

Levels of Automation and User Participation in Usability Testing

Kent L. Norman

Department of Psychology

Laboratory for Automation Psychology and Decision Processes

University of Maryland, College Park, USA

Emanuele Panizzi

Dipartimento di Informatica

Universita di Roma "La Sapienza"

Rome, Italy

ABSTRACT

This paper identifies a number of factors involved in current practices of usability testing and presents profiles for three prototype methods: think-aloud, subjective ratings, and history files. We then identify ideal levels to generate the profile for new methods. These methods involve either a human observer or a self-administration of the test by the user. We propose methods of automating the evaluation form by dynamically adding items and modifying the form and the tasks in the process of the usability test. For self-administration of testing, we propose similar ideas of dynamically automating the forms and the tasks. Furthermore, we propose methods of eliciting the user's goals and focus of attention. Finally, we propose that user testing methods and interfaces should be subjected to usability testing.

Keywords: Usability testing, user interface, WWW

INTRODUCTION

In the last decade, usability testing has become accepted as an important part of software design, testing, and acceptance. Many different methods and procedures for usability testing have been developed that vary in important and consequential ways from one another. These approaches have been compared and contrasted in a number of ways, principally in their ability to detect and rectify user interface problems. However, there are many other ways that these methods differ that affect the cost and applicability of user testing of an application, particularly those on the World Wide Web (WWW). This paper first considers the ways in which usability testing methods and procedures vary along a set of relevant factors. We do this using a morphological analysis, which generates profiles of current approaches for detailed

comparison. Following inspection of these profiles, we propose several new methods of usability testing that provide a better profile of the factors. These methods seek to increase the level of automation of user testing, decrease labor on the part of testing personnel, modulate the level of involvement of the user in the data collection, and further automate the data collection and data analysis components of usability testing.

Usability Testing Methods and Factors

Usability testing is now accepted as an essential activity in the lifecycle of software design, implementation, testing, acceptance, and revision (Dix, Finlay, Adowd & Beale, 2003; Shneiderman & Plaisant, 2003). A number of usability methods have been developed and promoted by different researchers (Neilson & Mark, 1994).

Usability testing has a number of possible goals and purposes. Certainly one of the most important is to discover major problems in the user interface that could result in human error, terminate the interaction, and lead to frustration on the part of the user. Other goals might be to reduce training time, increase performance and efficiency, and increase user satisfaction (Shneiderman & Plaisant, 2003).

Some usability testing can be accomplished through the use of checklists, guidelines, and principles. For Web-based applications, these can also be semi-automated. But since the goals of usability testing involve human users, most usability testing methods naturally involve tests on real users and require human observers to evaluate the outcomes of the test. Consequently, usability testing tends to be rather labor intensive. While the user must by necessity be doing some task with the interface, usability testing often involves a task administrator giving

instructions and multiple observers logging codes for the interactions, taking notes, and interpreting the interaction. The user may be minimally involved in the evaluation and only perform the tasks as directed or the user may be highly involved in the process of evaluation such as being asked to think-aloud, make subjective ratings, and even provide suggestions for improvements to the interface as a design partner.

To meet the need for usability testing, many companies and government agencies have set up usability laboratories. These labs are equipped with hardware and software that automatically capture the interactions. But automation of data capture and analysis has generally been a problem owing to the unique situation of each application. Moreover, access to usability labs is limited and it is often more practical for observers to set up testing areas at locations convenient for the users without the use of special equipment such as video cameras and eye trackers. Several studies have attempted to compare usability-testing methods in terms of their ability to identify types of usability

problems (John & Marks, 1997) and their influence on software designers.

It is clear that competing methods of usability testing differ along a number of dimensions. In the sections that follow, we discuss a number of these factors. Then we use these dimensions to construct the morphological box (Ritchey, 1998; Zwicky, 1969; Zwicky & Wilson, 1967), which is shown in Figure 1. To aid in the discussion, we explore the profiles of three dominant and exemplary usability approaches: (a) the think-aloud method (TA) in which users are instructed to provide a running verbal commentary on what they are thinking and doing, (b) the method of automatically recording of user interactions in a history file (HF), and (c) the method of assessing subjective ratings (SR) of the interaction by the users. The reader is invited to graph the profile of other methods that we do not present here as each factor is discussed. Finally, we list what we might call the “ideal method”(IM) that capitalizes on the best values and mixes of factors in the box. After discussing each factor and the three prototypical methods, we will return to the “ideal method.”

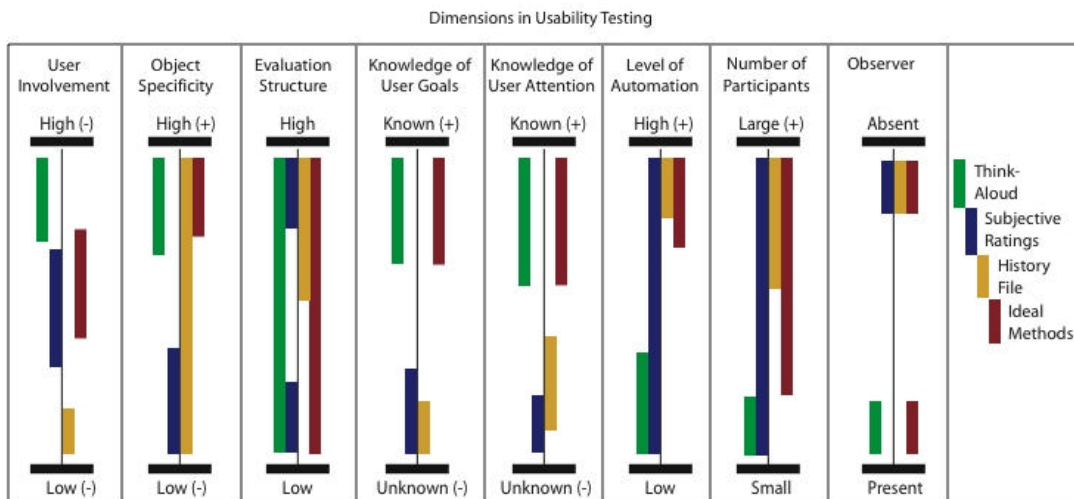


Figure 1. Morphological box of eight dimensions of usability methods showing profiles of three prototypical methods and one ideal method. (Pluses and minuses indicate desirability of end points.)

