process.

Only one of these six projects followed this initial user input with user testing on early prototypes; two others provided demonstrations to users for their comments. In addition, two projects that began without any initial user study tested interactive prototypes; two more built demonstration systems for user reactions. In all of these cases, the designers indicated that the early user input had been very useful, in contrast to designers who were provided with the results of human factors testing, or field tests, late in the process. Many designers specifically mentioned inadequate information about their users' needs, or an inability to do early testing with users, as a major problem in the design of their system; reasons given for these problems included user disinterest, lack of prototyping tools, lack of resource, confidentiality of the product under development, and problems with the group assigned testing responsibility.

The nature of the user testing done varied as a function of the business goal of the project. In all cases where the final outcome was to be a system used in a business setting, at least some form of evaluative testing occurred somewhere in the process -- sometimes both early and late in the cycle, but most often in the form of an internal or external field test after a first version had been implemented. For the research projects, user contact was much less evaluative in nature, being seen more as a "show and tell" process that might lead to additional interesting ideas.

A surprising finding was that the likelihood of early user testing was not related to the design model being followed: designers using incremental development were no more likely to offer an early interactive prototype to users than were those using the phased approach.

This observation points to another possible tradeoff between the two types of prototyping described earlier. Several of the designers using the incremental approach commented that one of the main reasons for not bringing users in to interact with their early prototypes was the system's lack of robustness; implementing existing function to a level adequate for usability testing would have taken time away from exploring new function. A special-purpose tool designed explicitly for simulating enough function to support user scenarios might have helped these designers to better assess their design as they progressed, as well as providing a critical exploratory evaluation tool for designers using the phased approach.

Design of the User Interface

Recently, there has been a good deal of discussion about the benefits of separating the user interface of a system from the rest of system functionality. The argument has been that this sort of modularity may aid system design in a number of ways, by promoting iterative development of the user interface and perhaps making it possible for experts other than application programmers to develop the interface [19]. In our interviews, we asked designers if the user interface to their projects had been designed or implemented separately from the rest of the system.

The replies we received were quite interesting. Eight designers acknowledged such a separation, with all but one indicating that the user interface had been considered first. Eleven others indicated that the user interface had not been considered distinct from the rest of the system during design; many seemed to have real difficulty in even imagining how such a separation might apply to the system they had designed, and a few made strong statements about the inadvisability or impossibility of making such a distinction. Two others indicated that while there was no distinction initially, they began to see and make one as the design progressed.

An important distinguishing characteristic between the designers making and not making a user interface distinction was the extent to which system functionality was understood in advance. So for example, in the group treating the user interface as a separate component, two acknowledged explicitly that the function was known before beginning; two others were developing systems that were essentially new interfaces to existing systems; the other four were developing systems to support function with which they themselves were very experienced. In contrast, the other group of designers was working on systems for unfamiliar user sets or on function that was breaking new ground. The two designers who had described a change in the way the interface was viewed indicated that as the design progressed, they began to recognize common operations that could be modularized in the user interface.

These findings raise intriguing questions about the meaning of "user interface," especially in the context of design. While it is quite common for researchers in the field of human-computer interaction to think of the user interface as simply the dialog between the user and the system, people engaged in system design often balk at this distinction. For them, the user interface is "what the user does", and this includes not just dialog, but dialog *with* something -- the system function. The user's perception of the system is an interaction of the function available and the procedures provided for accessing it, and it is this perception that designers are striving to optimize (see also the discussion of usability found in [7]).

The findings also point to questions about the generalized use of tools that guide the appearance and feel of the user interface. In some situations, a tool encouraging a particular style of menu layout and interaction techniques may be very appropriate -- situations in which the application domain is understood well enough to determine effective interaction styles in advance. But for novel designs, it may be desirable to use less specialized tools, ones that make no assumptions about interaction style, so that system function and the interface to it can evolve together. There was some evidence for this in our interview results: five designers reported the use of tools associated with a very specific user interface style (in all cases, a screen design and control facility); four of the five were ones who began with a clear understanding of the function to be provided. In contrast, the designers with more open-ended functional goals tended to work with rich, less constraining tools.

