Incorporating Behavioral Techniques into the Systems Development Life Cycle

By: Marilyn M. Mantei
Department of Computer Science
Faculty of Library and Information Science
University of Toronto
10 Kings College Road
Toronto, Ontario
Canada M5S 1A4

Toby J. Teorey
Center for Information Technology Integration and Dept. of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109-2122

Abstract
The gathering of a variety of human-oriented information is vital in the development stages of a software system. This information can be applied at a given stage to improve the human-computer interface of the software product. To reflect this, new categories of design and/or development effort need to be added to the traditional systems development stages. These efforts, termed user factor stages, differ from the traditional feasibility studies, requirements analyses, and tests that are currently conducted. The stages offer a flexible series of techniques, which can be compared and contrasted in terms of their expected information benefit, cost, and reliability of data obtained. As a concrete example, the development of a forms interface to a relational database management system illustrates these techniques.

Keywords: Design, human factors, management, human computer interaction, user factors life cycle, prototyping, user testing
ACM Categories: D.2.2, H.1.2, K.6.3

Introduction
A large variety of techniques exist for measuring human behavior in the field of psychology. The techniques are not general but depend upon the type of data to be captured and the context in which the data resides. A psychologist is deeply concerned that what is chosen to be measured is actually being measured and measured accurately.

If we apply psychology to user interface design, we run into two major difficulties. First, although a theoretical base exists for predicting human behavior, it is far from applicable to complex user interfaces. Second, even when it is possible to predict human behavior with a particular interface design, the cost required to generate the prediction is too high. Psychology concentrates on the minute detail of cognitive and social processes, not on the pragmatic aspect of getting an acceptable software product to market.

Because of these two reasons, generating user interface designs derived solely from principles of psychology is not practical. Yet user interface design is not an art. Engineering principles can be derived from the theory to suggest potential designs. Once a design or series of potential designs have been selected, psychology can be drawn upon for its measurement theory and techniques. These can be used to evaluate design prototypes. Evaluation studies indicate problems with the current designs; they do not predict how to fix them. Improvement is accomplished through an iterative process of test, fix, test, fix, etc.

Given these limitations for applying psychology to user interface design, how does the manager of a software project incorporate psychology into the software design process? Although much is known about the applicability of psychological theory in various contexts, this information has not been transferred to a suitable form for software project management; nor has the set of engineering techniques now available from the field of human-computer interaction been included in an overall development strategy.
This article is an effort directed at communicating such information across paradigms, from psychology to information systems. To aid in this communication, the traditional systems development life cycle (SDLC) is chosen. The SDLC is a management process used to assess a project's progress and control its cost. The life cycle is divided into distinct conceptual tasks with deliverables set at the end of each of the tasks. In presenting the addition of psychological techniques to the system development process, we split apart similar conceptual units.

At each stage of software development we describe the type of user information that is needed. Because the field of human-computer interaction is rapidly evolving, new techniques are constantly being created for capturing and applying user information. We call our modified life cycle the user factors life cycle. It serves as a framework for incorporating the new techniques.

In order to make our discussion of user interface techniques clearer, we use an example software development project, that of a form screen interface for a database management system.

The User Factors Life Cycle

We begin our presentation of methods for applying psychological principles and measurement techniques to the design of user interfaces by discussing the user factors life cycle. We assume that prototyping software is a part of the design process. Readers wishing to review the systems development life cycle are referred to Mills (1976) and Rubin (1970). The prototyping cycle is discussed in Boar (1984), Budde, et al. (1984), Harrison (1985), Wasserman (1982) and Wirth (1971).

Market analysis

The development stages for a user factors life cycle are shown in bold print in Figure 1. The market analysis stage precedes the traditional feasibility study stage. It determines what product to develop by examining people's perceptions and feelings about the computer tasks they currently perform.

The user information obtained at this level is difficult to capture and is untrustworthy. It documents people's attitudes and intentions and their work and play behaviors. This stage is almost always ignored in software development projects but can often be the primary determinant of the success or failure of the software (Ehrlich, 1987). The grain size of the data captured is large and only applicable to decisions such as what types of general features the product should have and where political problems might occur during installation. The information for this stage is often available from national surveys or meeting minutes.

Product acceptance analysis

Two user factors stages, product acceptance analysis and task analysis, are inserted between the requirements definition and system design (global design and prototype construction) stages. Product acceptance analysis involves developing some type of mockup of the proposed software product and presenting it to a sampling of the intended market. Future users are polled via focus group studies and/or user surveys to determine their reactions and acceptance or rejection of the software product. They are also asked to make recommendations for changes in the product.