User involvement

The first factor of usability testing that we will consider is the degree to which the user is mentally involved either introspectively or in terms of self-assessment. At one extreme, the user may be highly involved. Think-aloud procedures require the user to constantly verbalize what he or she is thinking, expecting, deciding, etc as shown by high value in Figure 1

for user involvement (Lewis, 1982). The verbal protocols that are elicited in this method can be very rich in diagnostic and evaluative information, but they can also be contrived, biased, and misleading. Verbalizing while doing can be very demanding and can change the course of one’s behavior (Rhenius & Deffner, 1990). To avoid the problem of trying to do two things at once, the method of retrospective report

can be employed in which the user first performs the task without verbalization and then reviews a recording of the interaction and comments on the events as they are replayed. While the user may recall the reasons for particular actions and reactions to system responses, it is possible that imperfect memory can distort recall and result in false rationalizations and constructed explanations. But the biggest problem with the retrospective account is that it takes substantially longer than the think-aloud method (Norman & Murphy, 2004).

At the other extreme, the user may be minimally involved in the evaluation. Only the interaction between the user and the system is observed in a history file of recorded events and the user is neither asked to think-aloud nor to give opinions or ratings. This is the least invasive method as shown by the profile in Figure 1. Only the user responses to system events are captured. The advantages are that the user is performing the task in the most natural way. The disadvantage is that the responses do not always capture what is happening. For example, the user may have different goals and may at one time be engaged in causal browsing and at another time rushing to complete a transaction before a deadline. Nor do the responses convey any information about expectations, frustrations, or satisfaction.

Users are often asked to provide subject assessments using questionnaires and rating scales. Usually, these ratings are done at the end of formal user testing sessions, but sometimes after each task is performed. Users are asked to provide ratings of satisfaction, confusion, frustration, and particular interface attributes (e.g., system responsiveness, meaningfulness of messages) either during the interaction or in a debriefing session after the task is complete. Figure 1 shows that methods of subjective assessments fall at an intermediate range for user involvement.

Degree of Interface Object Specificity

The second factor that we consider is the specificity or granularity of the evaluation. User testing may focus at one extreme on the overall usability or user satisfaction with an application or at the other extreme on very specific interface objects such as the location or icon of a particular button on a particular screen. Global assessments are important for evaluating one application relative to others and for gaining a sense of the user's overall satisfaction with the

system. User rating scales such as the Questionnaire for User Interaction Satisfaction (QUIS) (Chin, Diehl, & Norman, 1988) and the (SUMMI) (Kirakowski & Corbett, 1990) are useful instruments for this purpose. However, if an evaluation is very low, such instruments might be able to point to a type of problem (e.g., confusing terminology), but they do not identify the specific interface objects that are causing the problem. Figure 1 shows these methods at the low end of the scale of object specificity.

To achieve a more specific identification of problems, user testing typically involves a screen-by-screen and object-by-object evaluation. Cognitive walk-throughs by experts are used to identify specific interface problems. Observations of users performing tasks and instructions to "think-aloud" help to identify specific problems. But as noted earlier, these approaches are very labor intensive and costly. Consequently, user testing is often limited to just a handful of participants and only a dozen or so hours of observation. While high frequency problems are likely to be observed, less frequent, peculiar problems may escape observation and detection. Figure 1 shows these methods at the high end of object specificity.

History files of user interaction can be programmed at different levels of specificity. They may record only high level transitions from screen to screen and times to complete tasks or screens; or they may record every mouse click and movement and every keystroke. Figure 1 shows that these methods can span a range of levels of object specificity.

Degree of Structure

Evaluations can be either open-ended with little structure or highly structured with rating scales, multiple-choice questions, and checklists. Open-ended questions are necessary when the evaluators do not know what to expect from the user or when they want to discover new categories of problems and issues. Open-ended questions result in qualitative data that need to be interpreted and categorized by the evaluators. This can require a lot of time and effort. On the other hand, structured evaluations generally result in quantitative data that can be quickly summarized and analyzed with statistical programs and easily interpreted by the evaluators. Most usability testing uses both structured and unstructured questions to benefit from the advantages of both.

Think-aloud methods tend to be very open-ended; and hence in Figure 1, they are shown at the low end. It is thought by some advocates of the think-aloud method that adding structured questions might lead the user or suggest answers that they would not otherwise have made. Subjective rating scales, on the other hand, are much more structured. History files by their nature must be pre-programmed. However, if one captures everything, there really is very little structure in terms of what actions or problems are being investigated. Additional programming is required to add structure and to capture more meaningful events, such as failure to efficiently navigate through a site rather than to use the back button. Nevertheless, Figure 1 shows that the history file tends to be more structured.