There remains a question as to whether we can guide the design of function as well as the interface to it. According to our designers, a critical aspect of getting the right function is a comprehensive understanding of the task domain and of the target users. A design tool in and of itself will provide none of this understanding -- the designer must hold and practice the philosophy of user-centered design. However, it may be that by providing comprehensive, easy-to-use prototyping tools for simulating user scenarios, we can make early interaction with users a more attractive component of early design.

Generating Ideas in Design

When asked to introspect on the source of their most creative or interesting ideas, almost all of the designers found it difficult to articulate where their ideas came from, and were able to make little distinction between the process of getting an idea and that of developing or refining it. To get as comprehensive a picture as possible, therefore, we collapsed introspections about generating and refining ideas.

Three general categories of activities accounted for over 60% of designers' comments; these were *logical analysis* (analyzing and classifying system function, studying data structures, top-down design languages, representational techniques such as diagramming), *discussion/consultation* (talking with colleagues, experts or users about design ideas, brainstorming), and what we have labeled *development activities* (prototyping or implementing function, making "to do" notes). Other activities included looking at other systems, literature reviews, usage scenarios, preliminary meetings with prospective users, relevant prior experience, intense concentration on the design problem, and periods of noninvolvement with the design problem.

Interestingly, although individual designers seemed to have a hard time describing their creative processes, the composite picture that emerged fits well with the traditional psychological literature on problem solving and creative thought (see [17] for an overview). Traditional work on creative problem solving emphasizes the importance of the initial representation of a problem; the logical analysis category reflects this type of activity. If a problem is not solvable in terms of the initial representation, then additional information must be sought to support a restructuring of this representation; designers' reported consultation with others, and their gathering of information generally can be viewed as examples of this restructuring. Another finding which appeared to map well onto the traditional account were comments describing initial intense concentration followed by a decision to put the problem aside; such a strategy has often been observed to lead to spontaneous problem solutions.

The "development activities" category was interesting in that it does not as clearly map onto general observations of problem-solving. It may, however, reflect a later stage in the process, a stage when ideas are represented more concretely for evaluation purposes. This notion is consistent with the finding that such techniques were more likely to be reported by designers with well-specified starting goals than by designers working with very general, open-ended design goals. There was a complementary distinction in the use of logical analysis strategies, with designers who were working on open-ended goals reporting more reliance on this category of techniques than designers with well-specified goals. Discussion and consultation with colleagues was important to both groups of designers, although one may speculate that the nature of the discussion depended on whether the designer was at the problem representation or idea evaluation stage.

It is difficult to know how best to support the creative process in design, a process that is admittedly hard to articulate and often mysterious (as one designer put it: "I just sort of scratch around."). There exist tools intended to facilitate logical analysis; some systems even hope to automate the process of generating code from some type of formal notation [16]. And certainly tools exist for prototyping; both types of prototyping described earlier would seem relevant to this process of generating and refining ideas. But designers' introspections also point to more informal methods of idea generation, often referring to discussions with colleagues, domain experts, or users. This suggests that one requirement for a design environment is a communication facility oriented toward explanation and rationalization of design ideas.

Some Final Comments

Our analysis of the interview data points to two types of prototyping tools. For incremental development, designers need a rich, modular, prototyping environment: because it is assumed that the prototype will grow into the final system, and because the design is likely to change in unknown ways, modularity of tools will be especially important. For phased development, continued iteration on a design may be impossible. One approach would be to ignore this model of development, under the assumption that it will always fail and eventually be replaced. But a more constructive solution would be to work within its constraints, to focus on providing a tool that would allow as much iteration as possible. Key will be the provision of prototyping tools for simulating function-interface combinations early in design, at the time when most flexibility exists. Good simulation tools would be useful within the incremental model as well, because they would support user testing of function prior to availability of a robust implementation.

Our analyses also point to a distinction between two classes of user interface design tools. For some design problems, the design space is wide enough that an initial decision about the most effective user interface will be impossible; for more well-specified problems, a designer may feel confident about (or business reasons may dictate) a particular style of interaction. These two situations call for different kinds of tools -- for the former, an integrated, relatively unconstrained environment, where interface and function can evolve together; for the latter, a structured tool that guides the designer toward the best implementation of the chosen interaction techniques.

