# APPLYING COGNITIVE WALKTHROUGHS TO MORE COMPLEX USER INTERFACES: EXPERIENCES, ISSUES, AND RECOMMENDATIONS

*Cathleen Wharton*

Hewlett-Packard Laboratories
and
Dept. of Computer Science and
Institute of Cognitive Science
University of Colorado
Boulder, Colorado 80309-0430
cwharton@cs.colorado.edu

*Janice Bradford*
*Robin Jeffries*

Hewlett-Packard Laboratories
P. O. Box 10490
Palo Alto, CA 94303-0867
bradford@hplabs.hpl.hp.com
jeffries@hplabs.hpl.hp.com

*Marita Franzke*

U S WEST Advanced Technologies
and
Dept. of Psychology and
Institute of Cognitive Science
University of Colorado
Boulder, Colorado 80309-0345
mfranzke@clipr.colorado.edu

**ABSTRACT**

The Cognitive Walkthrough methodology was developed in an effort to bring cognitive theory closer to practice; to enhance the design and evaluation of user interfaces in industrial settings. For the first time, small teams of professional developers have used this method to critique three complex software systems. In this paper we report evidence about how the methodology worked for these evaluations. We focus on five core issues: (1) task selection, coverage, and evaluation, (2) the process of doing a Cognitive Walkthrough, (3) requisite knowledge for the evaluators, (4) group walkthroughs, and (5) the interpretation of results. Our findings show that many variables can affect the success of the technique; we believe that if the Cognitive Walkthrough is ultimately to be successful in industrial settings, the method must be refined and augmented in a variety of ways.

**KEYWORDS:** Cognitive Walkthrough, group walkthroughs, task-based evaluations, usability inspection method, user interface evaluation.

## INTRODUCTION

The need for practical techniques for critiquing and iterating a user interface design early and often in the development process is well recognized. The ideal technique would be usable early in the development cycle and inexpensive in monetary cost, time, and the need for access to scarce expertise. Several evaluation techniques are available that attempt to meet various of those goals, e.g., usability testing, heuristic evaluation, guidelines and style guides, GOMS analyses, and Cognitive Walkthroughs [3, 5, 6, 8, 11]. This paper reports on experiences using the Cognitive Walkthrough method in real development environments.



The Cognitive Walkthrough is a model-specific methodology originally designed for the evaluation of simple *Walk Up and Use* interfaces. Consequently, most applications of the method have been based on interfaces of that type (see [7, 8, 13]). Recently, however, there have been attempts to understand how the method scales up to interfaces that are themselves more complex, but still support infrequent or novice users [2, 5, 7]. Additionally, it has been a goal [of 2 and 5] to conduct these evaluations in more realistic contexts by having them carried out in industrial environments by groups of software developers, rather than by HCI specialists.

In this paper we discuss both issues and recommendations for this methodology in light of our experiences with three complex user interfaces. We describe recurring issues we observed and their implications for the use of the technique by others. The fact that these same issues came up during independent applications of the method by different individuals to different systems lends credence to the robustness of the phenomena we describe; however, one must keep in mind that we are reporting anecdotal evidence, not the results of controlled experiments.

## THE COGNITIVE WALKTHROUGH: AN OVERVIEW

The Cognitive Walkthrough is a methodology for performing theory-based usability evaluations of user interfaces. Analogous to the traditional structured walkthroughs used by the software engineering community [17], the Cognitive Walkthrough has the goal of improving software usability by defect detection, amplification, and removal. Like other forms of usability walkthroughs [6, 11], Cognitive Walkthrough evaluations emphasize basic usability principles. In contrast to other types of usability evaluations, the Cognitive Walkthrough focuses on a user's cognitive activities; specifically, the goals and knowledge of a user while performing a specific task.

The Cognitive Walkthrough is designed to be used iteratively, early in the design cycle, by individuals or

groups. Either software developers or usability specialists can perform the Walkthrough. It is a task-based methodology that serves to focus an evaluator's attention on the user's goals and actions, and on the system affordances that support or hinder the effective accomplishment of those goals. During the Walkthrough, the steps required to accomplish a task are evaluated by examining the behavior of the interface and its effect on the prototypical user. Both problematic and successful task steps are recorded. Steps are deemed successful if the expected goals and knowledge of the typical user would result in selection of an action that leads the user closer to her ultimate goal; they are problematic otherwise.

