Journal of Visual Languages and Computing (2002) **13,** 517–544
doi:10.1006/S1045-926X(02)00042-3 available online at http://www.idealibrary.com on IDEAL®

Improving the Design of Visual Programming Language Experiments Using Cognitive Walkthroughs☆

A. J. Ko†, M. M. Burnett†, T. R. G. Green‡, K. J. Rothermel† and C. R. Cook†

† *Department of Computer Science, Oregon State University, Corvallis, OR 97331, U.S.A. E-mail:burnett@cs.orst.edu and* ‡ *Computer-Based Learning Unit, University of Leeds, Leeds LS2 9JT, U.K.*

Received 23 January 2001; accepted 6 June 2002

Visual programming languages aim to promote usability, but their usability is rarely examined scientifically. One reason is the difficulty of designing successful experimental evaluations. We propose the Cognitive Walkthrough (CW) as an aid to improve experiments' designs. The CW is an HCI technique designed for evaluating interfaces. The technique focuses on the potential difficulties of novice users and is therefore particularly suited for evaluating programming situations that arise in visual programming language experiments performed in laboratory settings. We first describe an empirical study performed without benefit of a walkthrough and show how the study was improved by a series of walkthroughs. We also describe two other empirical studies and how they were improved with the help of the CW. We found the method to be quick to use, effective at improving the experimental design, and usable by non-specialists.

1. Introduction

The professed *raison d'être* of most visual programming languages (VPLs) is usability [1]. Yet until recently it has been regrettably rare to present any but the most cursory test of whether a VPL is usable. One reason, we believe, is the difficulty of designing well-controlled experiments involving VPLs. This paper describes an approach to helping with this task of experimental design.

Designing a valid VPL experiment is difficult, not so much because advanced knowledge of experiment design is needed, but because the results can easily be disturbed by extraneous problems. For example, participants might not understand instructions, or they might not be able find the right button to press. These factors can create so much experimental noise that even a well-designed VPL being used in a well-designed environment can appear to be quite poor. Technically, such random perturbations come under the heading of *uncontrolled situational variables*, and in this context, they pertain to the variability in the way participants can reason about the VPLs being studied.

☆ An earlier version of portions of this paper appeared in [12].

The scale and pervasiveness of this problem is shown by Gurka and Citrin's review [2] of empirical studies of the effectiveness of algorithm animation. They list six variables that are often uncontrolled. Although some of these variables are specific to algorithm animation, such as animation quality and the difficulty of the algorithms being animated, the implication to VPL experiments is clear: designing valid experiments involving the combination of humans, interfaces, and programming logic is difficult.

How can designers of VPL experiments detect potential uncontrolled situational variables at an early stage, before performing the actual experiments? Often experimenters use methods such as pilots and protocol analyses but these must be done late in the design process. To be more viable, a method would be usable much earlier, and would therefore be less resource intensive. There is a parallel here to evaluating a design for a user interface, where as many usability problems as possible should be detected and eliminated before performing actual user testing. This paper draws on that parallel by using the HCI technique of the Cognitive Walkthrough (CW) [3] as a means to identify uncontrolled variables in experiments' designs.

The CW is familiar in HCI as a tool to improve interface usability. But improving an *experiment's design* is not the same as improving an *interface*. Interface details are only part of the issue. For example, in an experiment, a tutorial must be designed to prepare the participants for the experimental task. If the tutorial does not cover exactly the right skills for the experimental task, or if the tutorial is not effective due to any number of pedagogical pitfalls, the experiment will not correctly measure the variables the experimenters are trying to measure because the participants will not have mastered the assumed set of skills. Other issues include the task design and the interplay between interface details and the realities of research systems.

In this paper, we report on three VPL experiments improved by the CW and investigate four research questions along the way:

Research Question 1: Is the CW effective in improving a VPL experiment?
Research Question 2: Does the CW possess advantages over traditional approaches to improving experiments?
Research Question 3: Is an HCI specialist necessary for effective use?
Research Question 4: What general lessons can be learned about experiment design?

2. The Cognitive Walkthrough

2.1. Background

The Cognitive Walkthrough [3] was primarily designed as a 'desktop' evaluation tool for usability engineering, aimed at predicting potential difficulties for novice users. Its strengths are that it requires neither human subjects nor a fully functioning model of the product, and that it rests on an acceptable cognitive model of user activity during the phase of exploratory learning. That model describes four phases of activity:

- the user sets a goal to be accomplished,
- the user searches the interface for available actions,
- the user selects an action that seems likely to make progress toward the goal, and

- the user performs the action and checks to see whether feedback indicates that progress is being made towards the goal.

To perform a 'classic' CW, first the evaluation team defines the expected users and estimates their prior knowledge (users who come with knowledge of many related or similar interfaces will have fewer problems than will those who come with little knowledge). Next the team prepares a detailed description of one or more tasks, and a list of action steps comprising the optimal sequence of execution.

During the actual walkthrough the team works through each step of the execution sequence, following a printed form to answer the following questions.

- Will the user form the right goal?
- Is an appropriate action readily available?
- Will the user find that action?
- Will the user know that progress has been made?

The team notes steps where the user may not take the correct action. At the end of the walkthrough, they will have identified a set of problems associated with the system.

As first proposed [4], and considered solely in the context of evaluating interfaces, the CW proved reasonably successful. Subsequent revisions and refinements to the original CW procedure have addressed criticisms about the difficulty of determining the required level of granularity, the tightly specified procedure, and the problem of choosing appropriate tasks. Rieman, Franzke, and Redmiles presented a revised procedure for the walkthrough, involving small teams, rotation of walkthrough duties, and a slackening of walkthrough form use once it is familiar [5]. Refining the method further, Sears and Hess reported on the effect of task description detail on evaluator performance, noting that the granularity of description results in different types of problems discovered via the CW [6]. This result seemingly patched up many of the early complaints that the walkthrough provided no criteria for selecting task descriptions.

Extending past the CW's traditional use, Bell *et al.* showed that it could be adapted to evaluating the design for a programming language, which supplies evidence that it may be adaptable beyond its original domain of user interfaces [7]. The CW has also been used by Hundhausen and Douglas to evaluate a programming environment to be used in an empirical study [8], and by Lewis *et al.* in the design of educational activities [9]. Closest to our work is that by Stasko *et al.*, who used a CW*after* an experiment, to discover why their experiment failed to yield a significant result [10]. Their CW proved successful in uncovering several design flaws in their experiment.

2.2. Relevance to Empirical Studies of VPLs

Although HCI evaluation methods have been developed to evaluate the usability of interfaces, no studies, to our knowledge, have reported using the CW method or any other usability evaluation method to assist in *designing an experiment*. To address the problem of eliminating situational uncontrolled variables from VPL experiments, why choose the CW method over some other evaluation method?

Surveying the development of evaluation methods in HCI in general, it is clear that the field has not yet settled down, and no obvious choice emerges of an evaluation method that could be applied to experimental designs. There are a few examples of applying HCI

techniques directly to the design of VPLs and their environments: Green and Petre applied the Cognitive Dimensions framework [11,13], and Williams and Buehler applied the Keystroke Level Model [14, 15]. Both these approaches target experienced users who have found out how to work the system, rather than novices meeting the system for the first time, and are therefore inappropriate for evaluating an experimental situation.

Quite a few HCI evaluation methods have been proposed. Attempts to compare the efficacy of different methods [16, 17] have been partially vitiated by weaknesses in the design of the comparisons [18]. The choice of which method to explore for evaluating experiments needed therefore to be determined by apparent rather than demonstrated suitability. The CW appeared particularly suitable to evaluate VPL experiments in laboratory settings for two reasons:

- the CW is intended for users new to an interface, and
- the CW has potential for use by non-specialists.

2.2.1. Focus on Users New to a System

When a VPL is to be evaluated in a laboratory setting, we can be sure that the system is new to its users. We can also be sure that they will not have adequate time to progress to being experienced users. Thus, HCI methods that target experienced users, such as GOMS [14] are inappropriate here.

VPLs include user interfaces; the CW method focuses on users who are new to a given interface and who are trying to figure out how to use it. It does not focus on speed and accuracy, but on *reasoning*. The tasks given to VPL experiment participants are programming-related, which emphasize reasoning. These factors make the CW particularly suitable for evaluations of people's ability to perform programming tasks in an experiment. Further, the creators of the CW approach have themselves previously successfully applied it to a VPL [7], although they ruefully noted some of the things they missed [19].

Other evaluative methods derived from HCI typically operate at a more surface level, paying less attention to reasoning and knowledge. Heuristic Evaluation [20], for example, addresses a very mixed bag of questions, none of which deal with the participants' prior knowledge. In a very different style, but equally much concerned with surface issues, the Keystroke Level Model [21] is good for predicting how fast a user can perform a well-learned routine task, but has no relevance to non-routine tasks with a high element of reasoning.