Knowledge of User's Goals and Intentions

Behavior is generally goal driven. Even browsing Web sites in a seemingly random manner is affected by the user's goals, aspirations, and intentions. Consequently, it is extremely important in usability testing to know the user's goals at any moment. In many user testing situations, the goal is set by the task administrator who tells the user what task to perform and why. For example, the user may be asked to find the cost and availability of some product or to fill out a form as fast and accurately as he or she can; or if the user is free to choose his own task, he will communicate it to the administrator.

In many testing situations, the task administrator may let the user go to explore and work on their own. In field-testing, users may come to the application with different goals in mind (e.g., make a purchase, compare prices, etc). The only way to know the user's goal is to ask. In most think-aloud tests, a task administrator gives a specific task to be performed. Consequently, Figure 1 shows that for think-aloud methods the level of goal knowledge is high.

On the other hand, when using only a history file in user testing, there is little way of knowing what the user's goals are unless they are somehow inferred after the fact. For example, after the user has entered credit card information and completed a transaction, we might infer that the user intended to buy the item. Nevertheless, for history files, Figure 1 shows a low level of goal knowledge. Finally, with subjective ratings, there is usually little indication of what the user's goals are.

Tracking the Focus of Attention

A substantial part of the user's interaction with a system is looking at or listening to the interface without generating any feedback to the system. It is important to know where the focus of attention is for the user because it involves the locus of information acquisition or the point of decision-making. Knowledge of the focus of attention is acquired in the think-aloud procedure when the users verbalize what they are reading or looking at. But these data are not automated and not easily logged. So, while Figure 1 shows the think-aloud method high in knowledge of focus of attention, it is also low in automation. The history file in its current implementation gives an indication of focus of attention only when the user clicks a button or link or types in a field. Finally, the subjective ratings give virtually no information about the focus of attention.

A popular option is to use eye tracking as a proxy for the user's focus of attention. A close coupling exists between gaze and attention although it is not perfect. The user may stare at one part of the screen while thinking about something else. But the real problem with eye tracking is that the equipment and software is expensive and can require substantial time to set up and expertise on the part of a usability technician. New systems do not require cumbersome and distracting hardware for the user, but they are still expensive and require some calibration; and of course, they are not an option for self-administered user testing on the WWW. The question is whether the user can provide input to the system for focus of attention. As will be seen in the sections to follow, we will propose a method of adding this information to the history file.

Automation

It is odd that most usability testing is done in a very manual, labor-intensive way when computers and automation surround the usability laboratory. Why can't we automate user testing? Programs such as Noldus Observer (Noldus, et al, 1999) help the observer to code behavioral events, record times, and associate events with video capture. This and other programs also have facilities to sort, organize, and analyze the data, but still require more manual labor than seems appropriate. Why aren't the observations recorded automatically? The reasons become obvious when one looks at the process. The software being tested is often an application that

cannot be easily modified to capture and record the user interactions. Consequently, one has to resort to recording the interaction using video cameras, scan converters, or screen capture programs. Some of these approaches capture the user's input, but they do not capture the user's focus of attention or the user's goals and intentions. These require other labor-intensive methods.

The most ambitious attempts at automation of usability testing come from the artificial intelligence (AI) community. The idea is to totally automate user testing by running surrogate users in the form of artificial intelligent user agents that model the perceptual and cognitive processes of real users. These approaches are extremely compelling and interesting but doomed to fail for a variety of reasons. The first is that we really have not been able to generate truly autonomous user agents that act like humans. Past and current attempts have been extremely costly, overly simplistic, and totally specific to the task and application (Norman, 1991).

More modest approaches to automation attempt to record, analyze, and interpret interface events of real users performing either simulated tasks in the laboratory or real tasks in an environment. Software that records a history file is being developed more and more for this purpose (Cugini, 2000), but again it has the problem of leaving out the user's goals and intentions and much of the user's focus of attention when not actually clicking on a button or typing in a field. Subjective ratings of usability have appeared in both paper and pencil form (low automation) and in on-line form (high automation). Analysis of the data has also spanned the range from low to high levels of automation.

Number of Participants

Usability methods by their nature have been associated with either very small samples or large samples. Since think-aloud methods are so labor intensive, they have leaned toward very small samples and researcher have tried to justify the use of small samples (Lewis, 1994; Virzi, 1992). Subjective assessments either in paper-and-pencil mode or online have leaned toward large samples especially when statistical methods are used. History files, which record interface events, can be used with both small and large samples, but since they are highly automated and involve substantial effort to set up, are usually

used with moderate to large samples as shown in Figure 1.

Presence of Observer

Finally, although we have referred to this factor repeatedly in preceding sections, we must consider the factor of whether a human observer is required. The question of whether an observer is needed depends on the extent to which (a) the user in conjunction with the usability system can self-administer the test and (b) the user and/or the system can record diagnostic events on their own.

The think-aloud method, of course, requires a human observer whether onsite or in a teleconferencing mode. The human observer is generally also the task administrator who instructs the user as to what tasks to perform. The subjective rating method is generally self-administered and the history file is totally automated so neither of these methods requires a human observer as shown in Figure 1.

Other Related Factors

One can think of a number of additional factors that vary among usability testing methods. For example, one important dimension is whether one does the testing in a controlled laboratory setting as opposed to a field study in the natural environment. There are advantages and disadvantages associated with each end of this factor. The usability laboratory has testing equipment and space for observation. Laboratory testing reduces variability and allows for controlled experimental designs, but it is often artificial and changes the context of testing for the user. Laboratory testing also requires the users to travel to the testing facilities often making it inconvenient for the user and more costly for the client. There are some solutions, such as portable usability labs, that try to maximize the benefits of both the laboratory and the field. Portable equipment can be taken to the user in the home, office, factory, or public space.