The Cognitive Walkthrough is based on a theory of exploratory learning, *CE+*, and some corresponding interface design guidelines, *Design for Successful Guessing*, geared toward *Walk Up and Use* systems [12]. A Walk Up and Use system (e.g., automatic teller machine or airport information kiosk) supports the notion of *learning by doing*[1]. Since the Walkthrough is tightly coupled to the above theory and guidelines, each step in a Walkthrough mirrors the underlying theory by testing whether these principles have been followed in the interface's design.

The Walkthrough is a form and task-based methodology, whereby a task is evaluated by completing a set of forms, each form comprising several evaluation steps. Each step, in turn, is designed to address underlying theoretical concepts through a list of questions to be asked about the interface. See Figure 1 which contains a portion of a Cognitive Walkthrough form to be used when evaluating each interface action [16]. For example, to determine if the user is likely to choose the appropriate action at a given stage of a task, the Walkthrough asks how well an identifier (e.g., a button labelled "time") is linked to the needed action (e.g., pressing the button) and how well the needed action is linked to the user's current goal (e.g., to set the time on an alarm clock).

The Walkthrough method consists of three basic phases: a preparation phase, an evaluation phase, and a result interpretation phase. The forms guide the evaluators through the preparation and evaluation phases with detailed instructions; the interpretation phase, however, is more ad hoc. The preparation phase is used to gather and record basic system information prior to the evaluation phase. For example, the suite of tasks to be evaluated is identified and information about the users is noted. In the evaluation phase questions like those in Figure 1 are asked of each step within a given user task. And finally, in the interpretation phase all information gathered and recorded from the Walkthrough process is interpreted according to the following metrics: positive responses support the inference that the interface is good, whereas negative answers highlight steps that are difficult for the user. The results thus hint at interface problems and the necessary changes.

1. People use learning by doing in situations where they are knowledge poor, and hence must rely on feedback from the interface to shape or refine their knowledge and behavior.

...

**Step [B] Choosing The Next Correct Action:**

**[B.1] Correct Action:** Describe the action that the user should take at this step in the sequence.

**[B.2] Knowledge Checkpoint:** If you have assumed user knowledge or experience, update the USER ASSUMPTION FORM.

**[B.3] System State Checkpoint:** If the system state may influence the user, update the SYSTEM STATE FORM.

**[B.4] Action Availability:** Is it obvious to the user that this action is a possible choice here? If not, indicate why.

How many users might miss this action (% 100 75 50 25 10 5 0)?

**[B.5] Action Identifiability:**

**[B.5.a] Identifier Location, Type, Wording, and Meaning:**

____ No identifier is provided. (Skip to subpart [B.5.d].)

Identifier Type: Label Prompt Description Other (Explain)

Identifier Wording_____

Is the identifier's location obvious? If not, indicate why.

**[B.5.b] Link Between Identifier and Action:** Is the identifier clearly linked with this action? If not, indicate why.

How many users won't make this connection (% 100 75 50 25 10 5 0)?

**[B.5.c] Link Between Identifier and Goal:** Is the identifier easily linked with an active goal? If not, indicate why.

How many users won't make this connection (% 100 75 50 25 10 5 0)?

...

**Figure 1: Excerpt From a Walkthrough Form**

## THE THREE INTERFACES EVALUATED

We have conducted Cognitive Walkthroughs for three different applications that are much more complex than the Walk Up and Use interfaces to which the method has previously been applied. Since the three systems are intended to be productively used by casual or intermittent users (users similar to those the technique is designed to support), the method should still be applicable. Each of the applications was designed for a different domain and class of users, and utilizes a different interface style. The three applications we evaluated are HP-VUE, REPS, and BCA.

### The HP-VUE System

HP-VUE is a visual interface to the Unix operating system.[2] It provides graphical tools for manipulating files, starting and stopping applications, requesting and browsing help, controlling the appearance of the screen, etc. A "beta-test" version of HP-VUE was evaluated.

The evaluation of this interface was done by a group of three software engineers similar to the actual designers of HP–VUE. Seven common tasks were evaluated; they were selected by someone who was expert in using the Walkthrough technique, rather than the evaluators

2. The Hewlett-Packard Visual User Environment (HP-VUE 2.0). HP-VUE is a trademark of the Hewlett-Packard Company. UNIX is a trademark of AT&T. The X Window System is a trademark of Massachusetts Institute of Technology. Motif is a trademark of the Open Software Foundation, Inc.

themselves. The set of paper forms used for this evaluation was developed by Wharton [16].

### The REPS System

REPS is an existing system to guide sales representatives through a sales call by providing them with necessary customer and product information. Sales representatives use the system frequently, but new users, who do not receive intensive training on the system, are introduced often. REPS is accessed via ASCII terminals using function keys; it does not support a graphical interface. At the time of the evaluation the system had been taken down because of user complaints. The application of the Cognitive Walkthrough was an attempt to locate and quantify the interface problems.