The grain size of the data collected at the product acceptance analysis stage is as large as that of the market analysis stage. The control over the reliability and validity of the data is better because future users find it easier to specify their reactions to a concrete visual presentation than to verbal descriptions of the proposed product. Questionnaires now exist at this level that give a reliable assessment of user satisfaction with a scenario or product mockup (Root and Draper, 1983). Davis (1985) has shown that discretionary users give an accurate assessment of their

\[1\] We use the word grain size in a very limited context. It does not imply that we are able to make smaller and smaller refinements on the same data and still observe the same phenomena. We use it loosely to describe the amount of time a subject spends in generating a response to an experimental stimulus. In the case of questionnaire responses, it is the average amount of time used to answer a question, on the order of one minute. Contrast this time to a reaction time study in which subjects respond in a few hundred milliseconds. Thus, we say that the grain size of the first measure is much larger than the second measure. An analogous situation is the relationship between the behavior of a microprocessor chip and a C program on the same computer.
intention to adopt a software product from videotape simulations of the product.

Task analysis

Task analysis, which is performed by a user factors specialist, looks at how the task is viewed by a user and what mental manipulations are employed. The results of this analysis are applied to system design. The task analysis is the most expensive and most detailed data collection activity in the user factors life cycle. The data collected is at the finest grain size, often in the area of 250 millisecond units of user behavior. Because of the time it takes to conduct the task analysis phase, it is often omitted. However, this step involves the strongest application of psychological theory to user interface design. Several engineering models have been generated to help with the analysis of this stage, the most used being the GOMS model (Card, et al., 1980) and its derivative, the keystroke level model (Card, et al., 1980). None of the models designs the interface, but they do show where in the design users will perform more slowly or generate more errors.

Software has been built to stimulate user performance (Kieras and Polson, 1989; Runciman, 1986; Bennett, et al., 1987; Young, et al., 1989), but they require a painstaking description of the user task and the interface design to be programmed into the system. Performing a task analysis means breaking the performance of the task into its smallest measureable subunits. What these units are is somewhat arbitrary, and psychologists can easily build completely different task representatives out of the same data. Nevertheless, there exists some rigor in defining what constitutes the smallest unit of a task (Card, et al., 1983), but the conduction and interpretation of a task analysis still requires the expertise of a trained psychologist. Data from the task analysis often leads to entirely new design ideas (Card, et al., 1987) or a recogni-
tion that the original design plan for the user was misguided (Masson, et al., 1988).

User testing and evaluation

The *user testing and evaluation* stage combines two goals, one of determining if the interface is usable and a second of determining if it is acceptable. In user testing, projected users are studied while they perform a variety of tasks for which the system is intended. Problems they have in either learning the system or in extended use of the system are incorporated into design change recommendations, which are then built into new prototypes. Testing continues until the learning and extended use patterns of the system are at acceptable levels of both effort and ease of use. In the evaluation portion, once users have become comfortable with the system, they can then assess whether the system performs the desired functions and whether the effort needed to learn the system is in concordance with the features the system provides.

The work that takes place in the user testing stage is often referred to as usability studies. It is very effective for catching and removing detail problems in the design but has little impact on larger design decisions, such as choosing a command language over a menu-based system. Its data are of a finer grain size than the market analysis or product acceptance analysis stages but not as detailed as that collected in the task analysis. The time required to perform various subtasks, the number of errors encountered, or the number of requests for help are used as measures of the user quality of the interface.

The evaluation part of the stage serves two purposes, one of assessing the user’s attitude toward the interface design and a second of assessing if the system meets the user’s functionality requirements. This latter assessment has long been part of the traditional SDLC. The evaluation portion collects less-detailed data than the testing portion. It uses techniques similar to those in the product acceptance analysis stage. Like that stage, this information has little applicability to the details of design but serves as an early warning siren that something is seriously wrong with the design. Evaluations of user satisfaction and intentions to adopt are more reliable at this stage because the prototype provides the potential user with a concrete instantiation of the design. User interface testers often conceptually lump the evaluation and testing data of this stage together. Psychologically, they are very different pieces of information with the user satisfaction measures having little to say about the efficacy of the design.

User testing

After system implementation and product testing comes *user testing*. It is equivalent in goal and operation to the user testing in the user testing and evaluation stage. User testing is performed a second time because a working system differs from a prototype in response time, system complexity, etc. The results can serve as benchmarks for expected human performance levels and learning rates, data that can be used in marketing and user training.

Product survey

The *product survey* stage obtains user response on the released product. It involves techniques at varying levels of data aggregation from that of the user testing stage up to the market analysis stage. The data on user problems with detailed items in the design can be applied directly to the next revision of the interface, while the data on user suggestions for new functionality or observations of innovative applications of the existing system can be used to generate new products.

Each of the above stages fits meaningfully within the framework of the traditional lifecycle: the feasibility study containing a market analysis; the requirements definition including the product acceptance analysis and task analysis; system design containing prototype construction; user testing and evaluation; and the update and maintenance stage encompassing product survey. The following discussion illustrates typical techniques applied in these added stages.

The Example System: A User Interface to a DBMS

For illustration purposes, we have selected a prototypical software project, that of building a user interface to a relational database management
system. The software system is called Metaform (Beard, 1985; Beard, et al., 1987). The user studies indicate how its interface design was chosen and modified.