Another factor is the presence of video cameras, recording devices, and other control equipment. But this factor is associated with the laboratory, whether fixed or portable. The factor of the availability of subjects, whether coming to the lab or the lab going to them, is associated with field-testing. Think-aloud methods are more likely to be associated with laboratory testing. Subjective ratings, whether on paper or online, are generally independent of location. History

files are more likely in controlled laboratory settings where the software can be modified to record interactions. However, for Web-based applications, this is not an issue since the software can be modified on the Web server.

Of course, cost is an important factor in usability methods, but cost is an integral function of many other factors and how those factors are implemented. One can also imagine a factor on which the user testing is either done in a quick and dirty manner or in painstaking and thorough manner. Finally, there are performance factors that vary among method including the diagnosticity of the test for identifying problems and the statistical reliability of the test for hypothesis testing. Since these factors are really outcomes of the test and the result of how other factors are implemented, they do not need to be considered on their own.

NEW MIXES OF USABILITY METHODS

The question then is how to generate a user testing/evaluation method that achieves an optimal mix of automation and human effort, structure and lack of structure, specificity, and level of user involvement. In this section, we will outline a method that attempts to do just this with the assumptions (a) that one can attach or plug in software that records user and system interactions, (b) that the user is sufficiently motivated to make a number of judgments and evaluations during the session, (c) that the user can inform the system of his or her goals and intentions even as they change, (d) that the user can convey to the system his or her focus of attention, and (e) that sufficient structure can be added to the evaluation to allow for automation of the analysis.

The last bar on each scale in Figure 1 shows what we consider to be the ideal level or levels for the factor. For user involvement, we believe that the user should be involved to an extent, but neither too taxed by the assessment nor totally ignored. A moderate level of involvement engages the user as a participant and helps to provide information about goals and focus of attention. While assessments of overall satisfaction are important and useful for motivating change, they do not indicate what should be changed. Ideally, we would like to acquire more specific information about the interface and problems with particular elements. Thus, new methods need assessments linked to specific screen elements (e.g., illegibility of a

part of text on a particular screen) and actions rather than overall assessments.

We believe that evaluation structure should include both pre-planned structure (specific questions and tasks) that can be modified and open-ended parts that capture events that were not foreseen by the evaluators. But the system must be dynamic to handle changes in the structure over time, dropping out or modifying questions and tasks throughout testing and adding new items as they appear in the open-ended comments of the users.

Knowledge of user's goals is important. The knowledge of the user's goal at any one time may be determined either by instructing the user to perform certain tasks with specific goals or by asking users what they are trying to do. Similarly, knowledge of the user's focus of attention is important. The question is whether expensive hardware is needed or whether we can gather this information in a different way.

Obviously, we would like to increase the level of automation in a usability system to reduce the load on human observers, increase the efficiency of the test, manage dynamic aspects of the test, record the data, analyze the data, and report the results. To the extent that more can be automated, we can increase the number of participants without incurring large increases in the cost of the evaluation. Although one can justify small samples in usability testing, it will always be the case that one can learn more from larger samples particularly when the user population is diverse and one would like a sufficient representation of each type or category of user in the population.

New methods can in many cases eliminate the need for human observers and in other cases they cannot. Therefore, we must consider both cases in the implementation of new methods. The question is how to make user testing more productive and efficient when an observer is definitely required and how to replace or make up for this information when an observer is not present. We will propose solutions to these problems in the next section.

IMPLEMENTATION

In this section, we present the potential implementation of the new methods suggested. We will first consider the new methods that involve a human observer and then methods that

involve a self- or user-administered test. The user-administered test will employ some of the same ideas for automation that help to manage the observer administered test. Finally, we recommend the use of tracking software to record history files but as well as methods to associate these events to observations recorded by human observers and/or users.

Methods Involving a Human Observer/Administrator

Human observers are often needed to record events in user testing and to write comments and evaluations as the tasks are performed. In many cases, there is no way around the need for a human observer who is able to detect complex and often unanticipated interactions and to classify them into categories of events. Particularly for time critical tasks where users need to respond quickly and cannot act as their own observers, human observers are a necessity. No automated method can be obtained at this point that could replace the human observer. However, the task of observation, recording and commenting on events and classifying observations could be aided by automation.

Observation Records

Currently, a variety of software packages have been written to help support the observation task (e.g., Noldus et al, 1999). In general, these applications, facilitate recording by automatically time stamping the records and by providing preprogrammed “hotkeys” or menus for behavioral events. Few, if any of these programs are Web-based and all require considerable effort to configure and set up before testing and considerable time to post-process the results. Even with careful pre-programming, one often misses important and frequent events that should have been planned. The result is that the observer must repeatedly add written comments for the same event and later have to count the frequency of these events.

Our approach is to strike a balance between pre-programming and a dynamic modification of classification of behavioral events. We start with a small set of obvious events that the observer might code $\mathbf{B} = \{\text{event}_1, \text{event}_2, \dots, \text{event}_n\}$ on a Web-based form. During a user testing session, the observer can either select from among this set or add either new events or write new

comments. Following each user testing session, the new events are automatically moved to set \mathbf{B} ; and if desired, the observer can select written comments and move them to set \mathbf{B} . As testing progresses, efficiency will increase since the observer will have fewer written comments to make. However, as set \mathbf{B} gets larger, it will be harder and harder to find and select the appropriate behavioral code.