The evaluation team was made up of a system developer who helped to implement the system, a requirements analyst who served as an advocate for the users and joined the group after the initial design decisions had been made, and a cognitive psychologist who joined the team as a user interface consultant. Four task scenarios were selected by the requirements analyst and the cognitive psychologist. Three simple interactions (i.e., task scenarios) with the system were formally evaluated; paper forms similar to those described in [12] were used. In a fourth task scenario, no forms were used. Instead the group tried to answer the standard questions informally.

### The BCA System

BCA is a research prototype CAD tool intended to be used by electrical engineers to design the construction parameters for a bare printed circuit board and to get feedback on the manufacturability of the design. This design task is complex, requiring the specification of many parameters and the weighing of tradeoffs between design choices affecting, among other things, fabrication cost, board production yield, and electrical performance. The targeted BCA user is an infrequent user who needs to do these tasks between one and three times per year. BCA runs on a Unix workstation under X-Windows with a Motif-based interface. The version evaluated had the requisite functionality, but had never been previously tested by a real user.

The Walkthrough was done by the current BCA design team, made up of a project manager, four computer scientists, and one electrical engineer. Only one of these people had been involved in the original design and implementation of the system, and only one other person was familiar with the Walkthrough method. Two tasks were evaluated, one simple and the other complex. The set of paper forms used was developed by Wharton [16].

## EXPERIENCES, ISSUES, AND RECOMMENDATIONS

The purpose of this paper is to describe the issues that arose during the Cognitive Walkthrough evaluations of these three applications. Common themes emerged across all the evaluations, which we believe expose the strengths and weaknesses of the current version of the method. Elsewhere we have published more formal analyses of two of these evaluations, including information about the number and types of problems found [2, 5]. Our approach here is more anecdotal. We describe various aspects of our experiences, illustrating the issues with examples from the evaluations, and draw conclusions about improvements to the method.

The three interfaces described above are similar in important ways. First, all are intended to be used by casual users or novices, so the capability to be productive on the system without extensive training is important. Second, all support a broad range of functionality and complex tasks. And third, in all cases the functionality being evaluated has been implemented; we did not evaluate design mock-ups. We now discuss the key issues, experiences, and recommendations relevant to the Cognitive Walkthrough evaluations of these three applications.

### Task Selection, Coverage, and Evaluation

The first step in performing a Walkthrough is to select the tasks to be evaluated. Any interface of even moderate complexity supports dozens or hundreds of tasks and task variants, and only a small fraction of them can be evaluated. However, the Walkthrough methodology does not provide guidance on how to select tasks, because task selection is not within the scope of the underlying theory. Nevertheless, to achieve good results during a Walkthrough it is necessary to understand those tasks that are most useful to evaluate, how many tasks are needed for sufficient interface coverage, and issues that arise when evaluating tasks.

*How Realistic and Complex Should the Tasks Be?* The REPS and BCA evaluation teams selected both simple and complex tasks, evaluating the simple tasks first. This reflected their need to gain experience with the Walkthrough method before doing more complex tasks. The simple tasks tend to correspond to the functional decomposition of the interface by its designers; the more complex tasks correspond to compositions of functionality and relevant transitions among subtasks. Although all evaluators strove for realism in the tasks they selected, the length and complexity of the evaluation process resulted in tasks being selected that were simplifications of what users would do in these rich environments, limiting consideration to only the most direct path through the interface. Consequently, potential problems may have been overlooked.

From our experiences we have found that tasks that mirror a simple, functional decomposition of the interface typically do not expose many problems with the interface. On the other hand, doing a simple task first can provide the experience necessary to perform a more complex Walkthrough. In general, we have found that it is most important to choose realistic tasks which exercise key system functionality; such tasks often comprise multiple core functions of the interface. By doing so, the evaluation covers not just the elements, but their combination and any necessary transitions among the subtasks. An important tradeoff to consider is the degree of realism or complexity of any individual task evaluation versus the number of tasks that can be covered, assuming fixed and limited resources. It is important to select some tasks to be covered as realistically

as possible, which will often imply complex action sequences with multiple alternatives, as well as to choose other tasks to cover the full range of functionality.