The goal of the Metaform project was to build a workable user interface to a relational database. The interface would need to support multi-relational, multi-entry database transactions. A human-computer interaction specialist and a project manager (the two authors) advised a systems designer who performed the design and implementation task. It was decided to build a prototype of the proposed interface on a PC and test out the prototype on individuals who could be expected to be typical users of the system. Before the design of the interface began, preliminary data were gathered on the user population to be used in working out the design.

We asked the systems designer to keep detailed notes of the time he spent on the project and the changes he made to the software as each stage in the design took place. Weekly meetings with the project manager and the human-computer interaction specialist provided input into the data collection and design process.

To give the reader an understanding of the size and difficulty of the project used as our example, we briefly describe the final user interface that was developed using the user factors life cycle.

Metaform automatically generates multi-relational form screen interfaces to the system's end users. The user does not need to know much about the database organization but only about the types of forms available. To construct a query, the user fills in various fields on the screen, specifying the data desired.

Figure 2a illustrates a form screen as seen by the user about to make the query, "display all the blue parts that are supplied by suppliers based in Detroit that are used by project Alpha." The user keys in "blue," "Detroit," and "Alpha" and receives the query result shown in Figure 2b.

Database systems are faced with the business problem of long training times and daily errors in database retrievals. Metaform, as a potential solution to this problem, evolved from the user factors development process. This begins with the market analysis.

**Conducting a Market Analysis on the Database Users**

The purpose of a market analysis is to determine what potential products users might buy. This is not a new technique but an old technique borrowed from marketing. The market analysis is different from a feasibility study because the problem is treated as a market problem, thereby changing the types of analyses that are employed. In particular, it brings in behavioral personnel to do an assessment of the customers' needs in the market. The three popularly employed techniques to obtain market data are:

1. Running focus groups
2. Conducting market surveys
3. Surveying customers using current products

The first two techniques have traditionally been used to evaluate the effectiveness of products that are already on the drawing board. They have been adapted to work as software product idea generators as well as product evaluators.

**Running focus groups**

One of the commonest ways to determine a customer's reactions to a product is to run a focus group. A focus group is a gathering of eight to 10 individuals run by a leader who, via his or her expertise in communicating, gets the gathering to talk about their feelings about a particular subject or item. Focus groups assess customers' reactions to such items as artificial bacon or motorized bikes for kids. We have adapted the focus group to generate product ideas. The subject area is introduced to the group, but no product ideas are presented. Instead, participants are encouraged to present their emotional reactions to the information processing tasks they perform manually or with the aid of existing computer systems. These sessions are videotaped and analyzed later for their potential suggestions for product solutions.

In the focus group run for the purpose of analyzing the problem of database retrievals, questions such as the following were asked:

1. What do you like most about performing database retrievals?
2. What do you like least about database retrievals?
3. Do you use pencil and paper to help you with your work?
4. Which database retrievals are very hard?
5. Why do you think they are so hard?
6. Once you have finished building a database query in the computer, what do you do next?
7. How do you decide that the database retrieval query you wrote is okay?
8. What would you prefer to do to obtain the same information you now obtain by writing your database queries?

Questions 1-2 are asked to assess what features of the work are enjoyed and thus are to be preserved and what features are to be eliminated. Question 3 assesses what types of paper and pencil work are still being carried out that could possibly be replaced by the computer software. Questions 4-5 find out where a new computer product could perhaps ease the pain of the database retrieval and also where it might make work more interesting. In questions 6-8, the potential customer is asked what types of additional thinking goes on in the task that would not be apparent by just looking at the retrieval requests. The forms/database management interface came out of question 8 where a participant in a focus group remarked that he preferred filling in office forms to generating a query.

Conducting market surveys
Market surveys are carried out to measure existing customer behavior that is determined to
be related to a product being proposed. They involve the distribution of a questionnaire or, more often, a questionnaire-driven telephone interview to a random selection of the potential user population. They are too structured for the idea generation phase of system development and cost three to ten times more than a focus group to run. The data collected can be trusted to be more representative of the user population but may not provide the information required at this user factors stage.

To run a market survey for assessing future user interface development potential in the database area, a non-biased but viable population sample is needed. The task of database retrieval is sufficiently complex that questions about a retrieval interface are confusing to interviewees who have never performed such a task. If, however, the survey was conducted on individuals performing database retrieval, their answers would be biased by their understanding of and emotional reaction to the system they were using. Because of the intricacies of database retrieval, the second population is the appropriate one to use, but with the caveat of inherent bias. Choosing a population using several different forms of database interfaces is appropriate in this case. Similar questions to those asked in the focus group can be asked, but they must be rewritten to generate a context for the question. Otherwise the individuals filling out the questionnaire will fill in their own context to answer the questions, making a summarization of the results invalid and useless. Examples of questions that could be asked in the market survey are as follows:

1. Many of the database retrievals I am asked to perform are too hard to do.  yes ( ) no ( )
2. I am afraid of making a mistake. yes ( ) no ( )
3. Others view me as doing difficult and demanding work. yes ( ) no ( )

The above questions assess people's emotional response to using their database retrieval system and collect data on why they like or dislike using the system.