At this point, it will be necessary to dynamically organize and cluster set \mathbf{B} to facilitate a selection of codes. First, the set could be organized by frequency of past occurrence. This would allow the observer fast access to frequent codes. Second, it could be organized by temporal position in the testing session, if that position tends to be consistent from one session to another. This would allow the observer to position the list temporally during the session. Finally, it may be observed that some codes are either behaviorally equivalent and can be combined or that they are subsets of other codes and can be hierarchically clustered and organized. One may have codes for general behaviors (e.g., confusion) and sub-codes to identify specific aspects of the behavior (e.g., confusion about navigation, confusion about input format). If time permits, immediate post-processing of the codes will take advantage of better memory for the behavioral events on the part of the observer and avoid memory loss and retrieval bias later on. Figure 2, shows a schematic diagram of the process of adding, clustering, and organizing events over a series of user testing sessions. Session 1 starts with only a few events selectable. During or after Session 1, the observer can convert comments to selectable events and can combine and cluster events. The system may automatically convert new events that are of a general nature (labeled “new behavioral event” in Figure 2) into selectable events for the next session. Comments of a particular nature (labeled “new comment” in Figure 2) would only be recorded for future processing. After all of the users have been tested, the process results in a hierarchy of events with associated frequencies and other information. The result is that most of the post-processing of the events and comments has already been done in the course of testing.

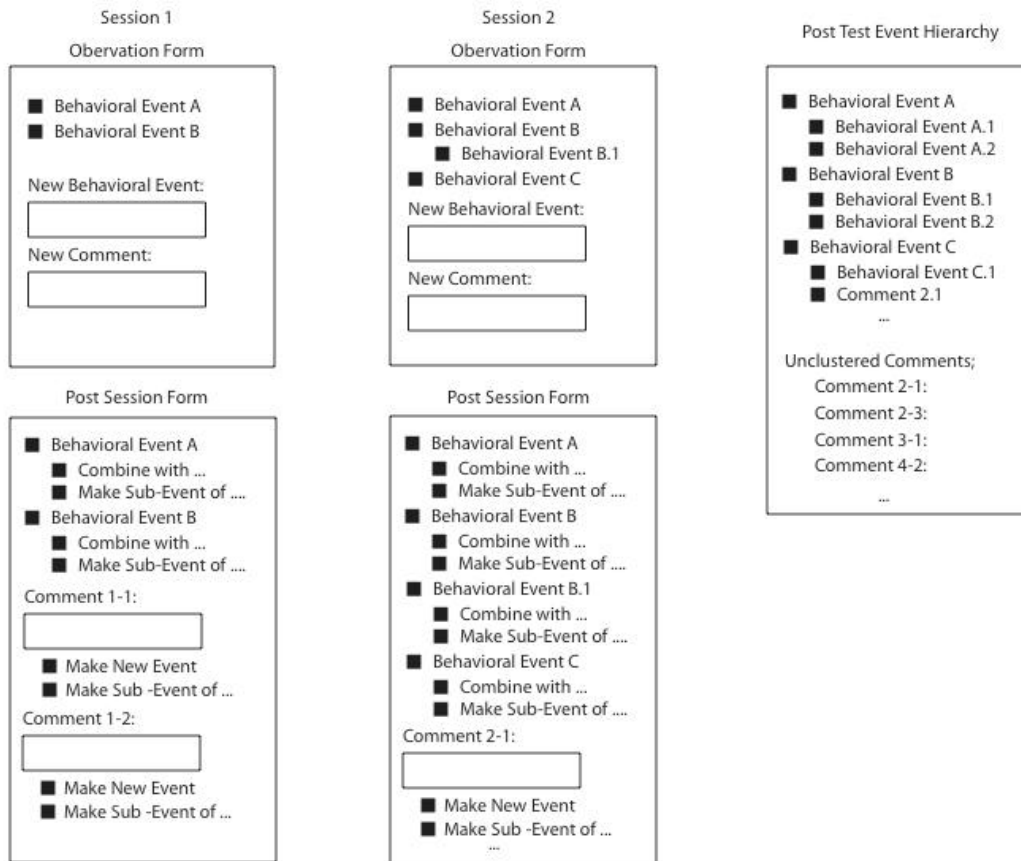


Figure 2. Diagram of the observer logging form from one session to the next with post-session processing.

Initially, events are organized by order of entry into the form; however, when a sufficient number of events have been observed, it will be possible to organize them by frequency and by temporal position in the testing session. Events may be given a temporal order in two ways. First, some events may be associated to tasks that are performed in a sequential order. Second, other events may be found to occur consistently at relative points in the session (e.g., near the beginning, in the middle, near the end). To order events by temporal position, the time of occurrence of events is recorded and its relative position between the start and end of the session is calculated. If over a number of sessions, this temporal position is relatively consistent, the events can be placed in a reasonable temporal order. Figure 3 shows the enhanced observer form that will facilitate the selection of events with sorting by frequency and temporal position.

Adaptive Tasks

Most usability testing involves the selection of a set of target tasks for the users to perform during

the testing session. These tasks are often selected to be representative of typical tasks that users will perform in the field. In addition, some tasks may be selected for testing because they are critical to the health or mission of the system; whereas, other tasks are highly destructive or unusually error prone. Task administrators try to select tasks that will prove diagnostic in revealing user interface problems, but it is often hard to second guess and to predict what the best tasks are. Some tasks can be omitted and the need to test other tasks becomes apparent only after the testing has commenced. Unfortunately, the convention generally adopted from experimental methods has been to run all of the users through the same set of tasks and to not vary any conditions. However, in the spirit of Bayesian and adaptive testing, there is no reason to continue to run tasks that are not diagnostic or that have already proven their point. Moreover, once tasks have been dropped new tasks can be added to replace the old.