*Where Should Boundaries Be Drawn?* Aside from complexity, many other issues need to be considered when determining which tasks to include in the evaluation. For example, evaluators must decide what constitutes the domain of the application. Should the tasks to be evaluated encompass only what the application currently does or what users will expect it to do? This was particularly an issue for HP-VUE, because HP-VUE provides access to functionality provided by both the X-Window system and the Unix operating system; however, the release evaluated did not cover all the functionality of X and Unix. Users have to switch to "native mode" to accomplish some of their tasks. When selecting the tasks to evaluate for HP-VUE, it was difficult to decide what to do about frequent tasks that straddled the boundaries of HP-VUE.

*How Many Tasks Are Enough?* All evaluations were able to cover only a small fraction of either the set of user tasks or the functionality of the interface. The evaluators were unable to commit the time to fully evaluate an interface of the complexity of the systems considered. To do so would easily take 100 hours or more, a figure inconsistent with the 1-2 sessions the evaluators believed a critique should take. As it was, the actual sessions were often 4-8 hours long. Because all evaluators stopped after 6-15 total hours and 2-7 tasks, we do not have measures of the number or kinds of problems that might have been uncovered had additional tasks been evaluated.

*What About Task Variants?* Once a candidate set of tasks is chosen, which variants to evaluate must also be considered. Rich interfaces often provide multiple ways to accomplish a task. Should all the paths be evaluated, or only the "most obvious" one for a given context? We have always evaluated only a single path assuming that this is the one the user would choose. Further, by decreasing the number of paths for a particular task, the number of tasks can be increased appropriately. Absolute tradeoffs between the number of tasks and paths, however, are not known.

*At What Granularity Should the Evaluation Be Carried Out?* Even when a suite of user tasks under evaluation can be refined to address these task selection and coverage concerns, other task-related issues arise when the evaluation is performed. Evaluators often have trouble deciding what the granularity of an individual action should be. For instance, should a user action consist of a meaningful set of keystrokes, e.g., a file name, or should each letter (keystroke) within the file name be counted as an individual user action? An example of this involves the HP-VUE interface, where one may log into the HP-VUE system by using one of two methods: (1) type login name followed by a <CR>, or (2) type login name and then mouse click on the "OK" button. It may seem that these methods would have identical implications, but depending on how the user approaches the task physically and mentally, the termination actions (i.e., pressing the <CR> button or the mouse click on "OK"), may lead to significant differences in the evaluation.

We believe the granularity of action evaluations needs to be determined with respect to the interface under evaluation. In most cases it seems that a reasonable collection of keystrokes, such as those used to input a file name, can be counted as a single user action. But if the interface requires different physical actions, such as keystroke and mouse click actions, then these should be treated as two separate actions. (cf. The approach of Card, Moran, and Newell's Model Human Processor [1], where a <CR> is always treated as a separate step.)

*What About Identical Subtasks?* Another issue is that of identical embedded subtasks within a set of different, larger tasks. When using realistic tasks, several different tasks may contain an identical subtask. Should this be treated as the same subtask each time it comes up, or will there be nuances of the different contexts that may change the evaluation of that subtask? When this situation arises during an evaluation process, the evaluators must decide whether to do a detailed evaluation of those identical aspects. During our evaluations we encountered some pairs of contexts in which the identical subtask would be performed differently. Had the evaluators treated the previously analyzed subtask as a "solved problem" in the second context, important problems would have been overlooked. Thus, for identical embedded actions, all transitions definitely should be evaluated, but it may be safe to short-circuit some of the actual subtask evaluations.

*What About a High-Level Treatment of User Tasks?* The final issue we raise concerns the Walkthrough's lack of a high-level treatment of the suite of user tasks. Because the tasks are evaluated at the granularity of individual user actions, there is no way to determine if the task as a whole evaluates well. It is only known how the individual user actions evaluate. This is a shortcoming of the method, because the designer also needs to know whether a task itself is sensible, non-circular, too long, or important to the user. The possibility of missing interface problems because the Walkthrough does not encourage evaluators to take a broad view of the task is one for which we have not found a solution. We have relied on informal critiques to ensure that such issues are not missed.

### The Process of Doing a Cognitive Walkthrough

As described previously, the Walkthrough methodology consists of a set of forms to be filled out while exploring user actions and system responses during one or more tasks. The central Walkthrough form asks a series of questions about a single atomic action in a task; thus, copies of this form are filled out dozens of times during a complete Walkthrough. Groups who adhered most closely to the Cognitive Walkthrough procedures found the repetitive form filling to be very tedious, enough so that it discouraged some evaluators from using the method in the future.

Similarly, the separation of bookkeeping requirements from the actual evaluation ended up being another procedural

impediment — often a wall was thrown up between those people who focused on the interface proper and those who focused on the recordkeeping. In our evaluations, we found that groups who rotated or otherwise shared responsibility for bookkeeping did better in this regard.