Surveying customers using current products
Many new product innovations come from customer suggestions. This technique formalizes the task of getting customer suggestions by conducting a survey requesting their suggestions. Because feedback from customers is so important an avenue for new product ideas, it has been incorporated as a final stage in the user factors life cycle. Additional details on the customer survey are given in the discussion of the product survey stage.

A comparison of the recommended techniques
The cheapest and best technique to use for this unstructured stage of system development is the focus group. Its main problem is a lack of generalizability and a high potential for bias in interpreting the results. If the potential user population and its needs are well understood, this step can be easily bypassed. Surveying customers using current products is the second best technique, with market surveys being a poor third. The reliability of the information received is poorest for the focus group method, while the appropriateness of the information obtained is best from focus groups and worse from market surveys.

The Product Acceptance Analysis
The product acceptance analysis takes the proposed solution back to its future users for their reaction. It is distinct from that of showing users the written requirements document in that it demonstrates actual system usage and does not require the user to imagine the system.

Facading
Two techniques are used for conducting this analysis, facading and running a focus group. The first of these is a technique developed by Davis (1985) and Davis and Olson (1986). A mock demonstration of the system in operation was videotaped and shown to potential users, who were asked to assess their probability of using each system feature. Davis (1985) has shown the user's responses to reflect later adop-
tion of the feature. Using and analyzing the set of questions developed by Davis is straightforward and can be carried out by software personnel.

A variation on the facading technique was used on Metaform. Instead of videotapes of the proposed forms interface, paper-and-pencil tests were used to simulate the database retrieval. The main concern over the effectiveness of the proposed interface design was whether forms helped resolve users' problems in formulating queries. Although forms seemed, conceptually, to be a simpler idea, users' problems with database retrieval could be based on their inability to understand relationships between elements of the database and not on their inability to learn the query language. It was felt that the expense of building a videotape facade could be avoided and that the paper facade would answer the effectiveness question.

Individuals without computer experience were trained to fill in forms to reflect retrieval requests they might be asking of a messenger. The retrieval correctness rate was 95 percent. Students participating in the study were significantly faster than secretaries, but did not have significantly more correct answers. This paper-and-pencil pretest indicated that the use of forms might be a correct user interface metaphor, but it did not test out the acceptance of the interface. The videotape facade coupled with a user questionnaire or focus groups can help ferret out this information.

Running a focus group with a product in mind

The second technique is another round of focus groups. Instead of finding out how the potential users of the system feel about their current work, the group focuses on the new product and the user population's receptiveness of the product. For example, the following questions could be asked of the group's participants:

1. How much time do you think it would take you to learn how to use the forms system?
2. How long do you think it should take someone to learn the database retrieval task that you now do?
3. How would you rate the forms interface in comparison to the one you use now? Why?

The first two questions probe users to find out whether they think the new system is worth learning. The final one tries to get an emotional response to the new system by having its future users compare it to current working conditions.

By gathering adoption data at an early stage of the product development, projects that are in danger of non-adoption or of causing high job dissatisfaction can be modified or even cancelled to avoid these disasters.

A comparison of the recommended techniques

Facading using videotapes and the Davis acceptance questionnaire is an expensive operation but gives trustworthy data. Using storyboards or paper-and-pencil simulations has the disadvantage of misrepresenting the potential product to the user population. Videotaping should definitely be used if considerable expense will be involved in developing the product. Videotapes can be used to present the software design to focus groups, but, as with the focus group analysis in the market analysis stage, the results are inherently subject to small sample bias and group leader misinterpretation.

A Task Analysis of the Retrieval Process

A task analysis is a breakdown of a task into its individual independent components (both mental and physical) (Card, et al., 1983). In this stage of the user factors life cycle, a task analysis is performed on existing tasks. This is done in order to get an understanding of how the future user of the new system thinks about the work that will be supported. The task analysis can then be applied to keep the design from violating the user's approach to the task. Two methods are used to obtain data for a task analysis. The first of these is called verbal protocol analysis and involves having individuals describe what they are doing as they perform a designated task. The second method is memory organization analysis and encompasses a variety of cognitive psychology techniques for obtaining the memory organization an individual has built of the task under analysis.
**Verbal protocol analysis**

In a verbal protocol analysis, an individual is asked to speak out loud, saying what they are thinking or planning as they perform a particular task. A skilled psychologist can break the pauses and the verbal contents into well-defined objects and processes that reflect how the person giving the protocol thinks about the task. An example of such a protocol for database retrieval is:

"Now, uh, <pause> I can get all the, uh, red parts by setting the condition to red, and, uh <pause> I, uh, need the parts record because, uh, <pause> that is, uh, where the color and price are located."

The verbalization is in response to a needed query of the form, "It is Valentine’s day; find all the products we sell that are red." In the protocol, the person doing the retrieval mentally selects the retrieval condition and then finds a mapping between the condition name and the database structure. Once the mapping is found, the condition is set. The structured breakdown of this task analysis looks as follows:

**GOAL:** Retrieve all products that are red

**SUBGOAL:** Set retrieval condition to red

**SUBGOAL:** Find record containing color field

In this scenario, the user of the system needed to know the location of the field for color. Knowing the location of a data element in a large database is a difficult task. In the design of the user interface for the new system, the protocols indicated that providing external memory with this information might be useful.