2.2.2. Potential for Non-specialist Use

The CW method also shows potential for being usable by computer scientists without the assistance of an HCI or cognitive science expert. This is in contrast to many HCI approaches, which explicitly assume HCI or cognitive science expertise (e.g., GOMS [14], Task Knowledge Structures [22], and Interaction Framework [23]). Neilsen's Heuristic Evaluation is an exception, but as we have seen, that approach is not relevant to our needs.

However, evidence for non-specialist usability is scanty and mixed. On one hand Wharton *et al.*, after evaluating three user interfaces, identified a few important weaknesses of the CW process, and claimed 'it will be difficult to eliminate the need for a cognitive science background both to make sense and to take full advantage of the technique' [24]. On the other hand, John and Packer reported a case study in which 'the

Cognitive Walkthrough evaluation technique [was] learnable and usable for a computer designer with little psychological or HCI training' [25]. Their single-user case study is encouraging but data about a single user can hardly be considered a definitive demonstration. In this paper, further data supporting their findings are supplied.

3. Experiments 1A and 1B: Evaluations of a Visual Testing Methodology

Our first experience using the CW to improve a VPL experiment design was motivated by counterintuitive results of a previous experimental evaluation of the effectiveness of a visual testing methodology (described below). In this section we describe the 'before' experiment (Experiment 1A), discuss our use of the CW to refine it, and describe the refined 'after' experiment (Experiment 1B). Lastly we compare the outcomes of the 'before' and 'after' experiments.

3.1. The Visual Testing Methodology

The experiments described below evaluate a visual testing methodology, applicable to commercial spreadsheet systems as well as some VPLs [26, 27]. The methodology incrementally analyzes, behind the scenes, the relationships among spreadsheet cells and how thoroughly tested each relationship is. It provides immediate visual feedback about the 'testedness' of the spreadsheet, which may change as the user edits formulas. We term this methodology the 'What You See Is What You Test' (WYSIWYT) methodology. The experimental question to be answered was whether spreadsheet users can successfully employ this methodology.

Part of the WYSIWYT methodology is a tight integration with its host VPL. Our experiments used the research VPL Forms/3 [28, 29]. In Forms/3, ordinary formulas, contained in movable cells, can be used for both numeric and graphical computations. Figure 1 contains a program (spreadsheet), one of the two used in the experiments. The WYSIWYT testing methodology can be seen in several of the features of Figure 1. The fine granularity of integration can be seen in the figure: there is no separate window for testing, no separate testing mode, etc. Instead, all testing information is combined with the display of the program and the values. In the figure, red cell borders (shown as light gray in this black-and-white paper) indicate that a cell is untested, blue cell borders (black in this paper) indicate that a cell is fully tested, and purple cell borders indicate partial 'testedness'. Users record decisions that values are correct by checking off the checkboxes in the upper right corner of cells (some checkboxes contain ?'s in the figure). Users can also invoke other visual devices, such as dataflow arrows (between subexpressions or entire cells) that follow the same 'testedness' colors.

3.2. Experiment 1A: the 'Before' WYSIWYT Study

The aim of the 'before' study was to determine whether, by using the WYSIWYT technology, participants would produce better-tested spreadsheets. The participants, students from three computer science courses, were randomly assigned to three conditions. The *Ad Hoc Group* did not have access to the WYSIWYT testing methodology. The other two groups, the *WYSIWYT-No-Training Group* and the *WYSIWYT-With-Training Group*, did have

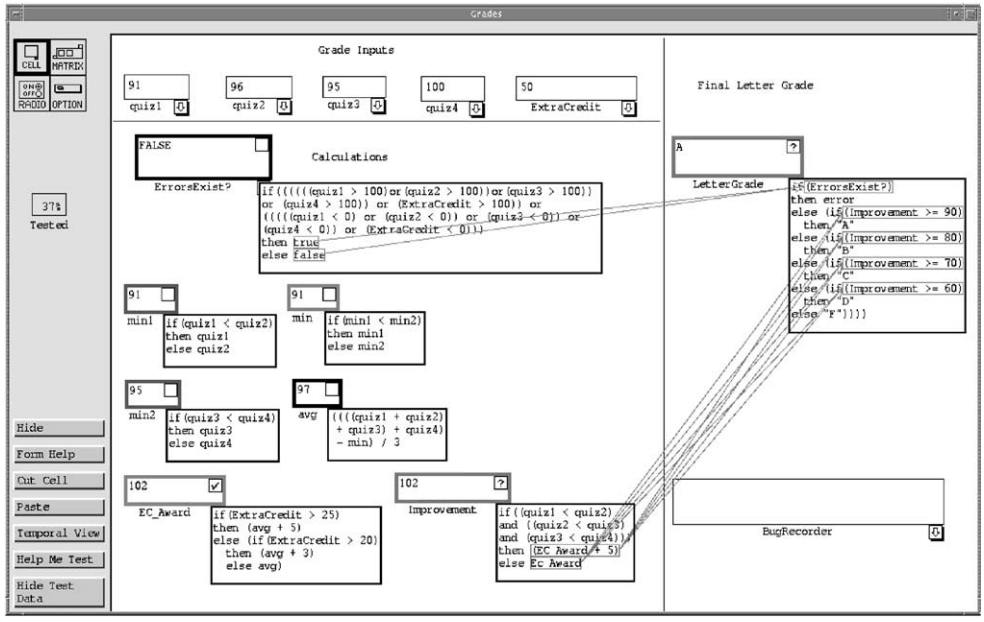

Figure 1. A Forms/3 spreadsheet that calculates a student's grade. Cell relation arrows in and out of formulas are colored in the same way borders are colored: from red (untested) to blue (tested), light gray to black in this paper

access to the WYSIWYT methodology. The experiment started with a 25-min interactive tutorial of Forms/3, in which each participant actively participated by working with the example spreadsheets on their workstations as instructed by the lecturer. The tutorial introduced all participants to language features and environmental features. Both WYSIWYT groups' tutorials also included instruction on the testing interface, and the With-Training Group received additional instruction on the underlying testing theory. Following the tutorial, all participants were given two spreadsheet problems to test, one of which is shown in Figure 1. Problem order was counterbalanced.

The statistical outcomes of this study were mixed. One major hypothesis had significant results, and some minor analyses showed or came close to showing significance, but most hypotheses lacked significant results. We felt a plausible reason for this was because of various uncontrolled situational variables we had not identified before the experiment. In order to refine the experiment, we used three CWs to help identify the situational variables we had not controlled.

3.3. A Walkthrough for Common Testing Tasks

Our team, consisting of 10 members, did not have previous experience using the CW. Thus one of the authors (Green), an HCI expert, gave a brief explanation of the ideas behind the CW, followed by a presentation of a few example walkthroughs to teach the specifics of the technique. This introduction to the CW lasted about 90 minutes.

In order to gather relevant information for Research Question 3, the HCI expert monitored the first walkthrough. His functions were (1) to answer questions about the CW that came up along the way, and (2) to observe whether the team seemed to be able to use the CW technique effectively. A few members of the team were selected to be recorders, and a single member of the team was chosen to be the facilitator (*recorder* and *facilitator* being roles defined in the CW process).

First, the 10 members of the team filled out the setup form required by the CW, which is presented in Figure 2. The *task description* was the same as the work to be done in the experiment: the participant had to test the spreadsheet. The *action sequence* was a bit more difficult to construct; what was an optimal sequence of atomic actions for completing the task? In the literature, the action sequence has been criticized as difficult to prepare and evaluate, but Sears and Hess provide limited empirical guidelines for preparing this sequence of actions [6], and Rieman *et al*. [5] provide general rules for choosing tasks and constructing action sequences. In this experiment, participants must open the formula, then change the formula, and then validate the output, and these are expressed in the 'Action Sequence' in the figure.

Next, looking ahead to future studies, we chose our *anticipated users* to be end users, even though our actual participants would initially be computer science students. To keep our thinking about end users concrete, we chose a real person as a model, whose experience and skills were known to us. We picked a department employee experienced with spreadsheets and formulas as a representative end user and used that person as a model for 'Martha', a fictional participant. Finally, the team explicitly set the *user's initial goals*: to test the spreadsheet.

After completing the startup form, the team began on action (1), 'double-click on the formula tab' (see Figure 3). First, the CW poses the question, 'What are the current goals for this point in the interaction?' Since the final goal is to validate a cell, a good goal for

Cognitive Walkthrough Form: Start up sheet.
Experiment: *WYSIWYT experiment*
Task: *Validate the output corresponding to a set of inputs on the Grades spreadsheet*

Task Description. Describe the task from the point of view of the first time user. Include any special assumptions about the state of the system assumed when the user begins to work.