While doing a Cognitive Walkthrough, evaluators often noticed problems that were not directly relevant to the current task. For example, while working on a step that involves typing a file name into a text field, the evaluators might notice the absence of wild card options, even though the task being evaluated did not require the use of wild cards. It is always difficult to know what to do with such issues. Taking the time to resolve them greatly lengthens the Walkthrough and causes the group to lose context information about the current task that can be time-consuming to regain. On the other hand, it is easy to lose track of these side issues and never get back to them if they are not captured along the way.

Thus, the methodology needs explicit procedures for keeping track of side issues and changes needed in the interface, together with appropriate context to reconstruct the situation later. The group leader will still need to use judgment about when to allow such digressions and when to refocus the discussion, but the task may be made easier by having a formal way to table important discussion.

A similar situation arose when the evaluators were also the developers. They often wanted to pursue a problem beyond the limits of the Walkthrough — to design a fix, followed immediately with a Walkthrough on the fix. This, too, can be time-consuming, but developers seemed to want closure on understanding what better solutions were available.

We found the looser application of the method, as done by the REPS group on their final task, to be more successful and more satisfying to the evaluators. This group found the process to be less tedious than did any other group. This is consistent with other findings [14]. We believe the REPS group was successful with this "broad brush" approach because they had first explored the interface using the more structured methodology; they had learned when to be precise and when they could be more casual. However, we don't know how to formalize the circumstances under which each of the two different approaches would be most appropriate.

The evaluators suggested many changes to the procedural aspects of the method, several of them having to do with group aspects. One was to capture information that the entire group needed to refer to (e.g., assumptions about the user population) on overheads or flip charts, to make it more publicly available. Another was to share the various recordkeeping tasks, possibly rotating them between user tasks, so that all participants felt involved in both the critique of the interface and the identification of problems. Finally, since evaluators found the main source of tedium to be the task action evaluation form, they would have preferred to have a "review card" to summarize the relevant questions. Using this the evaluators could quickly go over the issues for a task step, only committing to the paper record those issues for which there was a problem. This would prevent them from overlooking any important aspects, while speeding up the process quite a bit.

#### Requisite Knowledge for the Evaluators

The Walkthrough methodology presupposes more knowledge of cognitive science terms, concepts, and skills than most software developers have. The forms refer heavily to the specialized vocabulary of cognitive science, containing terms such as *goal structures*, *activation of goals*, and *supergoal kill-off* [3]. The terminology could be changed (or softened as in [7]), but the concepts they represent are equally specialized and will be foreign to the typical software developer. Certain of these concepts are critical to the successful use of the method. For example, one evaluator questioned the whole notion of goals, and whether people actually break their primary goal into smaller subgoals when doing a task. How does one explain the Cognitive Walkthrough notion of supergoal kill-off to someone who does not presuppose the existence of goals?

Because of a lack of familiarity with terminology, misunderstandings can occur. The most common misunderstanding was seen in task decomposition, where goals were often indistinguishable from interface actions. For example, one group came up with the following set of goals for the task of *loading a file*:

find the file name in the browser
select file to be loaded
click to load the file
wait

These goals are essentially the interface operations that are performed. A more plausible decomposition might be:

determine the name of the file to use
select the file
load the file

Notice that "determine the name" is a purely mental operation that has no analog in the evaluators' list, "select the file" encompasses the first two goals in the earlier list, and "load the file" encompasses the latter two. But, it's not obvious that users would decompose loading a file into "click" and "wait" without either experience or feedback from the interface about the need to wait.

The Walkthrough developers have said that one of the most frequently asked questions they hear is "what is the difference between a goal and an action?" They have pointed out that even if goals have the granularity of interface actions, it shouldn't matter to the results of the Walkthrough, because if there is a mismatch between user goals and system actions, it will just show up in a different place in the evaluation. If goals are described as interface operations, the

3. The supergoal kill-off phenomenon occurs when an arbitrary final step, (e.g., typing carriage-return) is forgotten because a prior subgoal (e.g., type in the file name) gets associated with the complete goal; thus the user inadvertently considers the entire goal accomplished when only the distinguished subgoal is done.

evaluator will then have to justify why the user would form those goals from the system state, rather than justify why more realistic goals would lead to the needed actions [Clayton Lewis, personal communication, 1990]. We found that having goals at the granularity of individual actions did lead to problems. The example above demonstrates some of the issues that came up. In particular, for this interface, no salient feedback is given regarding the need to wait while the file is being loaded (which takes a long time). This problem was not noted by the evaluators because they had assumed that the user had the goal of waiting; they expected the user to be unconcerned about the amount of time before the system indicated completion of the load operation. Perhaps the evaluators should have questioned the applicability or obviousness of this goal, but in this instance, they found it to be perfectly natural.