Verbal protocol taking can be done with a tape recorder unless the task requires capturing an individual's behavior with a keyboard or paper and pencil. It is best done with problem-solving tasks in which the individual is conscious of the mental processes taking place. A number of constraints determines when verbal protocols are valid. They are described in Ericsson and Simon (1984). Verbal protocols are rich sources of information about the task being computerized, but their transcription and analysis is extremely time-consuming.

**Memory organization analysis**

When humans perform tasks a large number of times, they build an internal memory organization for the elements of the task but are rarely able to verbalize the existence of these task components that are retrieved and executed automatically. An example of such a process is shifting or braking in an automobile, which occurs without conscious effort.

Cognitive psychology has developed a variety of methods to obtain these mental organizations. One of these is the copy task where individuals are given five seconds to look at a given stimulus (for example, a set of database retrieval instructions) and then asked to copy the stimulus to a blank sheet of paper. The copy exercise is repeated until all information is transcribed. The amount of information copied in one transfer is believed to represent the component of information the individual stores in memory. These components represent the subsets of the task a user would execute when building a retrieval. The components can also be obtained by measuring the length of pauses that occur when an individual is asked to recall memorized text associated with the task being studied (e.g., a list of database commands). Longer pauses are believed to mark the end point of groups of words stored together in memory.

**A comparison of the two techniques**

Both techniques take considerable time. The verbal protocol technique is cheaper in time and money and more flexible and thus applicable to most user interface design tasks. Memory organization analysis is a very accurate but time-consuming approach. It does not work well on ill-structured tasks or for tasks in which the individual under study has had little experience.

An extensive task analysis with either method takes time. When completed, it represents the goals and subgoals users follow and the steps they take to achieve their goals. It can be used to both test and drive the system design. Task analyses are only used in system design by researchers. Good examples of task analyses are found in Card, et al. (1983), Kieras and Polson (1985), Runciman and Hammond (1986), Simon (1988), Walker and Olson (1988), and Young and MacLean (1988). Because this stage is so time-consuming and demands special user expertise, it is best in very large and innovative
types of system development projects. We did not use a task analysis for the database project. We have use it to examine how financial planners use modeling packages. Its use has pinpointed places were users are likely to make conceptual and syntactical errors in different types of modeling problems (Lerch, et al., 1988).

It is not expected that the proposed system will copy the task under analysis. The goal of the task analysis is to understand the internal structure the user has built of the task. This is not simply dependent on the current technique in use but also on the way the user may have learned to solve the problem in school, on the way the problem is usually written in English, and on the way the problem is communicated from one person to the next in the work setting. If a new system is built that significantly modifies the way the user has to think about the task, the user will have long-term difficulties in using the system, especially if the user is required to maintain other mental organizations of the task in order to communicate with co-workers.

User Testing and Evaluation

The goal of the user testing portion of this stage is to try out the user interface design embodied in the prototype. Users are observed learning and using the prototype. Difficulties they incur in learning the system and mistakes they make are recorded by the observer. The prototype is then examined in light of these difficulties and modified to remove the causes. User testing is then redone until an acceptable learning speed and reduced level of difficulty with the system are reached. The basic techniques used in this stage are:

1. Experimental testing of design decisions
2. Usability studies
3. User exams

Experimental testing of design decisions

In the process of building the system software, various detailed design decisions occur. Many of these make no difference whatsoever to the final product. Others are extremely important. A rough rule of thumb for ascertaining the importance of the design decision is how often the feature will be used. If usage is high, then selecting the appropriate design is very important. With a prototyping tool, both designs can be built and tested in a series of experiments.

Such a design choice occurred in our example project with cursor movement. In Design 1, the users could move the cursor wherever they wished on the screen. In order to fill in fields in the form, they had to position the cursor in a given field and begin typing. In Design 2, use of the return key advanced the cursor from field to field, automatically positioning it at the start of the field. If the users wished to move to a portion of text in the field, e.g., to correct a typographic error, they could press the left and right cursor keys. They were permitted to move only within the field they were currently positioned at. Design 1 was significantly easier to implement. It also allowed larger fields by permitting the user to type on several lines. Design 2 kept the user from mispositioning the cursor. To correct the positioning problem for Design 1, the field was designed to show a highlighted reverse video background when the cursor was placed inside it.

To run a controlled experiment on the two design choices, we would have trained potential users of the system to enter data into each of the fields on the retrieval forms displayed by the prototype we constructed. The users would have been matched on skills such as typing and previous computer experience and randomly assigned to two groups. One group would have used Design 1 to enter a set of data into the forms, while the other would have used Design 2. Time and error comparisons would have been used to determine if any significant differences in performance occurred with the two designs. We did not run this experiment. Instead, we built Design 1 and timed and observed users performing a data retrieval task with the system. Users had so much difficulty positioning the cursor correctly with Design 1 that we discarded it in favor of Design 2.