The task is to test the spreadsheet (this is what they'll be told). The system will be in a state such that someone could immediately start testing.

Action Sequence. Make a numbered list of the atomic actions that the user should perform to accomplish this task.

The optimal sequence of actions: Change an assignment grade to a different value by (1) double-clicking on the formula tab, (2) changing the window focus, (3) entering a value, and (4) accepting, (5) checking the final output box (hopefully the user will choose the "final grade" cell), and (6) repeat for different inputs.

Anticipated Users. Briefly describe the class of users who will use this system. Note what experience they are expected to have with systems similar to this one, or with earlier versions of this system.

People who have experience with spreadsheet basics, but limited experience inventing spreadsheet formulas. They should have basic algebra skills, and will have gone through the oral tutorial, but will not have had other Forms/3 training.

User's Initial Goals. List the goals the user is likely to form when starting the task. If there are likely goal structures list them.

We think it's going to be "test the spreadsheet," rather than something more concrete like "change an input value."

Figure 2. A Cognitive Walkthrough startup sheet. (The Cognitive Walkthrough forms portrayed in this paper are slight revisions of what Polson *et al*. presented [3]. We modified them in the presence of the HCI expert by removing the questions regarding the percentage of users likely to commit an error, which the HCI expert advised were not crucial.) Italicized text represents a summary of the notes the evaluators took during the walkthrough

Cognitive Walkthrough Form: A Single Step
Task: *Double click on the formula tab*
Action #: *1*

1. Goal Structure for this step
1.1 Correct Goals. What are the current goals for this point in the interaction?
Change the formula of a cell.

1.2 Mismatch in likely goals Check each goal in this structure against your analysis at the end of the previous step. Will all users have dropped it or failed to form it? Are there unwanted goals that will be formed or retained by some users?
Since this is the first step, there is no previous step.

Figure 3. The first two questions (and answers) of the form for the first action in Experiment 1B's first Cognitive Walkthrough

Martha would be to change the formula of a cell; a group member who had been given responsibility for recording the CW recorded this. The next question requires examination of whether this 'good' goal is likely, asking if there might be a mismatch in the intended goals and the actual goals of the participant. Since at this point, Martha had (hypothetically) just started the experiment, a team member recorded 'no'. The facilitator (the member of the group responsible for leading the CS) directed the group through the whole set of questions for action (1), and then proceeded to answer the same set of questions on the rest of the actions. The CW details on these actions are given in Appendix A.

By the time the set of questions had been answered for each of the six actions presented in Figure 2, the team had identified a set of potentially uncontrolled variables, which are listed in Table 1. We identified four types of situational variables in the experiment:

- Problem Design: These were aspects of the testing problems used in the experiment that could have caused participants to puzzle about elements unrelated to the WYSIWYT methodology.
- Tutorial Design: Many of the assumptions we made about the audience in the CW setup rested upon the tutorial's coverage of the necessary material and the participants' presumed mastery of the introductory tutorial material.
- Testing-Related User Interface: The user interface part of the WYSIWYT methodology needed improvements to make the action choices clear to the participants and future users. This subset of issues is approximately the same subset that would have been identified by a classic use of the CW for evaluating the user interface component of the methodology.
- Unrelated User Interface Distracters: Previous evaluations of the non-testing part of the user interface were never done in the context of testing tasks. Since our first CW focused on our participants' ability to do the task (test this spreadsheet), it turned up ways this mature interface could distract participant's attention away from testing, leading to more uncontrolled variables.

Some of the Problem Design issues can be seen in Figure 1. Note that each issue introduces distractions: the *ErrorsExist*? cell in the upper-left contains extensively nested parentheses, requiring participants to expend extra energy on parsing; cell *LetterGrade* on the right exhibits several nested conditionals, requiring participants to focus on the flow of logic; and cells *min*, *min1*, and *min2*, contain conditionals that would normally be handled by a *minimum* operator in most spreadsheet systems. Since the redesigned study would first be administered on computer science students, who are familiar with such

Table 1. Specific issues identified during our first walkthrough

Issue	Potential solution
Problem Design issues	
Nested 'if' conditionals might confuse participants.	Alter the problems so that nested conditionals are not required.
Overabundant nested parentheses might confuse participants.	Change the parsing engine to allow for fewer parentheses or else alter the problem formulas.
'Min' and 'max' operators are absent, introducing additional 'if' conditionals.	Implement 'min' and 'max' operators or design problems that do not need them.
Indentation is often lacking, reducing readability.	Indent cell formulas in a consistent and readable manner.
Tutorial Design issues	
Length of tutorial may be too long to hold the participants' attention, thereby invalidating our assumption about what the participants know.	*Eliminate unnecessary details and integrate methodology into the explanation of test feedback, for reduction of length. Eliminate the With-Training Group and therefore the training section of the tutorial.* *
Absorption of tutorial material may not be complete because of lack of participant practice.	*Attempt to balance the practice period time for all groups after the tutorial and before the problems.* (*The previous experiment provided a longer practice period for the Ad Hoc than the WYSIWYT groups, to equalize the treatment length.*)
Testing-Related User Interface issues	
While there is significant feedback when users validate a set of inputs, there is no feedback indicating long-term progress.	*Implement a 'testedness' indicator, showing the percentage that the user has tested a spreadsheet.*
Participants may not be able to determine what ranges of inputs to try in order to test different parts of the program.	Show cell relation arrows in the background of the spreadsheet.*
Participants do not have a way to undo validations in the event that they want to repeat validations or compare them to other options.	*Implement an undo feature.**
Unrelated User Interface issues	
Cell formula tabs do not suggest the action 'edit this cell's formula.'	*Bring up the formula edit window whenever a cell is selected.*
The formula edit window contains labeling problems with associating goals to appropriate actions.	*Add instructions at the top of the formula edit window, describing to the user what action is required next in order to change the formula.*

Table 1. (*continued*)

Issue	Potential solution
Editing a formula contains an 'and-then' structure, suggesting that participants may forget to click the accept button.	*Add a reminder the top of the formula edit window once a formula is entered in the field.*
Formula edit windows can get lost behind other windows.	*Attach the formula edit window to the bottom of the main spreadsheet window.*
The system generates fractions in ratio format, which can be misinterpreted as bugs.	*Change the format of fractional output to decimal format.*
The system is often slow to respond, due to multiple client applications on one server and frequent garbage collection.	*Reduce the number of client applications per server and implement better garbage collection timing.*

Note: Italicized text represents changes that were made as a result. Solutions marked with an asterisk (*) indicate solutions that were discussed before the CW was done, but were also revealed by the CW.

complexities, Problem Design changes were not made for Experiment 1B. However, these changes were incorporated in designing Experiment 2, because that experiment's participants were end users.

3.4. A Walkthrough for the '?'

With the results of the first CW, one of the significant changes we decided to make was the addition of a 'testing undo' feature, allowing participants not only to check but also to uncheck any cell. In the first walkthrough, we had discussed three cases where an undo would be necessary: a participant accidentally checks a cell, a participant does not notice what changed on the screen after a check and wants to see it again, or a participant wants to see the testedness of a different set of checks and removes checks in a different order than they were originally placed. While the proposed solution was straightforward in the first two cases, it was less clear in the third: undoing part of a set of checks may have made some blanks or '?' fail to reappear exactly as before, due to multiple checkmarks the user had placed providing duplicate coverage of some cells.

Worried about this inconsistency confusing the participants, therefore confounding the statistics, we performed a CW on this specific task in order to discover whether there was a quick solution, or whether a solution was required at all. A portion of this walkthrough is presented in Figure 4.

The walkthrough revealed that participants could indeed vary in reasoning about the third case, when it arose. However, we became convinced from the CW that few participants would encounter the situation because the tutorial did not encourage testing strategies leading to that case. The CW did, however, lead to a discussion of testing strategies that the tutorial did encourage, which revealed that there was no visual mechanism in the WYSIWYT interface that would suggest to participants a reasonable testing action to take next.

3. Modification of Goal Structure, Assuming the Correct Action has Been Taken

3.1 Quit or Backup. Will users see that they have made progress? What will indicate this?
The participants will see a change in cell border colors, a checkmark will appear, and arrow colors, if displayed, will also change.

3.2 Accomplished Goals. List all current goals that have been accomplished. Is it obvious that each has been accomplished?
The current goal of "check of the output cell" has been accomplished.

3.3 Incomplete Goals That Look Accomplished. Are there any current goals that have not been accomplished, but might appear to have been?
Since another goal is to test the whole spreadsheet, but the participant has received feedback, they may think that they've tested enough. Do we need some sort of indicator?

3.4 "And-Then" Structures. Is there an "And-Then" structure and does one of its sub-goals appear to be complete? How many users may prematurely terminate?
No, they just have to click.