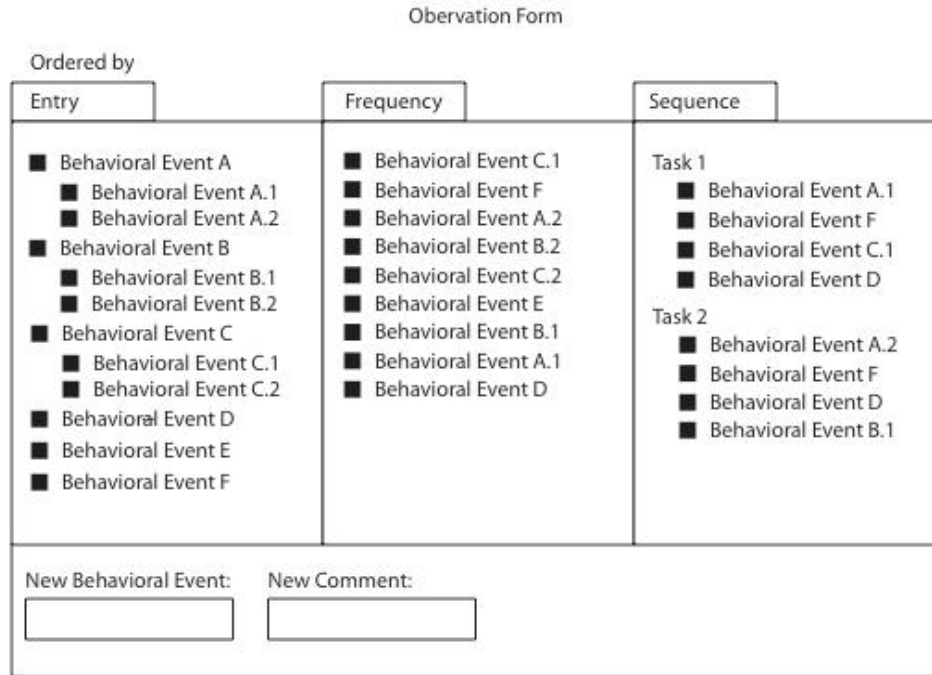


Figure 3. Diagram of the observer logging form showing three different methods of sorting the events.

In our approach, we consider a set of user tasks $T = \{task_1, task_2, \dots, task_k\}$. This set can be altered depending on the results. Figure 4 shows a diagram of tasks and sessions. We start with four tasks. Some tasks drop out because they are not diagnostic and do not reveal any problems. When tasks fail to report any negative events over a series of tests, they should be dropped. The question is how to set the criterion for the number of sessions before a task is dropped. We suggest using the Negative Binomial distribution, which gives the probability of no occurrences for X sessions. Thus, for example, if we set the probability of the occurrence of an event on any one session at $p = .20$, and no occurrences over sessions $P(X = N) = .10$, we would drop the task after $N = 8$ no event sessions.

Other tasks may be dropped out because they have consistently revealed the same problem. In this case, one needs to confirm that the event occurs for a sizable proportion of the user population (e.g., $p > .20$). This could be established by using a one-sided confidence interval on the proportion of occurrences. When

the lower limit of the interval exceeds the criterion (.20), then the task can be dropped.

Other tasks may be introduced as the task administrators discover new issues and problems with the system. These tasks are submitted to a pool in the database with estimations of the time to complete each task and priority values for inclusion. As testing time opens up, the system would insert new tasks from this list. Figure 4 shows a diagram of the task database starting with the initial set of tasks and showing tasks being dropped and added over sessions.

Methods Involving the User

In this section, we discuss new methods that involve input from the user. First, we will present methods for the user to serve as the observer and to report events and subjective assessments. Second, we will present methods to complete the information missing in observer records and history files about goals and focus of attention.

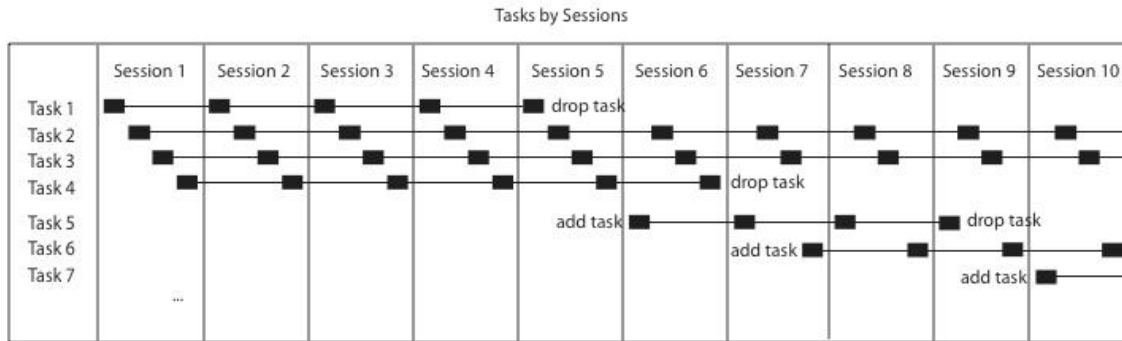


Figure 4. Diagram of the tasks presented to users over sessions.

Self Reporting of Events and Subjective Assessments

In the previous section, we presented methods of providing dynamic forms for the observer; in this section, we drop the observer and ask the users themselves to fill out the forms. The only change needed in the software and the database of events is to phrase the questions in terms of the first person and to add subjective ratings for the user to complete. In addition, for most tests, users can suggest new events and write comments, but they will not be automatically added to the form. All post-processing and changes to the form will be done by usability administrators. The only exception might be in cases where the user community itself is essentially in charge of the usability test.