In designing the forms interface we were using a series of metaphors about manual tasks that occurred in offices. Filling in a form with a typewriter was one of those metaphors; and others such as card files, file folders, desk piles, and file cabinets were also used. The typewriter seemed to match Design 1. However, as noted above, the eye-hand coordination demands of
Design 1 led to such disappointing results that we modified our system to take advantage of instantaneous positioning provided by Design 2. We also made the change because we reasoned that all use of the retrieval system would be semi-casual; that is, users were not likely to perform retrievals as the major portion of their work day. Without continued practice, users would not learn the motor skills demanded of Design 1 and would continue to struggle with cursor placement.

Usability studies

In addition to resolving design decisions, the prototype can undergo an open-ended evaluation called usability testing (Bennett, et al., 1987; Carroll and Kellogg, 1989; Manheimer, et al., 1989; Whiteside, et al., 1987). Both the design decision tests and the usability testing are done in a simulated work environment. If the final product is to be released in an office, this environment is an office. If assembly line workers are to use the product, then the environment is a factory floor.

Users are given everyday tasks to perform using the prototype. A camera records a user’s difficulties and successes on videotape. In most cases the individual in the study vocalizes the difficulties encountered as they occur. This person may also describe how the system is believed to operate and may offer various strategies for getting the system to perform a desired task. Secondary mechanisms, such as keystroke records (Good, 1985) and recorded comments by an unseen observer, are also used to obtain data.

The information captured in a user study is analyzed for problems encountered with the system. Among these are consistent errors that occur in a particular place in the interface, stated misconceptions the user has about using the software, high task-performance times, and numerous requests for help made by users at key points in the task. The user interface specialist and the design staff discuss each of the observed difficulties and agree on design changes that might alleviate the problems.

In the Metaform usability study, users of varying degrees of computer experience were brought into an office-like setting. They were trained in using the system by the experimenter and an experiment manual that walked them through a series of tasks. At the end of each series of tasks, users took two exams. The exams required them to perform queries on the database system they had just learned. Questions on the exam were of three types: answerable with the current set of instructions, answerable with earlier instruction sets, and not answerable. Users had to perform the desired retrieval and write the retrieval answer in an exam book. The first exam was a practice exam in which the users received feedback on their answers.

Instead of videotaping users, the experimenter took notes on the users’ problems. The prototype interface recorded the time to perform each of the retrievals, and the exams gave information on the difficulties the users had with various retrievals. We used this information to give us critical incidences, i.e., those points in the interface where the user constantly has trouble with the system. The usability study pointed out several unexpected flaws in the interface design. In one case, the response to pressing the “return key” was so immediate that users had difficulty positioning the cursor at the correct field in the form. This was changed by putting a delay in the key response. The organization of the menu and the naming of the database fields also caused the users difficulties.

User exams

A technique not as involved as controlled experiments or usability studies is that of giving tests to the user of the prototype system (Reisner, 1981). The tests are designed to examine what the user has learned about the system, where the user’s misconceptions occur, and what tasks the user does not know how to accomplish with the system. The results of these tests are used to ferret out the difficulties the user is having in learning and using the interface. A portion of the questions on the test are meta-questions about the task. They ask the user to describe problems that he or she had in using the system.

The tests can take three forms. Users can be administered a written test, an online test, or a verbalization test. The first type of test is a set of written questions that users respond to in writing. It takes place after users have had training in using the system and have been using it for a specified period of time. The online test is like
Behavioral Techniques in Systems Development

the written test except that the user answers the questions on the written portion of the test by executing sequences of commands on the prototype system. A log of the user responses is kept and analyzed for errors in the same way the written test would be analyzed. In the verbalization test, the user is asked to tell the person administering the test how they would perform a task they have been asked to do (Kato, 1986; Knox, et al., 1989). The user is often asked to describe to the examiner what is being done as the user executes the task on the prototype system. The written tests are easiest to generate and administer but are the most remote from measuring true system usage. The verbalization tests often reveal large misunderstandings that users form of system operations, but they are time-consuming and difficult to analyze.

User examinations were administered as part of the usability study in evaluating the form screen interface. We used both written and online tests to evaluate users’ ability to build retrieval queries. The tests were used in addition to the usability studies because we wanted to separate out aspects of the task that were related to the user understanding the context of database retrievals. We also needed to determine where, in the process of the user mapping the desired query on to the Metaform system, user difficulties occurred. They could occur in the interpretation of the English statement of the request, in the mapping of the request on to the retrieval representation, i.e., the form screen, or in the interpretation of how the form screen system worked. To obtain this information, we built a test that required users to perform a multiple choice selection for three different types of task.

1. English-to-graphics query.
2. English-to-Metaform query.

In the English-to-graphics query, users had to pick from a set of graphic processes representing paper file retrievals, the process that matched the stated English query. In the English-to-Metaform query, users needed to select (from a set of form screen processes) the process that would obtain the desired query. In the Metaform-to-English query, users selected the English statement that corresponded to the process being illustrated. An example of a Metaform-to-English query is shown in Figure 3.