In all but one of the Walkthroughs, at least one evaluator had more than a nodding acquaintance with cognitive science. This seemed to mitigate the effects of the specialized concepts and terminology reasonably well. The group that was least knowledgeable about cognitive science had the most problems at various levels — in task definition, in understanding the forms, and in accepting the results of the Walkthrough. Overall, we believe that it will be difficult to eliminate the need for cognitive science background both to make sense and to take full advantage of the technique. A better approach would be to enlist someone with at least a moderate HCI or Cognitive Science background as a member of the evaluation team.

### Group Walkthroughs

All the interfaces were evaluated by groups of two to six evaluators. In general, the method seems to adapt quite naturally to a group evaluation situation. However, in comparing the REPS evaluation, viewed as successful by the REPS team [2], and the BCA evaluation, rated as poor by its participants, we noticed a number of situational parameters that are correlated with the success of the method when applied with a group. These are: division of duties, size of the team, length of the sessions, and the length and format of the particular Walkthrough forms used.

In both of these evaluations, one 'walkthrough advocate' introduced the method to the evaluation team and also emerged as the discussion leader and facilitator. In both cases it appeared to be important to have this person be responsible for bringing the discussion back on focus when it had digressed. The key issue for leading a group walkthrough seems to be the flexible managing of the group process between expansion (giving the team members the chance to discuss related topics to prevent frustration) and turning the focus back on the method. Research by social psychologists on the functions of leaders in small problem solving tasks suggests that an effective leader must function as both a "task leader", who keeps the group focused on the current problem, and as a "social-emotional leader", who motivates members to work hard and coordinates inter-member interaction [9, 15].

The BCA evaluators felt that having six people on the walkthrough team was too many; they suggested a size of three. Having more people can make the discussion less focused; sharing the walkthrough forms and the interface becomes a problem; and the proportion of errors discovered per time investment of each team member is not economical.

The Walkthrough sessions can easily become too lengthy. The BCA session took six hours, whereas the REPS sessions were limited to two hours and spread over several days (as were the VUE sessions). In comparing these two different approaches we found that tiring the team with lengthy meetings seemed more damaging to the process than the possibility of losing momentum between meetings. In the REPS evaluation the previous task evaluation was summarized by two of the group members between meetings, which helped the team stay focused and kept its members aware of already localized problems.

In both the REPS and BCA evaluations we observed two positive side effects not covered by the theory. First, the Walkthrough served as a method of learning about the importance of considering the background knowledge and environment of the intended users, and how to do a careful task analysis. For both the REPS and BCA applications the team members' consciousness about these issues has been raised and they have been taken into account in further steps of those projects.

Second, the Walkthrough served as a method of mediation between the requirements and the development side of the design team. When the design team is divided between requirements analysts and system developers, a communication gap may easily develop between them. Both types of expertise — detailed knowledge of the users' tasks and needs, *and* the options and constraints of the development platform — is needed to optimize the design of a particular interface. In our experience the Walkthrough application gave both parties a neutral ground and shared vocabulary to negotiate these design decisions, and hence to make optimal use of their complementary expertise. Similarly, having a 'Walkthrough advocate' on the team seemed to help to establish this focus on the interface issues and prevent digressions into old conflicts. Research on group decision making has frequently demonstrated that the decomposition of a global decision problem into its components, analysis and decision on each component separately, and then recombination into a group solution is an effective method to reduce or eliminate conflict [e.g., 4, 10]. The Walkthrough may provide similar benefits by imposing an analogous highly-structured, task-driven, component-by-component organization on the group process.

### Interpreting Cognitive Walkthrough Results

According to the Cognitive Walkthrough originators [Clayton Lewis, personal communication, 1991], the Walkthrough does not identify problems with an interface; it identifies mismatches between system affordances and user goals. Because of the nature of the underlying theory, the

Walkthrough seems to do a better job of pointing out mismatches that are linguistic in origin (e.g., mislabelled buttons or menu items) than those of a more graphical nature. Whereas the Walkthrough does inform developers of the aspects of the interface that are troublesome, identifying specific problems and generating their solutions are tasks beyond the scope of the Walkthrough proper, although the data generated by doing a Walkthrough would be highly relevant to such a task. In practice, our engineer evaluators went beyond identifying mismatches to identifying problem statements and often immediately to solutions.