In general, for Web-based interfaces, a second window or a second frame will be used to present the questions and ratings. The elements of the evaluation screen will be tied to the elements of the screen being evaluated and the database will store these associations. The primary limitation of this approach is screen real estate. If multiple windows are employed, the user must flip between windows that will probably occlude one another. If frames are used, they may not be big enough to display all of the elements without scrolling. Large screen systems or two monitor set-ups would be ideal, but are not prevalent in the user community. Consequently, users will have to make an extra effort to manage this problem.

Dynamic and Self Reported User Tasks

When testing requires the user to perform a series of tasks, these can be generated from the task database discussed in the preceding section and presented in a task window or frame. An example is shown in Figure 5. The user may be required to indicate the starting and ending time of the task, whether it was successfully

completed, and answer a question regarding the task.

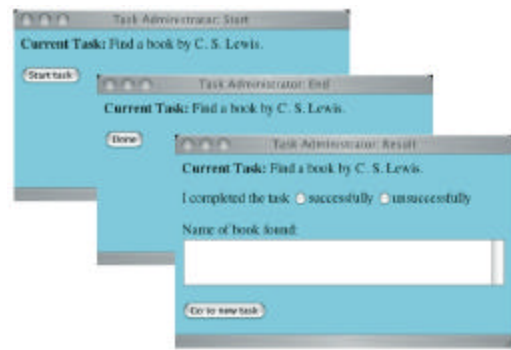


Figure 5. Illustration of a sequence of windows for task administration.

Alternatively, in cases where users are working on their own tasks, we can ask them to specify what they are doing at any point and to indicate when they change to a different task. Again, a new window or frame would allow users to input this task or goal information. Figure 6 gives an illustration of such a window.

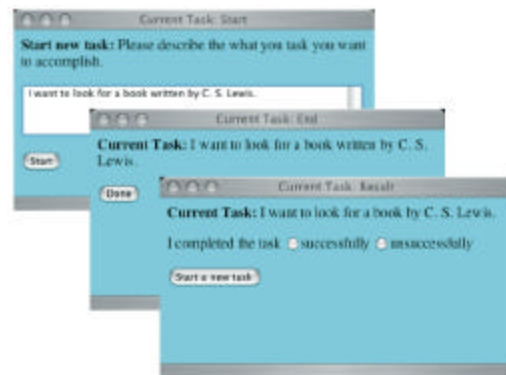


Figure 6. Illustration of a sequence of windows requesting the user to specify the current task.

Mouse Tracking as a Proxy for Eye Tracking

The user's focus of attention is a theoretical construct that is not directly observable. In usability testing, it has been associated with gaze and assessed using eye-tracking devices. However, there are other proxies for the user's focus of attention. The most obvious and easy to capture is where the user is pointing on the screen. Consequently, we propose to use mouse tracking to follow the focus of attention of the users. Mouse tracking is easy to record and will serve as a proxy for the user's focus of attention rather than eye tracking. But, it should be noted that while eye tracking may capture rapid preconscious attention processes, mouse tracking would only capture attention that follows consciousness. The question is whether the user can be instructed to continually move the mouse to the location of gaze and hence, focus of attention. Figure 7 shows a set of instructions to the user.

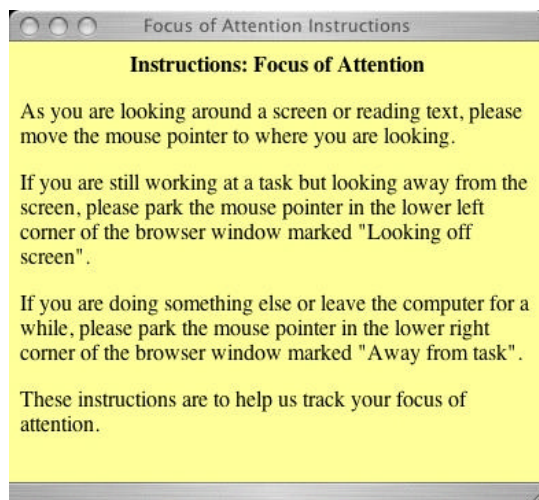


Figure 7. Focus of attention instructions for the user.

Mouse tracking can be recorded using screen capture or it can be directly stored in the history file as x-y coordinates. The problem to be solved in the later case, is the association of the x-y coordinates to objects and text on the screen particularly in resizable browser windows.

Application

We are currently developing the software and database to implement the new methods described in the preceding section. This system will run on a Web server and will be assessable anywhere; it is based on an automatic form generator (Panizzi).

User and observer forms

We use a form generator to define both user forms and observer forms. Our generator is a server-side application capable of storing an unlimited number of form descriptions, displaying these forms on web pages, and collecting input data in a database. The form description is given through a web-based user interface. We use this interface for setting up the initial observer forms and the user's subjective-rating and dynamic task forms. We are currently adding a new feature to the form generator in order to store alternate phrasing of the labels of the form fields. This will allow us to use the same form description to collect data both from the observer when present and from the user when the test is self-administered.

Dynamic observer form

We are currently working on the development of a software module capable of automatically modifying the description of the observer form by interacting directly with the form generator. In fact, when the observer submits the form at the end of each test, our module will read the declaration of new behavioral events (see Figure 2, column 1) and will add the corresponding new fields to the form description. The database tables will be altered accordingly. Thus, in the next session, the form generator will produce a different, enhanced form (see Figure 2, column 2).