The analyses of the tests demonstrated that some of the problems users were having in building retrievals in the prototype were associated with the English wordings of the questions and with the users’ understanding of the relationships between the data items. The tests also demonstrated that the multi-stage interface worked when users did not have to use the complement operation. It was felt that the difficulty the users had with complements was one of not having a metaphor to use for its understanding. This metaphor was created in the instruction given to users in subsequent testing but did not remove the problems. We subsequently increased the practice users had on this type of operation. This led to improved performance.

A comparison of the three techniques

Usability studies are the most common technique used. They are inexpensive and the most flexible form of gathering a large amount of user data. It takes time to develop training materials and a set of user tasks, but the training materials can then constitute part of the finished software product. Usability studies only point out problem areas; they do not tell the designer how to correct the problem. Experiments give direct feedback on a design decision but do not evaluate the entire interface design. They are often included in the usability studies. An experiment requires the building of two prototype interfaces and the duplication of studies on two user populations. They therefore take up time and money but do not provide the breadth of usability studies.

User exams are inexpensive but narrow in focus. In our example, they allowed us to separate out other aspects of the task that might be causing our users trouble. The exams did not uncover problems with design details. They are best at obtaining a user’s understanding of the underlying concepts needed for using the new interface. An experiment can obtain any level of information on the interface, from conceptual understanding to ergonomic performance. It depends on what the experiment is designed to measure. Usability studies gather a broad overview of the user’s understanding of the system but also capture information on the effect of interface details.
What you should do:
a. Circle the sentence or question which best matches the following retrieval instruction.
b. If you don’t know which is correct, circle “I don’t know.”
c. If you think none is correct, circle “None match.”

PLANE FORM

<table>
<thead>
<tr>
<th>Serial #</th>
<th>Type</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Piper</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Airport</th>
<th>Airport City</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New York</td>
<td></td>
</tr>
</tbody>
</table>

CHOOSE ONE OF THE FOLLOWING:
1. Retrieve from the File Cabinet
2. Add to the Desk Pile
3. Retrieve from the Desk Pile
4. Retrieve everything not in Desk Pile
5. Quit (clear Desk Pile)
6. New Form (start retrieval over)

YOUR INSTRUCTION TO COMPUTER: 1

1. Which city has a Piper airplane plant?
2. Which planes land in New York?
3. List the weight of Piper supplied airplanes.
4. Which planes are supplied by Piper and land at an airport in New York?
5. None match.
6. I don’t know.

Figure 3. Example of Metaform-to-English Query Question Used in a Set of Tests Given to Users of Metaform Prototype

Two basic problems with user testing exist. First, what is being tested is how easy it is for the user to learn the interface rather than how easy the interface is to use. The short time span given to users to work with the system is the root cause of this problem. Second, the prototype, as built with existing prototyping systems, does not represent the final system. Differences can be large in terms of response time, sizes of databases being used, and the demands on the user’s time.

Evaluation
The evaluation portion of this stage uses the same techniques as the product acceptance analysis stage. The results of the evaluation at this stage can be trusted significantly more than evaluations at any prior stage. Users now possess a tangible object in the prototype that they can relate to the tasks they wish to perform with the system. Performing tasks with the system also generates new ideas for ways in which the tasks can be performed easier.
User Testing — Performing the Bench Mark Tests

In the second user testing stage, usability studies and user exams are run on the actual system rather than a prototype. At this stage, it is useful to design the user tests in such a way that they capture information that can be used to predict an individual's performance with the system. Learning time, representative performance time, and retention are important data to capture.

The Product Survey — Did They Like it?

This stage is like the market analysis stage of the life cycle. It gathers information about the use of the product in the environment it was designed for (Lewis, 1984; Nielsen, et al., 1986). This information may lead to a redesign of the current user interface or to the development of a completely new software product. Even if the information obtained is not used for product development or enhancement, the retrospective analysis of the user interface that was designed can serve as a valuable source of learning for the systems development staff.

This stage is the only time that information can be gathered about the long-term use of the product. The prior prototype and bench mark testing cannot be carried out long enough to capture this type of data. Several techniques can be used for the product survey stage. They are:

1. The Customer Hot Line
2. On-site Observation
3. User Survey

The customer hot line

One of the ways in which the system development department can obtain a large amount of data on the product's user interface is by setting up a hot line or help service. All questions asked with the hot line service are recorded. These questions provide significant clues to the problems users are having with the system.

No hot line service was established for Metaform. However, for a different project one of the authors was involved with, a microcomputer software company did have a telephone number that purchasers of the previous version of its software package could call if they were having trouble with the software. When that software project began, a log was kept of each call. The calls showed several unanticipated difficulties. Users were having trouble telling their software package what type of printer they owned. The printer selection interface was designed based on this information. Users also had difficulty with following all the steps in a procedure described in the user manual. This corroborated the decision to build in an online tutorial that demonstrated the procedure to the user in a window next to the user's workspace. Finally, the words that the users chose to describe their problems were selected as keywords to use in the development of the user manual for the new system and in naming menus on the screen display.