The Cognitive Walkthrough technique can sometimes lead evaluators to solutions that are suboptimal. In one task evaluated, the user was required to go back and forth between different windows of the system, know which data entry fields were required and the correct sequence in which they had to be entered, with no cues from the system. The solution suggested by the evaluators was to highlight the data entry field or calculation that was to be done next. This solution is consistent with the Walkthrough forms, which focus on the salience of the appropriate action at the current step, but it is suboptimal to a solution that includes reorganizing the data entry fields and calculation buttons in a task-oriented manner.

We also found that the Walkthrough method can lead evaluators to propose erroneous solutions. For instance, one evaluation team proposed a change that would produce a simpler action sequence in the context of the task they were evaluating without realizing that it would remove functionality required for other user tasks.

As the above examples point out, by deriving solutions for a particular task, the evaluators may not be able to see the larger picture and may come up with an inappropriate problem statement or solution. In spite of this most of the problems identified by the evaluators were appropriate assessments of the interface they were analyzing. Occasionally, the narrow focus on individual task steps introduced what are essentially *set effects*, where the evaluators were so focused on a particular set of solutions that they were unable to recognize when a problem required a solution outside of this set. Consequently, methods that assess an interface more globally are needed as a supplement to mitigate these problems.

## CONCLUSIONS

For the first time the Cognitive Walkthrough has been applied to highly complex, high functionality systems by groups of software engineers. If the Walkthrough is to be successful in helping to design better software systems, it must be able to deal with real systems in environments such as those we tested.

Is the Cognitive Walkthrough ready for use by real development teams on high functionality applications? Based on our experiences, we say "not without substantial extensions". There are simply too many caveats for successful use of the method. We believe that these problems can be overcome by further research and extensions to the technique; however, widespread practical use must await those extensions.

The problems with Cognitive Walkthroughs are at two levels. The first involves process mechanics. Those can be mitigated by straightforward changes to the Walkthrough process; specific recommendations for such changes are summarized in Figure 2. The second class of problems involves limitations in the method as it currently exists. It is those limitations that keep the method from being viable at this time. We hope that the developers of the Cognitive Walkthrough will address these issues in their research.

One limitation of the Walkthrough method is that it doesn't match well with current software development practice, at least in the organizations where these evaluations were done. The developers we work with are interested in usability issues, but don't have the training or the inclination to become expert on the topic. Furthermore, usability is only one of a large number of aspects of the product they must focus on — aspects like reliability and performance demand equal attention, and are more clearly understood by software

**Task Issues**

- Start with a simple task and then work on more complex tasks. Include at least one task whose complexity matches what users will typically encounter. Balance the complexity of individual tasks with the range of functionality to be covered.
- Chose realistic tasks that exercise key system functionality. Look for tasks that cover multiple core functions, so as to evaluate transitions between subtasks.
- When selecting the tasks consider issues of: task granularity, action granularity, identical subtasks, and task variants.

**Process and Group Issues**

- At least one member of the group should be familiar with the terminology and concepts of cognitive science and with the Cognitive Walkthrough's process and assumptions.
- Provide aids to help organize the task. Possible aids include: a mechanism for keeping track of side issues and solution alternatives for later re-examination, a review card or other mechanism to summarize the questions to be asked at each step, overheads or other group-visible means to capture information frequently referred to, such as assumptions about user knowledge.
- Minimize the impact of bookkeeping tasks. Rotating the tasks is one method; tools to automate the process are another.
- The leader/facilitator needs to pay careful attention to group process issues. Things to consider: size and composition of the group, ensuring appropriate participants' expectations (as to time required, nature of the process, etc.), and the need for effective group management skills, since the walkthrough may expose existing conflicts among group members.
- Break the walkthrough up into sessions of reasonable length (2-3 hours). It is important to break only between tasks, so that context carefully built up will not be lost.

**Figure 2: Some Recommendations for an Effective Cognitive Walkthrough**

engineers. Developers are under constant time pressure; thus, any technique that takes more than a few person-days would need very strong support, both among management and the technical community to be worth jeopardizing product schedules. They are looking to identify the most critical interface problems; it's hard to relate the detail-oriented procedures of the Walkthrough to the identification of high-impact, pervasive problems. Finally, developers need solutions. The intentional interposing of an intermediate step between problem identification and solution may have benefits, but developers see it as a roadblock to their primary goal in doing the evaluation.

The issues above create problems for other task-oriented, developer implemented usability methods as well. There may need to be significant changes in software development life-cycles before any developer implemented methodology can be successful. On the other hand, requiring major changes to current practice is a serious impediment to the widespread use of Cognitive Walkthroughs.