3.5 New goals in response to prompts. Does the system response contain a prompt or cue that suggests any new goals?
If at this point the participant has the cells on the spreadsheet anything less than all blue, they will likely be cued to test more.

3.6 Other new goals. Are there any other new goals that the user will form given their current goals, the state of the interface, and their background knowledge?
We can't think of any.

Figure 4. A portion of the second CW for Experiment 1B. Italicized text represents evaluator responses and underlined text represents critical observations

To solve this problem, we used the discussion we recorded during the walkthrough for guidance. The team investigated expanding the existing checkbox testing mechanism in such a way that corresponded to the CW questions that first brought the discussion to this point (see Figure 4). The existing model was a three-state system: a checkmark represented the participants' belief that the cell's value was valid for the current inputs contributing to it, '?' meant a cell had been previously validated, but not for the current inputs, and a blank meant a cell had not ever been previously validated. The team's solution was to rearrange the meanings of the three states and incorporate into the new model another aspect of testing a cell: potential progress. A checkmark's meaning remained the same, and an empty box and a question mark would still indicate that a cell has not been validated for the current inputs. However, an empty box would also indicate that validation would make *no* progress in testing, while a question mark would indicate that validation would make *some* progress.

While the HCI expert did not participate in this walkthrough, he monitored a portion of it to be sure that the team was executing it in a reasonably correct fashion. His observation was that, although an HCI expert would have gotten more data from the technique than the team did without an HCI expert's participation, their use of it was still appropriate and within the boundaries of the CW method.

3.5. Did We Lose Control of Some Variables?

After making the italicized changes in Table 1 to the interface and other components of the experiment, a third walkthrough was performed for two reasons. First, the revisions to the experiment design could have themselves introduced new variables. For example, during each of the previous CWs, a number of questions on the CW forms were answered positively under the assumption that the user would retain all of the knowledge contained within the tutorial. Yet, we had since made changes to the tutorial. Second, some of the changes identified in the first CW, which were expected to solve certain problems with the experimental design, were not implemented (such as dataflow arrows in the background, and the problem design issues) because they had no practical solution or

Table 2. Comparison between the design of the 'before' and 'after' experiments

Variable	'After' version compared to 'before' version
Who: participants	
Participant count	Before: 61, After: 69.
Participant grouping	Before: WYSIWYT-No-Training (23), WYSIWYT-With-Training (21), Ad Hoc (17). After: WYSIWYT Group (39), Ad Hoc Group (30).
Participant pool	No change (computer science students in CS 381, CS 411, CS 511).
What: content and materials	
Tutorial	
Length	Before: about 25 min, After: about 20 min.
Content	Added explanations of new features and removed explanations about moving and resizing cells.
Examples	Two tutorial examples kept, one replaced.
Spreadsheet problems	
Grades spreadsheet	Boundary condition bug fixed.
Clock spreadsheet	Graphical clock output altered slightly because it could be perceived as incorrect.
Both problems	Cell in which participants entered reports of bugs they found was renamed from 'OutputErrors' to 'BugRecorder'.
Handout materials	
Forms/3 quick reference sheet	Name changed from 'Forms/3 Notes' to 'Forms/3 Quick Reference Card.' Notes regarding moving and resizing cells were removed, and notes regarding new testing features were added.
Problem descriptions	Added descriptions of expected inputs, error messages, and simplified the program descriptions.
Questionnaires	
Background	Added the question 'is English your native language?'
After first problem	Self-rating question scale was changed from ambiguous wording like 'very well' to an A–F grading scale. Question asking participants the length of time needed to complete the task was rephrased with more options.
After second problem	Added questions regarding the meaning of the user interface features in the Forms/3 environment, such as 'what does a blue arrow mean?' to get more accurate information. Rephrased questions about perceived usefulness of the methodology feedback.
Hardware and software	
Hardware	Same computers on the front end; backend was restructured so that there was less load on each server, meaning faster system response.

Table 2. (*continued*)

Variable	'After' version compared to 'before' version
Software	User interface changes as enumerated in Table 1.
When	
Time of day of experiment	No change (evening).
Where	
Location of experiment	No change (computer lab).

were deemed unnecessary for computer science students. The third walkthrough was intended to catch any problems that our changes might have introduced and to point out remaining unsolved problems.

Recalling difficulties doing the first walkthrough without having the VPL present, we performed the third walkthrough while actually performing each action on a computer running Forms/3, to be sure we had included all the options that would actually be available to the participants. This is not traditional in the use of CWs for user interfaces, since they are normally used at a stage of user interface design in which there is no executable user interface to use. However, it is viable in the design of an experiment involving an existing system, and we found that doing so added accuracy and completeness to our responses to the questions.

3.6. Experiment 1B: The 'After' WYSIWYT Study

In Experiment 1A, we chose to include the *WYSIWYT-With-Training Group* because we were not confident that participants using WYSIWYT could be successful without training in testing theory. The CW's emphasis on the user's prior knowledge made us confident that participants did not require such training, which allowed us to simplify the experimental design by abandoning the 'with training' condition. We could then make a simple comparison of WYSIWYT versus Ad Hoc, with equal numbers of participants in each group, instead of the more complicated ratio of 3:2 in the 'before' experiment.

After the encouraging results of the final CW, we proceeded to administer the redesigned experiment on new participants with two goals: to obtain clearer results about effectiveness of the WYSIWYT methodology, and to evaluate the benefits we had obtained through the use of the CW. Table 2 compares Experiments 1A and 1B.

3.7. How Can We Know Whether the CWs Made a Difference?

The aim of performing a CW on the experimental design was to increase the 'power' of the experiment, a term we shall shortly define. Although techniques exist for determining power they are not elementary, and our argument is that in the present context, it is adequate to use the extremely straightforward procedure of comparing patterns of

significance before and after the redesign inspired by CW, and reach conclusions purely by inspection. Here is why.

Experiments are run to help us make decisions based on the results we obtain. The fundamental task of experimental design is to increase the correctness of our decisions. That is, when populations genuinely differ, we want to detect the difference, or more accurately we want to reject the 'null hypothesis' of no difference; but when populations are identical, or do not differ to a degree that matters, we wish to conclude that we can observe no difference, or that 'we cannot reject the null hypothesis'. In hypothesis testing the probability of mistakenly concluding that a significant difference exists, denoted alpha, is customarily set at say 5%; this is a Type I error. The problem then is to decrease the probability of mistakenly failing to reject the null hypothesis (i.e. a Type II error). This probability is known as beta.

The power of a test is defined as 1 − beta. Thus, the smaller the Type II error probability, the greater the power of the test. One way to increase the power of a test is to collect more observations, and indeed there is a substantial literature on techniques to estimate the number of observations (or participants) required to achieve a desired level of power. The more scores that contribute to a statistic, the greater the precision of that statistic as an estimate of the population parameter. But collecting data costs time.

Alternatively, we can increase the power by eliminating sources of uncontrolled variation and thus reducing the estimate of error variance. That will increase the precision of the statistic, in the same way as collecting more scores, which will thereby increase the power. In algebraic terms, test statistics for the analysis of variance compute a term F which is the ratio of the variance between means of samples to the variance within samples:

$$F = MS_{\text{effect}}/MS_{\text{error}}$$

where the expected value of MS_{error} is the error variance, and the expected value of MS_{effect} is the error variance plus the treatment component. The smaller the error variance, the greater the value of F. The probability p of obtaining such a large value of F if indeed the null hypothesis were true is called the significance of the result. This is the probability that is compared to alpha.

The significance level cannot be used as a simple measure of power; nor can it be used as a measure of the relative treatment magnitude. Nevertheless, it is instructive to observe the effects of increasing the error variance. The following data are taken from [30, Tables 3 and 4 p. 60] (see Tables 3 and 4). Single factor analysis of variance (anova) returns $F(3,12) = 7.34$, $p < 0.005$, where p is the probability being compared to alpha of, say, 5%.

Now suppose that in that experiment, participants had had difficulty in following the instructions, in understanding the test problems, or in manipulating the screen controls, or in combinations of those. For illustrative purposes, we have added a random element of such 'noise' to each score, with a mean of zero and a standard deviation of approximately 16. The population treatment means are unchanged, of course, but the same single-factor anova now yields $F(3,12) = 3.35$, $p = 0.06$. The result has been transformed from one that was extremely significant to one that failed to reach the 5% level.

As we can see from this example, if all else is held constant, an increase in error variance will cause a decrease in significance, which in turn corresponds to a decrease in power, even though direct computation of that relationship is not easy. It is therefore essential to reduce sources of uncontrolled variation.