Adaptive tasks

Adaptive tasks will be managed in a similar way by modifying the user form description as well as the database tables. In this case, however, there is no automatic updating in response to user. The test administrator will use the form generator interface to drop old tasks and to add new tasks. The generator will then update the form description and modify the database tables appropriately.

Empirical Verification

The new methods proposed here seem like good ideas to improve the usability and effectiveness of usability testing. However, in the spirit of usability testing, one really needs to run empirical tests on the proposed methods and interfaces. Consequently, we propose a number of experiments to verify the effectiveness and usability of these methods as listed below:

- Automation of observer's form:
 - Will observers find the form easy to use?
 - Will the automatic addition of events to the form increase efficiency and effectiveness of the usability test
 - Will observers find post-session processing of the events and comments easy and useful?
 - Will the alternative methods of sorting the events help the observers to find events more quickly?
- Effectiveness of dropping and adding tasks
 - Will usability testing be more efficient?
 - Will testing be able to examine more tasks and detect more problems with the interface?
- Self-Administration of usability testing
 - Will users find the evaluation form easy to use?
 - Will users be too distracted by having to fill out the forms?
- Knowledge of user's goals
 - Will users find the screens for task specification easy to use?
 - Will users be able to identify their current tasks and goals?
- Mouse tracking
 - Will users be able to reliably follow the instructions regarding their focus of attention?
 - Is the mouse tracking reliably correlated with eye tracking data?
- Level of user involvement
 - Will users be able to manage the additional windows required for the self-administration of the evaluation?
 - How much extra time will it take to do self-administration of the testing?

CONCLUSION

As usability testing becomes more and more a part of the lifecycle of software, particularly for web-based applications, it will be more and more important that testing methods are efficient, effective, inexpensive, and easy to use. Consequently, continual development of new methods is important to move from expensive, labor intensive methods to inexpensive, automated methods while maintaining a high level of diagnosticity. The new methods described here are attempts to do just that. Finally, it is clear that usability testing methods like many other applications need to transition to the WWW. The software that we propose does

this to take advantage of server-side generation of rating forms, task administration, and data collection.

ACKNOWLEDGEMENTS

We thank the U.S. Census, Statistical Research Division for support during this project and particularly the Dipartimento di Informatica, Universita di Roma "La Sapienza", Rome, Italy for hosting the first author during his sabbatical leave Spring 2004.

REFERENCES

- Chin, J. P., Diehl, V. A. & Norman, K. L. (1988). Development of an instrument for measuring user satisfaction of the human-computer interface. *Proceedings of CHI '88*, 213-218.
- Cugini, J. (2000). Web usability logging: Tools and formats. Presented to the *Tools to Support Faster and Better Usability Engineering* workshop, August 15, 2000, Asheville, NC. (<http://www.itl.nist.gov/iaui/vvrg/cugini/webmet/paper-aug2000.html>)
- Dix, A., Finlay, J. E., Abowd, G. D., & Beale, R. (2003). *Human-computer interaction (3rd Ed)*. New York: Prentice Hall.
- John, B. E., & Marks, S. J. (1997). Tracking the effectiveness of usability evaluation methods. *Behaviour and Information Technology*. 16, 4, 188-202.
- Kirakowski, J, and Corbett, M. (1990). *Effective methodology for the study of HCI*, North-Holland, Amsterdam.
- Lewis, C. (1982). *Using the 'thinking-aloud' method in cognitive interface design*, (Research Report RC9265), IBM T. J. Watson Research Center.
- Lewis, J. R. (1994). Sample sizes for usability studies: Additional considerations. *Human Factors*, 36, 368 - 378.
- Nielsen, J. & Mack, R. L. (eds.), (1994). *Usability Inspection Methods*, New York: John Wiley.
- Noldus, L., Kwint, A., ten Hove, W., Derix, R. (1999). Software tools for collection and analysis of observational data. In: *Human-Computer Interaction: Communication, Cooperation and Application Design*. Proc. 8th Int. Conf. on Human-Computer Interaction, Vol. 2 (Eds. Bullinger, H.J.; Ziegler, J.), pp. 1114-1118.
- Norman, K. L. (1991). Models of mind and machine: Information flow and control between humans and computers. In M. C.

- Yovits (Ed.) *Advances in Computers*, New York: Academic Press, pp. 201-254.
- Norman, K. L., & Murphy, E. B. (2004). Usability testing of an Internet form for the 2004 Overseas Enumeration Test: A comparison of think-aloud and retrospective reports. *Proceedings of the Human Factors Society 48th Annual Meeting*. New Orleans, LA: Human Factors Society.
- Panizzi, E. An object-oriented system for automatic web-form generation and processing. In preparation.
- Rhenius, D., and Deffner, G. (1990). Evaluation of concurrent thinking aloud using eye-tracking data. *Proceedings of the Human Factors Society 34th Annual Meeting* (pp. 1265-1269). Santa Monica, CA: Human Factors Society.
- Ritchey, T. (1998). General morphological analysis: A general method for non-quantified modeling. *16th EURO Conference on Operational Analysis*, Brussels.
- Shneiderman, B. & Plaisant, C. (2003). *Designing the user interface: Strategies for effective human-computer interaction (4th Ed)*. Reading, MA: Addison-Wesley.
- Virzi, R. A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors*, 34, 457- 468.
- Zwicky, F. (1969). *Discovery, Invention, Research - Through the Morphological Approach*, Toronto: The Macmillian Company.
- Zwicky, F. & Wilson A. (eds.), (1967). *New Methods of Thought and Procedure: Contributions to the Symposium on Methodologies*. Berlin: Springer.