On-site user observation

On-site user observation is similar to a usability study but takes place where the software system is being used. Users of the system are videotaped performing the daily tasks they perform with the system. If videotaping is not possible, the user works with an observer, telling the observer what is taking place in the system and what task is being attempted. The observer takes notes or makes an audio transcription of the session. Critical incidents that were not discovered in the pre-release testing may show up at this point.

Although the Metaform system was never released to users, users of another database interface were videotaped performing updates and retrievals. The database was the alumni database at the University of Michigan. The personnel performing the updates were extremely skilled at their task and performed the updates very quickly. A slowed-down replay of the videotape demonstrated that field cursor placement was a large problem that occurred in almost all updates. The up/down cursors on the terminal were used to move from field to field in the data entry task. The keys were so sensitive to the amount of time they were depressed that the cursor continually bypassed the target. The alumni office personnel would pass the target as much as five times before finally landing on the desired field. This occurred for each field on the screen. The task of data entry had become an arcade game requiring the development of skilled eye-hand coordination.
**User survey**

Once a software product has been released, information about both the effectiveness of the software and about how to modify the software can be obtained through the use of questionnaires. These questionnaires can be administered early in the distribution of the product to determine how easy the system is to learn or at later stages to measure continuing user problems.

The distribution of a questionnaire for a software product developed for internal use inside a company is relatively simple. Obtaining this same information for a commercial product is more difficult. Several strategies are used in the latter case. These include mailings in response to a customer hotline, the use of a designated group of users who have agreed to provide feedback for an incentive, or the mailing of questionnaires to a customer registration list.

In order for the information from the survey to be of use, it has to be at the level of detail that illustrates individual commands and system responses. Example questions that probe at this level of detail are:

1. Which of the following retrieved questions do you think are impossible with this system?
2. What information do you write down on a piece of paper when using the software system?

These questions both probe how the user understands how to perform the tasks the system is designed for and how the system is actually used. They also look for ideas from the user of the system on how to change the design.

**A comparison of the three techniques**

If a customer hot line is planned for the released software system, recording the calls or keeping a logbook are inexpensive additions for gathering problems with the user interface. Analyzing the information that is captured takes time because the data is very unstructured and not necessarily related to the software interface but to the logistics of the person owning the software, e.g., "When are you going to come out with version 4.0?"

The hot line data is often at such a low level of resolution that it does not provide much insight into users' problems. For example, a user called the microcomputer software company to ask why she couldn't use her printer. We could not extract the information on what she had done to specify the type of printer she owned. Because there is so much data and because useful information is at a low density, the hot line calls have to be aggregated before trends in problems with the interface are seen.

Alternatively, on-site observation gathers large amounts of detailed data with a few short visits. The constraints on this type of data capture are its intrusion and the impossibility, in some cases, of obtaining access to a site. If on-site observation is possible, it is recommended over the other two methods.

In contrast to both on-site observation and the customer hot line, the user survey allows more control over the type of data being gathered. In an on-site observation, it may not be possible to determine a user's understanding of the system, or certain features in the system may never be used during the visit because of job demands that use the system in an uneven fashion. The user survey cannot get at the detailed, second-by-second interface problems, and answers to survey questions mailed to users are often suspect, i.e., they may not have been generated by the chosen respondent.

A user survey is expensive to administer and relies on the good will of the user population. This means that the most wanted feedback, that of disgruntled users, is the least likely obtained. Due to better user compliance, the use of surveys for evaluating the user problems of in-house software projects has much more success. The cost of distributing these surveys is also lower. Analyzing user surveys is considerably easier than analyzing videotapes of on-site usage or hot line logs.

This stage could be dropped if you could be assured that getting a product out the door is the end of it. However, developers have painfully found that no matter how good a job they do, they can't do it perfectly for all customers. Only after your product has gone through sustained, heavy use do you have all the survey (live test) information you need.
About the Authors

Marilyn M. Mantei is associate professor of computer science and library and information science at the University of Toronto. She received her Ph.D. in communications theory from the Annenberg School of Communications at the University of Southern California in 1982. She has previously taught at Carnegie Mellon University and the University of Michigan, where she built a doctoral program in human-computer interaction and started the Human Computer Interaction Laboratory. She has worked in industry for the Xerox Palo Alto Research Center (1975-77). Her current research interests include computer-supported cooperative work, cognitive models of user interfaces to complex systems, and the management of user interface design.

Toby J. Teorey is associate professor of electrical engineering and computer science at the University of Michigan and is a senior staff member at its Center for Information Technology Integration (CITI). He received his Ph.D. in computer science in 1972 from the University of Wisconsin, where he was coordinator of the Operations Research Group at the Madison Academic Computing Center. He has served as an EDP officer in the U.S. Air Force (1965-69) and was a social aide for President Lyndon Johnson from 1966 to 1968. He is the co-author with James P. Fry of Design of Database Structures, published by Prentice-Hall. His current research interests include semantic data models, distributed databases, and the design and analysis of large, interconnected local area networks.