Another class of limitations concerns task selection. Effective task selection is a critical issue for any methodology based on user tasks. Furthermore, since the Cognitive Walkthrough is intended to be used by developers, the evaluators may not have access to the informed intuitions of a user interface professional to select appropriate tasks. While, in principle, a task selection methodology could be added to the method, the field of HCI simply does not know enough about task selection for a pragmatically viable method to be developed at this time. We believe that, for the foreseeable future, the task suite should be generated by a knowledgeable professional. Finally, the Cognitive Walkthrough is limited by its focus on the lower level interface issues. Evaluators need to be provided with a process for stepping back and looking at the application more globally. This is probably done most directly by combining Walkthroughs with other evaluation methods. However, it could be done as an extension of the Walkthrough method.

The challenges identified above — incongruence with current software development practice, task selection, and the lack of focus on higher level issues — are not unique to Cognitive Walkthroughs. They must be resolved for any user interface evaluation method based on tasks and applied by the developers themselves to be successful. We hope that this paper will spur research into ways to successfully incorporate the Cognitive Walkthrough and other methods of this type into current practice.

## ACKNOWLEDGMENTS

We would like to thank the following people for their contributions to this work: Lucy Berlin, Nancy Bruner, Norman Chang, Scott Conradson, Felix Frayman, Bruce Hamilton, Reid Hastie, Stan Jefferson, Clare-Marie Karat, Nancy Kendzierski, Dan Kuokka, Clayton Lewis, Catherine Marshall, Jakob Nielsen, Vicki O'Day, Andreas Paepcke, Peter Polson, John Rieman, Terry Roberts, Craig Zarmer, and the anonymous CHI'92 reviewers.

## REFERENCES

1. Card, S.K., Moran, T.P., and Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
2. Franzke, M. Evaluation Technique Evaluated: Experience Using the Cognitive Walkthrough. *Proc. of the Bellcore/BCC Symposium on User-Centered Design.* Livingston, NJ, November 1991.
3. Gray, W.D., John, B.E., and Atwood, M.E. The Précis of Project Ernestine, or An Overview of a Validation of GOMS. *Proc. ACM CHI'92.* (Monterey, California, May 3-7, 1992).
4. Hammond, K.R. and Adelman, L. Science, Values, and Human Judgement. *Science*, 1976, Volume 194, pp. 389-396.
5. Jeffries, R., Miller, J.R., Wharton, C., and Uyeda, K.M. User Interface Evaluation in the Real World: A Comparison of Four Techniques. *Proc. ACM CHI'91.* (New Orleans, Louisiana, April 27 – May 2, 1991) pp. 119–124.
6. Karat, C., Campbell, R., and Fiegel, T. Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation. *Proc. ACM CHI'92.* (Monterey, California, May 3-7, 1992).
7. Lewis, C., and Polson P.G. Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. Tutorial presented at *ACM CHI'91.* (New Orleans, Louisiana, April 27 – May 2, 1991).
8. Lewis, C., Polson, P., Wharton, C., and Rieman, J. Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces. *Proc. ACM CHI'90.* (Seattle, Washington, April 1 – 5, 1990) pp. 235–242.
9. McGrath, J.E. *Groups: Interaction and Performance.* Prentice-Hall, Englewood Cliffs, NJ, 1984.
10. Neale, M.A. and Bazerman, M.H. *Cognition and Rationality in Negotiation.* Free Press, New York, NY, 1991.
11. Nielsen, J. Finding Usability Problems Through Heuristic Evaluation. *Proc. ACM CHI'92.* (Monterey, California, May 3-7, 1992).
12. Polson, P.G., and Lewis, C. Theory-Based Design for Easily Learned Interfaces. *Human-Computer Interaction,* 1990, Volume 5, pp. 191–220.
13. Polson, P., Lewis, C., Rieman, J., and Wharton, C. Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. To appear in *International Journal of Man-Machine Studies,* 1992.
14. Rowley, D.E., and Rhoades, D.G. The Cognitive Jogthrough: A Fast-Paced User Interface Evaluation Procedure. *Proc. ACM CHI'92.* (Monterey, California, May 3-7, 1992).
15. Shaw, M.E. *Group Dynamics: The Psychology of Small Group Behavior.* 3rd edition. McGraw-Hill, New York, NY, 1981.
16. Wharton, C. *Cognitive Walkthroughs: Instructions, Forms, and Examples.* Technical Report University of Colorado at Boulder, Institute of Cognitive Science, 1992.
17. Yourdon, E. *Structured Walkthroughs.* 4th edition. Yourdon Press, Englewood Cliffs, NJ, 1989.