Table 3. Example data taken from [30]

Treatments			
A	B	C	D
37	36	43	76
22	45	75	66
22	47	66	43
25	23	46	62

Table 4. Same data as Table 3 but with additional random noise

Treatments			
A	B	C	D
36.55	16.37	48.94	82.54
35.90	70.44	76.03	68.04
21.60	63.61	81.94	46.42
22.48	18.70	48.31	61.39

How can we decide whether our efforts to reduce uncontrolled variation (e.g. by performing CW and redesigning as indicated) have been successful? Perhaps the best approach would be to compute estimates of the proportion of variance accounted for by treatment differences, using say the ω^2 index [30]. But to do so would introduce more technical apparatus than is really necessary to make a very simple point: if experimenters find ways to reduce uncontrolled variation, they will get better results, because of increased power. Complete enumeration of uncontrolled variation's removal, paired with inspection of the changes in results, is sufficient to demonstrate this.

Following this approach, Table 1 has already enumerated all the uncontrolled variations eliminated by the CWs we performed. The next section shows the changes in results.

3.8. A Comparison of Before and After

Table 5 presents a summary comparison of statistical analyses of the 'before' and 'after' WYSIWYT studies. In general, as the summary shows, whereas the 'before' study produced mixed results, the 'after' study supported all the major hypotheses with strong statistical results. Although there was one measure—the number of edits—which moved from significance in Experiment 1A to non-significance in Experiment 1B, this measure was dependent on the participants' relative success. That is, in Experiment 1A, the WYSIWYT groups achieved approximately the same coverage as did the Ad Hoc Group in significantly fewer edits (more coverage per edit), whereas in Experiment 1B, the WYSIWYT Group achieved more coverage than did the Ad Hoc Group in approximately the same number of edits (which is still more coverage per edit). Thus, this drop in significance does not imply a drop in efficiency.

Table 5. Major results from the 'before' and 'after' experiments

Hypothesis	'Before'	'After'
WYSIWYT participants more effective	NS	**
WYSIWYT participants more efficient:		
Number of edits to achieve coverage	**	NS
Number of redundant tests	NS	***
WYSIWYT participants less overconfident	NS	*
Training in testing theory not necessary to achieve better effectiveness and efficiency	NS	WYSIWYT Group was given no testing theory training, so results support this hypothesis

***Indicates $p < 0.001$; **, $p < 0.01$; *, $p < 0.05$. NS indicates $p > 0.1$. (No significance levels between 0.05 and 0.1 were found in the major hypothesis results.)

Bare comparisons of main-effect significance levels in Table 5 hide a great deal of information that further supports the differences in results. Those interested in details of the analyses are invited to see [31] for full details of Experiment 1B.

4. Experiment 2: A Forms/3 Modification Task

Our second use of the CW was to evaluate an experiment that was different from Experiment 1A/1B in two ways. First, we did not have a 'before' experiment to reflect on in order to improve the experimental design; thus it provides a different kind of information relevant to Research Question 1. Second, the participants in Experiment 2 would modify a spreadsheet in addition to testing it, complicating the participants' task and extending the range of evaluation that the CW would perform. This section briefly describes Experiment 2, and the impact of the CW upon it.

4.1. The Experiment

The experiment aimed to investigate the efficacy of the WYSIWYT testing methodology in the context of a spreadsheet modification task by end users [32]. The participants, students in their second or third year of a business degree, were divided into two conditions: one with WYSIWYT and one without. Each group completed a 25-min tutorial on Forms/3, which provided the necessary instruction to accomplish the modification task. Next the participants were given a written description of the spreadsheet, along with a list of modifications that were to be made. A questionnaire was given to test their understanding of the problem. Following this, participants were told that they had 15 min to complete the modification (nothing was mentioned about testing). If participants finished early, they worked on a dummy task. After the modification session, participants completed a questionnaire designed to test their understanding of the problem and their confidence of success.

4.2. A Cognitive Walkthrough of the Modification Task

While this experimental design was very similar to Experiments 1A and 1B, during the time between Experiments 1B and 2, a number of things had changed. The new team, consisting of seven evaluators, replaced the previous group of 10, keeping three original members, including the facilitator. The HCI expert was not present at any time. Furthermore, the group was finishing a complete reimplementation of the user interface for Forms/3 in Java. Moreover, components of the experiment were quite different. Our participant population was now end users instead of computer science students, the participants were expected to perform several actions not required of the previous experiment's participants, there was a different tutorial tailored to the new tasks, new post-modification questionnaires were created, and a different spreadsheet problem was used. Any of these differences could have introduced uncontrolled situational variables.

To guard against such variables, the group decided to perform a CW. Drafts of the materials that participants would encounter, including problem descriptions and tutorials were made available for the evaluators. This time, the HCI expert was not present to introduce the CW method, so the three continuing members introduced the method to the other group members. The original facilitator gave a 10-min introduction to the method describing motivations and preparing guidelines for the walkthrough procedure. Furthermore, the three evaluators who were familiar with the CW decided to follow rules similar to Spencer's to avoid design discussion detours and defensiveness [33].

The results of the walkthrough are summarized in Table 6. The CW revealed issues in each component of the experiment that had the potential to confuse participants. For example, the representation of the spreadsheet on the screen was inconsistent with the representation of the problem on the paper description, and the tutorial was inconsistent in places with the new interface. Furthermore, there were a number of interface variables, both regarding WYSIWYT and not, that required attention.

The changes were made and the experiment was run. The results for testing behavior were extremely clear-cut. The main results were that WYSIWYT participants were much more likely than the Ad Hoc participants to conduct at least some testing ($p = 0.0004$). Further, although testing did not help any Ad Hoc participants find even one error, three of the 19 WYSIWYT participants found errors during the process of testing. Taken together, these two findings suggest that WYSIWYT increases not only the raw quantity, but also the coverage of testing. These results are consistent with Experiment 1B.

Table 6. Frequencies of specific types of variables identified in Experiment 2's CW

Type of situational variable	Frequency
Problem design	3
Tutorial	6
Treatment user interface	4
Unrelated user interface	11

In addition, accuracy of the participants' modifications was considered. (Accuracy was not a factor in Experiment 1B.) The results were in the marginal significance range: WYSIWYT participants achieved better accuracy at a significance level of $p = 0.0861$.

We take this opportunity to emphasize that a CW cannot cause an experiment to produce significant results. Its relationship to achieving greater significance lies only in its ability to increase power by eliminating 'noise,' as already explained in Section 3.7.

5. The Cognitive Walkthrough of Experiment 3: A Comparison of Forms/3 Time Models

Like Experiment 2, Experiment 3 was designed without the help of a 'before' experiment. But unlike Experiment 2, it required participants to reason about a new model of time [34, 35].

Contrary to the structure of previous CWs, in the CW performed on this experiment there were only two evaluators conducting the CW: the original facilitator and the new lead experimenter (who had no experience with CWs). The CW they conducted identified only two variables; both were of the 'treatment user interface' type. It is possible, of course, that these were the only two variables that needed identification. Perhaps other problems had already been resolved in previous experiments and CWs; for example, the method for changing a cell's formula and the way our tutorials teach that skill had been walked through numerous times before. Also, the experiment was very simple, and there seemed to be few variables to worry about controlling. The hypothesis that there were no more variables to find is consistent with fact that the statistical analysis of the data ultimately collected in the experiment shows extremely clear results: The end-user participants using the new model of time were significantly more correct in the programming task ($p < 0.0001$), made fewer errors along the way ($p = 0.0002$), were faster in terms of both time spent ($p < 0.0001$) and number of edits ($p < 0.0001$), and demonstrated significantly greater understanding of the model of time than did participants using the traditional model of time ($p = 0.0003$).

Alternately, it could be that the team with only two evaluators was too small. With so few team members, the walkthrough proceeded much more quickly, and there were rarely disagreements between the two evaluators, unlike earlier CWs. Since these disagreements often led to important discoveries in earlier CWs, we may have overlooked uncontrolled variables of the experiment. Furthermore, since the CW was more a conversation between the two experimenters than a meeting, there was a tendency to slacken the rules and converge on agreement in the interest of time. This too could have reduced the effectiveness of the walkthrough to discover variables in the experimental design. To guard against these possibilities, we now always involve more than two evaluators.

6. Discussion

In this section we return to the research questions posed in the introduction.

6.1. Research Question 1: Is the CW Effective in Improving a VPL Experiment?

6.1.1. Hindsight and Foresight

Any experiment can be improved with hindsight. For example, in Experiments 1A and 1B, it is easy to think of reasons for the improvement from 'before' to 'after' in hindsight. Reducing the number of participant groups from 3 to 2 increased power-effectiveness of statistical tests. Improving the tutorial increased the likelihood that all participants understood what to do and how to do it. Improving details of the experimental interface increased the likelihood that participants would form correct goals, be able to achieve them, and be aware that they had made progress. All these can be obvious, after an experiment fails to produce clear results. However, it is not clarity of hindsight that is needed for better experimental designs, it is clarity of foresight. It is this improved clarity of foresight that is the benefit we gained from use of the CW.