Our conclusions at this point are of course only tentative; they are based on qualitative analyses of introspective reports. But they do suggest avenues of research in the tool domain. What are the characteristics of a quick but comprehensive tool for simulating function and interface for early testing? How much "simulated functionality" will be necessary for realistic usability testing? Will simulators promote usability engineering within both incremental and phased development models? Will the availability of specialized user interface tools inhibit designers' creativity in developing optimal function-interface combinations? These questions can be answered only by studying design tools in the context of actual design practice.

Acknowledgements

Thanks to David N. Smith, Tim Breen, and John Richards for discussions on this project as well as on the more general topic of tools for design.

References

1. Bennett, J.L. Managing to meet usability requirements: Establishing and meeting software development goals. In *Visual Display Terminals*, J. Bennett, D. Case, J. Sandelin and M. Smith, Eds., Prentice-Hall, Englewood Cliffs, NJ, 1984, pp. 161-184.
2. Boies, S.J., Gould, J.D., Levy, S., Richards, J.T., and Schoonard, J. The 1984 Olympic Message System -- A case study in system design. *Commun. ACM*, in press.
3. Budde, R. and Zullighoven, H. Internal Report, Gesellschaft fur Mathematik und Datenverarbeitung, GMD - F2G2, D 5205 St.Augustin, 1986.
4. Buxton, W., Lamb, M.R., Sherman, D., and Smith, K.C. Towards a comprehensive user interface management system. *Computer Graphics 17*, 3 (July 1983), 35-42.
5. Carroll, J.M. and Rosson, M.B. Usability specifications as a tool in iterative development. In *Advances in Human-Computer Interaction*, Vol 1, H. R. Hartson, Ed., Ablex, Norwood, NJ, 1985, pp. 1-28.
6. Carroll, J.M., Thomas, J.C. and Malhotra, A. A clinical-experimental analysis of design problem solving. *British Journal of Psychology 71*, (1979), 143-153.
7. Dehning, W., Essig, H. and Maass, S. *The adaptation of virtual man-computer interfaces to user requirements in dialogs*. Springer-Verlag, Heidelberg, Germany, 1981.
8. Goldberg, A. and Robson, D. *Smalltalk-80: The Language and its Implementation*. Addison-Wesley, Reading, Mass., 1983.

9. Good, M., Spine, T.M., Whiteside, J., and George, P. User-derived impact analysis as a tool for usability engineering. In *Human Factors in Computing Systems: CHI'86 Conference Proceedings*, (Boston, Mass., April). ACM, 1986, pp. 241-246.
10. Gould, J.D. and Lewis, C. Designing for usability: Key principles and what designers think. *Commun. ACM 28*, 3 (1985), 300-311.
11. Green, M. The University of Alberta User Interface Management System. *Computer Graphics 19*, 3 (1985), 205-213.
12. Hammond, N., Jorgensen, A., MacLean, A., Barnard, P., and Long, J. Design practice and interface usability: Evidence from interviews with designers. In *Human Factors in Computing Systems: CHI'83 Conference Proceedings* (Boston, Mass, Dec.), ACM, 1983, pp. 40-44.
13. Lammers, S. *Programmers at work.* Microsoft Press, Redmond, Washington, 1986.
14. Maass, S., Rosson, M.B. and Kellogg, W.A. User-friendliness, system consistency and other hard-to-define principles: Interviews with designers. To appear in *Proceedings of Software-Ergonomie '87*, (Berlin, Germany), Teubner, 1987.
15. Myers, B.A. and Buxton, W. Creating highly-interactive and graphical user interfaces. In *SIGGRAPH'86 Conference Proceedings* (Dallas, Texas, Aug.), ACM, 1986, pp. nn-nn.
16. Olsen, D.R. and Dempsey, E.R. (1983). Syngraph: A graphical user interface generator. *Computer Graphics 17*, 3 (July 1983), 43-50.
17. Posner, M.I. *Cognition: An introduction.* Scott, Foresman and Company, Glenview, Ill., 1973.
18. Shneiderman, B. A model programming environment. In *Advances in Human-Computer Interaction* Vol 1, H. R. Hartson, Ed., Ablex, Norwood, NJ, 1985, pp. 105-132.
19. Tanner, P.P. and Buxton, W. Some issues in future user interface management systems. In *User Interface Management Systems*, G. E. Pfaff, Ed., Springer-Verlag, Berlin, 1985, pp. 67-80.