6.1.2. Structured Evaluation

While an experiment can be designed in an ad hoc fashion, it can be aided by more structured design techniques. For example, in an ad hoc design process, a polite member of the team might not criticize an experiment's design. The CW provides an open session for criticism and evaluation, where individuals are expected to be critical and inventive. This was also observed by Spencer, who comments on social constraints of the CW in a software development environment [33].

6.1.3. Low-level Evaluation

Experiments involving human participants in fields such as psychology or sociology often must control situational variables such as typographical errors in questionnaires or a broken pencil. VPL experiments must be designed to control these variables *and* the rich user interfaces that can allow for countless unpredictable actions to be performed by participants. Thus, one critical way in which the CW is effective is in evaluating the lower-level aspects of an experiment: it is capable of helping experimenters find subtle problems which can be large sources of variation. For example, in the first line of Table 1 note that there was a question as to whether or not nested 'if' conditions may confuse participants. As computer scientists, we may not have noticed this potential problem. Yet, end-user participants might have been unable to complete the experimental task due to this kind of unnecessarily complex logic, thus making the collected data useless.

6.1.4. The CW and Issues Particular to Experiment Design

It is already known that CWs can reveal problems with an interface. Here we discuss the subtle differences between doing a CW to evaluate an interface versus doing one to evaluate an experiment, and how these subtle differences help to reveal problems in an experiment's design in a relatively cost-efficient way.

Problem Design: Traditional CWs pick actions of interest, but obviously cannot have a specific example that describes what *every* user is supposed to do. In contrast to this, the

experimental tasks are precisely that—specific goals under specific conditions that dictate the participants' goals and subgoals. This allows a CW for an experiment to be very specific and focused. Thus, many actions that would need to be evaluated in walking through an interface are not needed when evaluating an experiment. Because of this, a CW for an experiment can be faster than a CW for an interface.

Even so, the CW for an experiment considers problem-oriented data not considered by traditional CWs. For example, the problem design may introduce issues that will sidetrack the experiment's participants, such as layouts that are unintuitive, confusing spacing conventions, obtuse names, and so on.

Tutorial and Descriptive Materials: All CWs involve assumptions about what users have in the way of prior knowledge. In an experiment, much of this knowledge is obtained through the tutorial and supplemental materials (problem descriptions, quick reference cards, etc.) Doing a CW on an experiment focuses attention to how helpful the tutorial and materials are *specifically* to participants ability to perform the experimental tasks. The tutorial's impact is particularly important to consider throughout the CW, because the experiment rests upon its success. Both the completeness of the skills taught and the effectiveness of the way they are taught need to be considered throughout the CW's consideration of what skills participants have when performing the tasks. A tutorial that does not effectively teach the skills needed for the tasks dooms an experiment to producing useless data.

Evaluating the experimental tasks in the context of tutorials and experimental materials is in marked contrast to using a CW for interface design, in which there are usually no tutorial or materials at all to consider.

Interface Issues: Doing a CW on an interface can potentially reveal a large number of problems—too many in fact. That is, since most VPL researchers do not have the resources to improve their research prototypes to commercial standards, they cannot afford to follow up on a large list of improvements to their interface. However, doing a CW on an experiment focuses attention on precisely the subset of the interface likely to be encountered during the experimental tasks. This allows VPL researchers to concentrate their resources on resolving only those interface difficulties that could obscure experimental results.

For example, because Forms/3 is a research prototype, imperfections in its interface were expected. One such imperfection was that editing a cell formula was unintuitive and physically tricky. The team had already been aware of several such imperfections, but had considered them unimportant in designing a successful experiment. The CW dispelled this idea, and revealed which particular set of imperfections would directly impact the participants' ability to perform the experiment task.

In summary, the CW for an experiment is more focused than a CW for an interface. Regarding interfaces, this allows the VPL researcher to focus directly on the issues that will allow the experiment to produce clear results, without having to produce an interface up to commercial quality. It also takes into account the specific tasks and specific training materials to be used, which are not considered in CWs aimed at interface evaluation. The tasks and training are as important to the experiment's outcome as the interface itself, and the experiment's CW keeps the importance of these factors continually in focus.

6.2. Research Question 2: Does the CW Possess Advantages Over More Traditional Approaches to Improving VPL Experiments?

There are two traditional approaches to experiment design evaluation that are commonly used: pilot studies and protocol analyses. To discuss Research Question 2, these two methods will be discussed briefly and then compared to the CW.

6.2.1. Pilot Studies

Pilot studies are studies with fewer participants. The goal of a pilot is to determine if there are any glaring flaws in the experimental design. For example, a pilot may help experimenters determine if the experimental task is too simple or too difficult, thereby avoiding floor and ceiling effects in data collection. This information can be very helpful in identifying large problems with an experimental design.

Unfortunately, to perform a pilot, an experimenter must have the majority of the experiment designed and the materials, such as tutorial and questionnaires nearly complete. If large problems are found in the design, it can be very expensive to redesign these portions of the experiment. However, an advantage of the pilot is that it involves real humans, real data collection, and so on; thus it is very useful as a final test before running the real experiment.

6.2.2. Protocol Analyses

The goal of a protocol analysis is to observe a participant while performing a task, and use their behavior as data to analyze. Participants usually 'think aloud' and experimenters can use the data to infer rules and properties that reflect the participant's problem-solving process. At one level, this can be used by experimenters to validate assumptions about participants' reasoning processes for the experimental task. For example, experimenters may assume that a participant will make a certain modification first; a protocol analysis may show that in fact, participants do this modification last. At another level, it simply allows experimenters to learn about any aspects of the experimental procedure that they have not anticipated.

Of course, since it is similar to a pilot study, the majority of the experiment must be designed before executing the analysis. Furthermore, since the participants are encouraged to speak at any time, protocol analyses can be very time consuming. Thus, fewer participants can be tested, usually only one at a time.

6.2.3. Advantages (and Disadvantages)

In essence, pilot studies and protocol analyses are too time consuming and expensive to do repeatedly. The CW, on the other hand, is inexpensive enough to do multiple times. Further, because it does not require an almost finished experiment, it can be done early. To be more specific, we observed several advantages of evaluating experiments with the CW:

- It was relatively quick; with a small group of evaluators, the longest walkthrough spanned a total of about 4 hr. This is much faster than preparing an 'almost polished' version of the experiment, as in a pilot or protocol analysis.

- It focused on possible problems with each specific subtask, which led to a list of specific design issues. In contrast, a pilot does not provide this level of subtask detail.
- The wording of the CW's questions, while not prescriptive, was relatively constructive, pointing out where better information would help participants to perform their task. For example, in Figure 4, our response to question 3.3 suggested that user interface and tutorial changes could solve the problem.
- It was simple to see how a number of aspects of the experimental design remained constant, and thus did not require further evaluation until changed. For example, the user interface elements to edit a cell's formula remained the same throughout Experiments 2 and 3, and we could save time by avoiding evaluation of this element for Experiment 3.
- Although the first CW was a walkthrough of the entire experimental task, it is also possible to use CWs to analyze some portion of an experiment at a lower resolution. This is seen by comparing the scope of Experiment 1A and 1B's three CWs.

Why might the CW possess these advantages? All three experimental designs required participants, supported by the VPL interface display, to explore, reason and act. Understanding such behavior is the strong point in the CWs evaluative abilities. This may explain why the technique was so useful in this type of VPL experiment. For experiments not so reliant on the ability to support reasoning skills about programs, some other evaluative technique might be needed.

The CW method has some limitations. For example, although the CW does focus on the individual user steps, it has little to say about the cost of making and repairing an error. Other issues that the method does not address include whether the user is compelled to look ahead before choosing an action or whether changing one value in the system is likely to require further changes to restore an internally consistent state, neither of which was an issue in our experiments.

Thus, although the CW possesses certain advantages over the other methods, it is not a panacea. Our results can be taken as strong support within a circumscribed area, but the exact definition of its boundaries requires further study. As a result of our experiences, the methodology now used by our group is to conduct one or more CWs for every experiment as early as possible, with a pilot near the final experiment. If questions arise from the CWs that the team cannot answer, we also sometimes use a protocol analysis sometime after the first CW and before the pilot.

6.3. Research Question 3: Is an HCI Specialist Needed for Effective Use?

Over the course of three experiences using the CW, the level of training and the results of the walkthroughs give reasonable evidence for Research Question 3. As stated before, our group received very little training on the CW. In Experiments 1A and 1B, our team consisted of computer scientists with no prior experience using the CW, with the exception of the HCI expert. The HCI expert taught the team the method in about 90 min and the rest of the team did the walkthroughs of the experiment. In Experiment 2, only three of seven group members had experience from the preceding CWs; in this case, the undergraduate facilitator gave a brief ten-minute training period. In Experiment 3, the facilitator gave a brief 5-min training period to the other evaluator without the help of the HCI expert.

With these levels of training, a number of variables were identified, such as shown in Tables 1 and 6. Thus, although a team of HCI experts might have extracted more from the walkthroughs, our team certainly used it successfully enough to obtain useful results.

6.4. Research Question 4: What General Lessons Can Be Learned About Experiment Design?

Other domains of experimentation have guidelines regarding variable types, to help experimenters to design a successful experiment. The VPL domain could likewise benefit from guidelines that take into account variables commonly encountered in VPL experiments. In our experiences with the CW, we noticed four such situational variable types (given in Section 3.3) and propose these four types as a start at such a guideline list for VPL experimenters:

- Problem Design. Design the problems given to participants so that their attention remains on the experimental task, rather than on extraneous elements such as syntax oddities or nested logic.
- Tutorial Design. Design the tutorial with close scrutiny to participants' ability to master the material, considering not only content, but also length and practice time.
- Treatment-Related Interface. If there is a user interface component *in* the feature under evaluation, make sure it has helpful action choices and labeling.
- Unrelated Interface. If there is a user interface component present in the experiment *unrelated* to the feature under evaluation but still likely to be used by participants, eliminate distracters that could impact ability to perform the experiment's task.

We offer this experimental design checklist to other VPL experimenters, and welcome additions.

7. Conclusion

Like John and Packer, we found that the CW could be used successfully by a computer science team, even without the presence of an HCI expert or cognitive scientist [25]. In our case, we did learn the technique via a short training session from such a person, but he did not participate in the walkthroughs *per se*.

In our experiences, use of the CW improved the design of VPL-focused empirical studies. We believe it may also be useful for other types of empirical studies that focus on problem-solving situations supported by interfaces, but we can only speculate about this point, since we have not tried it outside the domain of VPL experiments.

For improving VPL experiments, we found the CW to be *effective*: it exposed uncontrolled variables and prompted us to think of solutions. It was *easy to learn*, needing just one training session with an HCI expert. It was *quick*, taking about 4 hr for the longest walkthrough. It was *concretely focused*, requiring very little working out how an abstract theory or model could be applied to the specific situation. Finally, it was *constructive*, leading directly to suggestions for improving the experiment design.

Acknowledgements

We thank Miguel Arredondo-Castro, John Atwood, Josh Cantrell, Mingming Cao, Nanyu Cao, Dan Keller, Vijay Krishna, Dusty Reichwein, Justin Schonfeld, and Andrei Sheretov, for their contributions to the CWs and to the associated experiments. This work has been supported in part by the National Science Foundation under awards CCR-9806821 and ITR-002265.

References

1. A. Blackwell (1996) Metacognitive theories of visual programming: what do we think we are doing? *IEEE Symposium on Visual Languages*, Boulder, CO, September 1996, pp. 240–246.
2. J. Gurka & W. Citrin. Testing effectiveness of algorithm animation (1996) *IEEE Symposium Visual Languages*, Boulder, CO, September 1996, pp. 182–189.
3. P. Polson, C. Lewis, J. Rieman & C. Wharton (1992) Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man–Machine Studies* **36,** 741–773.
4. C. Lewis, P. Polson, C. Wharton & J. Rieman (1990) Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. *ACM CHI '90 Conference on Human Factors in Computing Systems*, Seattle, WA, April 1990.
5. J. Rieman, M. Franzke & D. Redmiles (1995) Usability evaluation with the cognitive walkthrough. *ACM CHI '95 Conference on Human Factors in Computing Systems*, Denver, CO, May 1995, pp. 387–388.
6. A. Sears & D. Hess (1998) The effect of task description detail on evaluator performance with cognitive walkthroughs. *ACM CHI '98 Conference on Human Factors in Computing Systems*, Los Angeles, CA, April 1998.
7. B. Bell, W. Citrin, C. Lewis, J. Rieman, R. Weaver, N. Wilde, & B. Zorn (1994) Using the programming walkthrough to aid in programming language design. *Software Practice and Experience* **24,** 1–25.
8. C. Hundhausen & S. Douglas (2000) Using visualizations to learn algorithms: should students construct their own, or view an expert's? *IEEE Symposium on Visual Languages*, Seattle, WA, September 2000, pp. 21–28.
9. C. Lewis, C. Brand, G. Cherry & C. Rader (1998) Adapting user interface design methods to the design of educational activities. *ACM CHI '98 Conference on Human Factors in Computing Systems*, Los Angeles, CA, April 1998, pp. 619–626.
10. J. Stasko, A. Badre & C. Lewis (1993) Do algorithm animations assist learning? an empirical study and analysis. *ACM INTERCHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, pp. 61–66.
11. T. Green (1989) Cognitive dimensions of notations. In: *People and Computers* (A. Sutcliffe & L. Macaulay, eds), Vol. V. Cambridge University Press, Cambridge, pp. 443–460.
12. T. Green, M. Burnett, A. Ko, K. Rothermel, C. Cook & J. Schonfeld (2000) Using the cognitive walkthrough to improve the design of a visual programming experiment. *IEEE Symposium on Visual Languages*, Seattle, WA, September 2000, pp. 172–179.
13. T. Green & M. Petre (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages and Computing* **7,** 131–174.
14. S. Card, T. Moran & A. Newell (1983) *The Psychology of Human-Computer Interaction*. Erlbaum, Hillsdale, NJ.
15. M. Williams & J. Buehler (1999) Comparison of visual and textual languages via task modeling. *International Journal of Human–Computer Studies* **51,** 89–115.
16. R. Jeffries, J. R. Miller, C. Wharton & K. M. Uyeda (1991) User interface evaluation in the real world: a comparison of four techniques. *ACM CHI '91 Conference on Human Factors in Computing Systems*, New Orleans, pp. 119–124.

17. C. Karat, R. Campbell & T. Fiegel (1992) Comparison of empirical testing and walkthrough methods in user interface evaluation. *ACM CHI '92 Conference on Human Factors in Computing Systems*, Monterey, CA, 1992, pp. 397–404.
18. W. Gray & M. Salzman. Damaged Merchandise? A review of experiments that compare usability evaluation methods (1998) *Human-Computer Interaction* **13,** 203–261.
19. B. Bell, J. Rieman & C. Lewis (1991) Usability testing of a graphical programming system: things we missed in a programming walkthrough. *ACM CHI* '91 *Conference on Human Factors in Computing Systems*, April 1991, pp. 7–12.
20. J. Nielsen & R. Molich (1990) Heuristic evaluation of user interfaces. *ACM CHI '90 Conference on Human Factors in Computing Systems*, Seattle, WA, April 1990, pp. 249–256.
21. S. Card, T. Moran & A. Newell (1980) The keystroke-level model for user performance time with interactive systems. *Communications of the ACM* **23,** 396–410.
22. H. Johnson & P. Johnson (1992) Task knowledge structures: psychological basis and integration into system design. In: *Cognitive Ergonomics: Contributions from Experimental Psychology*, (G. van der Veer, S. Bagnara & G. Kempen, eds) North-Holland, Amsterdam, pp. 3–26.
23. R. Butterworth, A. Blandford & D. Duke (1999) Using formal models to explore display based usability issues. *Journal of Visual Languages and Computing* **10,** 455–479.
24. C. Wharton, J. Bradford, R. Jeffries & M. Franzke (1992) Applying cognitive walkthroughs to more complex user interfaces: experiences, issues, and recommendations. *ACM CHI '92 Conference on Human Factors in Computing Systems*, Monterey, CA, pp. 381–388.
25. B. John & H. Packer (1995) Learning and using the cognitive walkthrough method: a case study approach. *ACM CHI '95 Conference on Human Factors in Computing Systems*, Denver, CO, May 1995, pp. 429–436.
26. G. Rothermel, M. Burnett, L. Li, C. DuPuis & A. Sheretov (2001) A methodology for testing spreadsheets. *ACM Transactions on Software Engineering and Methodology* **10,** 110–147.
27. G. Rothermel, L. Li, C. DuPuis & M. Burnett (1998) What you see is what you test: a methodology for testing form-based visual programs. *20th International Conference on Software Engineering*, Kyoto, Japan, April 1998, pp. 198–297.
28. M. Burnett, J. Atwood, R. Djang, H. Gottfried, J. Reichwein & S. Yang (2001) Forms/3: a first-order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming* **11,** 155–206.
29. M. Burnett & H. Gottfried (1998) Graphical definitions: expanding spreadsheet languages through direct manipulation and gestures. *ACM Transactions on Computer-Human Interaction* **5,** 1–33.
30. G. Keppel (1982) *Design and Analysis: A Researcher's Handbook*, 2nd edn. Prentice-Hall, Englewood Cliffs, NJ.
31. K. J. Rothermel, C. Cook, M. Burnett, J. Schonfeld, T. R. G. Green & G. Rothermel (2000) WYSIWYT testing in the spreadsheet paradigm: an empirical evaluation. *International Conference on Software Engineering*, Limerick, Ireland, June 2000, pp. 230–239.
32. V. Krishna, C. Cook, D. Keller, J. Cantrell, C. Wallace, M. Burnett & G. Rothermel (2001) Incorporating incremental validation and impact analysis into spreadsheet maintenance: an empirical study. *IEEE International Conference on Software Maintenance*, Florence, Italy, November 2001, pp. 72–81.
33. R. Spencer (2000) The streamlined cognitive walkthrough method, working around social constraints encountered in a software development company. *ACM CHI '00 Conference on Human Factors in Computing Systems*, The Hague, Amsterdam, April 2000.
34. M. Burnett, N. Cao, M. Arredondo-Castro & J. Atwood (2001) End-user programming of time as an 'ordinary' dimension in grid-oriented visual programming languages, TR 01-60-01, Oregon State University, Jan. 2001. *Journal of Visual Languages and Computing* **13,** 421–447.
35. M. Burnett, N. Cao & J. Atwood (2000) Time in grid-oriented VPLs: just another dimension? *IEEE Symposium on Visual Languages*, Seattle, WA, September 2000, pp. 137–144.
36. C. Wharton, J. Rieman, C. Lewis & P. Polson (1994) The cognitive walkthrough method: a practitioner's guide. In: (J. Nielsen & R. L. Mack, eds), *Usability Inspection Methods*. John Wiley, NY.

Appendix A: A Detailed Example of a Cognitive Walkthrough

The following details the portion of the CW on action (5), described in Section 3.3. At this point in the CW, Martha, the hypothetical experiment participant, had changed the formula of an input cell and was ready to click the checkbox of the output cell (recall Figure 1). With the form in Figure A1 in front of them, the group proceeded with the first question.

In our CWs, Martha is a 'good' participant, who always forms good goals in answer to 1.1. Question 1.2 then asks whether it is reasonable to expect all participants to have this goal. Quickly answering question 1.1, the current goal was to check the checkbox of the output cell. Reaching question 1.2, the group wanted to stop and discuss. Would the experiment participants actually have this goal, or would they just observe the output, say okay in their heads, and proceed to change another input cell without clicking? Since the experiment intended to discover if the WYSIWYT methodology resulted in better testing, it was difficult to find a solution that did not *tell* the participants to test more. This was a serious issue, so the group recorded the issue for further discussion in a later meeting.

After completing the first set of questions, the facilitator proceeded with the second set, shown in Figure A2. The correct action to take was to check the checkbox of the output cell 'LetterGrade', but at question 2.2 the group was unsure if the empty checkbox was a good enough label to prompt an experiment participant to check it. Would the participants even know that the non-traditional looking checkbox (seen in Figure 1) was a checkbox at all? Instead of solving the problem at this point, the group recorded the issue, and proceeded to answer the successive questions.

After recording the group's answer to 2.9, the facilitator proceeded to the last group of questions, and reminded the group that at this point in the CW, the hypothetical experiment participant had correctly checked the output cell's checkbox. Reaching 3.3, the group was reminded that the hypothetical experiment participant still had the goal of testing the spreadsheet, and not just clicking this checkbox. Would the participants in the experiment need some sort of reminder of their progress to make sure they continued testing? The group thought so, and recorded a group member's suggestion for a 'testedness indicator' which would remind participants of their progress in testing the spreadsheet. See Figure A3.

Cognitive Walkthrough Form: A Single Step
Task: *Check off the final output box on the Grades spreadsheet*
Action #: *5*

1. Goal Structure for this step

1.1 Correct Goals. What are the current goals for this point in the interaction?
The current goal is to check the checkbox of the output cell.

1.2 Mismatch in likely goals Check each goal in this structure against your analysis at the end of the previous step. Will all users have dropped it or failed to form it? Are there unwanted goals that will be formed or retained by some users?
There's a possibility that some users may not have this goal after changing the formula. They might just look at the value, say okay, and proceed to try another input value. How much do we really want to emphasize the importance of testing in the tutorial?

Figure A1. The first set of questions and answers for action (5). Italicized words indicate evaluator responses and underlined phrases indicate responses that were emphasized in discussion

2. Choosing and executing the action.
What is the correct action at this step?
check the final output cell "final grade"

2.1 Availability. Is it obvious that the correct action is a choice here?
There's nothing obvious about the checkbox. Our anticipated users should, however, understand the concept of the checkbox (which is fairly common in user interfaces these days) and realize that the box in the upper right of each cell is exactly that. Furthermore, the correct action lies on a continuum: if they click on a cell in the middle of the flow of data, it's not as great as if they'd clicked on the final output cell. Does that matter though, as long as they make progress?

2.2 Label. What label or description is associated with the correct action?
There is an empty box, which unfortunately doesn't indicate much: possibly no label?

2.3 Link of label to action. If there is a label or description associated with the correct action is it obvious, and is there a good match between the label and the user's representation?
A checkbox is pretty close to the action of "checking," we'd say.

2.4 Link of label to goal. If there is a label or description associated with the correct action, how it obviously connected with one of the current goals for this step?
There's an empty checkbox: does that scream out "test me!"?

2.5 No Label. If there is no label associated with the correct actions, how will users relate to this action to the current goal?
The users will have the oral tutorial and the Forms/3 quick reference sheet. The checkbox is also a common user interface concept. A fairly small percentage will have a problem if the tutorial is good enough. Maybe we should have a better indicator to ensure this.

2.6 Wrong Choices. Are there other actions that might seem appropriate to some current goal?
There are a lot. The user could change other formulas, validate different cells, and lots of other things. The interface allows almost anything when it comes to testing. How will we get the participants to validate the useful cells?

2.7 Too many Choices. Are there more than 10 screen actions to be considered at this time?
There are too many to count!

2.8 Time-out. If there is a time-out in the interface at this step does it allow time for the user to select the appropriate action?
Yes; the participants have the freedom of the mouse cursor.

2.9 Hard to do. Is there anything physically tricky about executing the action?
The checkbox is actually pretty small, but nobody in this group seems to have a problem with it. Would participants?

Figure A2. The second set of questions and answers for action (5)

3. Modification of Goal Structure, Assuming the Correct Action has Been Taken
3.1 Quit or Backup. Will users see that they have made progress? What will indicate this?
The participants will see a change in cell border colors, a checkmark will appear, and arrow colors, if displayed, will also change.

3.2 Accomplished Goals. List all current goals that have been accomplished. Is it obvious that each has been accomplished?
The current goal of "check of the output cell" has been accomplished.

3.3 Incomplete Goals That Look Accomplished. Are there any current goals that have not been accomplished, but might appear to have been?
Since another goal is to test the whole spreadsheet, but the participant has received feedback, they may think that they've tested enough. Do we need some sort of indicator?

3.4 "And-Then" Structures. Is there an "And-Then" structure and does one of its sub-goals appear to be complete? How many users may prematurely terminate?
No, they just have to click.

3.5 New goals in response to prompts. Does the system response contain a prompt or cue that suggests any new goals?
If at this point the participant has the cells on the spreadsheet anything less than all blue, they will likely be cued to test more.

3.6 Other new goals. Are there any other new goals that the user will form given their current goals, the state of the interface, and their background knowledge?
We can't think of any.

Figure A3. The third and final set of questions for action (5)

Appendix B: Getting Started

This appendix provides advice on how to get started using CWs in experiment design.

1. Learn the CW technique. We recommend [36] as a very good source for how to use the technique.

2. Assemble details of the experiment: the tutorial materials available, the experiment tasks, any materials to be provided to the participants, relevant screen shots, etc. It saves time and adds accuracy to have a live system available instead of static screen shots, but a live system is not required.
3. Appoint a facilitator, whose job is to lead the CW.
4. Appoint a recorder, whose job is to record the comments and results. (The facilitator will not have time to do this.)
5. If the group has never done CWs before, conduct a practice session on some tiny aspect of the experiment. The examples in this paper may be useful guidance; in particular, the detailed example of Appendix A is precisely for the purpose of providing a concrete start-to-finish example of using the CW in an experiment.
6. Dealing with procedural problems: The practice session may reveal group tendencies to drift off track, and other group dynamics that might be seen as counterproductive. Spencer offers good advice for avoiding and dealing with such matters [33] if they arise.