IEEE Symposium on Visual Languages, Seattle, ashington, Sept. 2001

Using the Cognitive Walkthrough to Improve the Design of a Visual Programming Experiment

T. R. G. Green[†], M. M. Burnett[*], A. J. Ko[*], K. J. Rothermel[*], C. R. Cook[*], and J. Schonfeld[*]
[†]*Computer-Based Learning Unit*
University of Leeds, Leeds LS2 9JT, U.K.
[*]*Department of Computer Science*
Oregon State University, Corvallis, OR 97331

Abstract

Visual programming languages aim to promote usability, but are rarely examined for it. One reason is the difficulty of designing successful experimental evaluations. We propose the Cognitive Walkthrough as an aid to improve experimental designs. This is a novel application of an HCI-derived technique designed for evaluating interfaces rather than experiments. The technique focuses on the potential difficulties of novice users and is therefore particularly suited for evaluating the programming situation, which is knowledge-based and non-routine. We describe an empirical study performed without benefit of a walkthrough and show how the study was improved by a series of walkthroughs. 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 [2]. Yet, until recently it has been regrettably rare to see any but the most cursory test of whether a VPL is usable. One reason may be the difficulty of designing well-controlled experiments. This paper describes an approach to helping with this task.

Designing a valid usability experiment is difficult, not so much because advanced knowledge of experimental design is needed, but because the results can easily be disturbed by extraneous problems. For example, the participants might not understand the instructions, or they might not be able to 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.

Essentially, for an empirical evaluation to be effective, everything must work smoothly. The participants must have very few problems in understanding their task and in using the VPL; otherwise the results will be contaminated by random delays or errors. Technically, such random perturbations come under the heading of *uncontrolled sources of variation*.

The scale and pervasiveness of the problem is shown by Gurka and Citrin's review [9] of empirical studies of the effectiveness of algorithm animation. They list six variables that are often uncontrolled. Although these variables are specific to algorithm animation, such as animation quality and the difficulty of the algorithms being animated, the implication is clear. Designing valid experiments involving humans is difficult.

How can designers of experiments detect potential problems at an early stage, before performing the actual experiments? 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 [12] as a means to reduce uncontrolled variables in experimental designs.

The Cognitive Walkthrough (CW) is familiar in HCI as a tool to improve interface usability. But improving an *evaluation* is not the same as improving an *interface*. Interface details are only part of the issue, as we shall see. Use of the CW for improving an evaluation has not hitherto been reported in the literature.

We report a case study in which, following a VPL evaluation experiment with mediocre results, a CW was used to refine the experiment, and the experiment was then re-run. Questions of interest are whether the CW possesses advantages over more traditional approaches to improving experiments; whether it was effective in improving the experiment; whether specialist training was required; and what general lessons could be learnt.

2. The Cognitive Walkthrough

2.1 Background: the walkthrough process

The Cognitive Walkthrough [12] was primarily designed as a 'desktop' evaluation tool for usability engineering, aimed at predicting potential difficulties for novice users. Its strengths are that it does not require a functioning model of the product or a group of users for extensive testing, 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 the feedback indicates that progress is being made towards the goal.

First, the evaluation team must define the expected users and estimate their prior knowledge (because users who come with knowledge of many related or similar interfaces will have fewer problems than those who come with little knowledge). Next the team must prepare 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 pre-set questions relevant to the four phases of activity:

- 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.

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 [13, 16].

2.2 Relevance to empirical studies of VPLs

Surveying the development of evaluation methods in HCI in general, it is clear that the field has not yet settled down, making difficult the choice of an evaluation method for experimental designs. Unfortunately, attempts to compare the efficacy of evaluation methods were partially vitiated by weaknesses in the design of the comparisons [6]. The choice must therefore be determined by apparent suitability, not by demonstrated superiority. The CW appears particularly suitable for evaluating a VPL experiment for two reasons.

2.2.1 Focus on 'finding the way'. VPLs include user interfaces; the CW method focuses on users who are new to a given interface and who are trying to find out how to use it. It does not focus on speed and accuracy, but on reasoning. Programming tasks are non-routine and require reasoning. These factors make it particularly suitable for evaluations of people's ability to perform programming tasks in a VPL. Further, the creators of the CW approach have previously successfully applied it to a VPL [1].

Other evaluative methods derived from HCI typically operate at a more surface level, paying less attention to reasoning and knowledge. Heuristic Evaluation [11], 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 [5] is good for predicting how fast a user can perform a well-learnt 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. Once again, that is in contrast to many HCI approaches, which explicitly assume HCI or cognitive science expertise. (Heuristic Evaluation is an exception, but as we have seen, that approach is not relevant to our needs.)

However, evidence for this potential is scanty and mixed. On one hand Wharton, Bradford, and Franzke [17], after evaluating three user interfaces, identified a few important weaknesses of the 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." However, John and Packer [10] 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." Their single-user case study is encouraging but can hardly be considered a definitive demonstration, and further evidence is required, which we aim to supply.

3. The WYSIWYT feature being evaluated

The experiments described below evaluate a visual testing methodology, applicable to commercial spreadsheet systems as well as some VPLs [14]. 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 is whether users can benefit from employing this methodology.

Our experiments used the research VPL Forms/3 [3, 4]. In Forms/3, ordinary formulas, contained in movable cells, can be used for both numeric and graphical computations. Figure 1 contains a spreadsheet, one of the two used in the experiments. The WYSIWYT testing methodology can be seen in several of the features of Figure 1. Red cell borders (shown as light gray in this

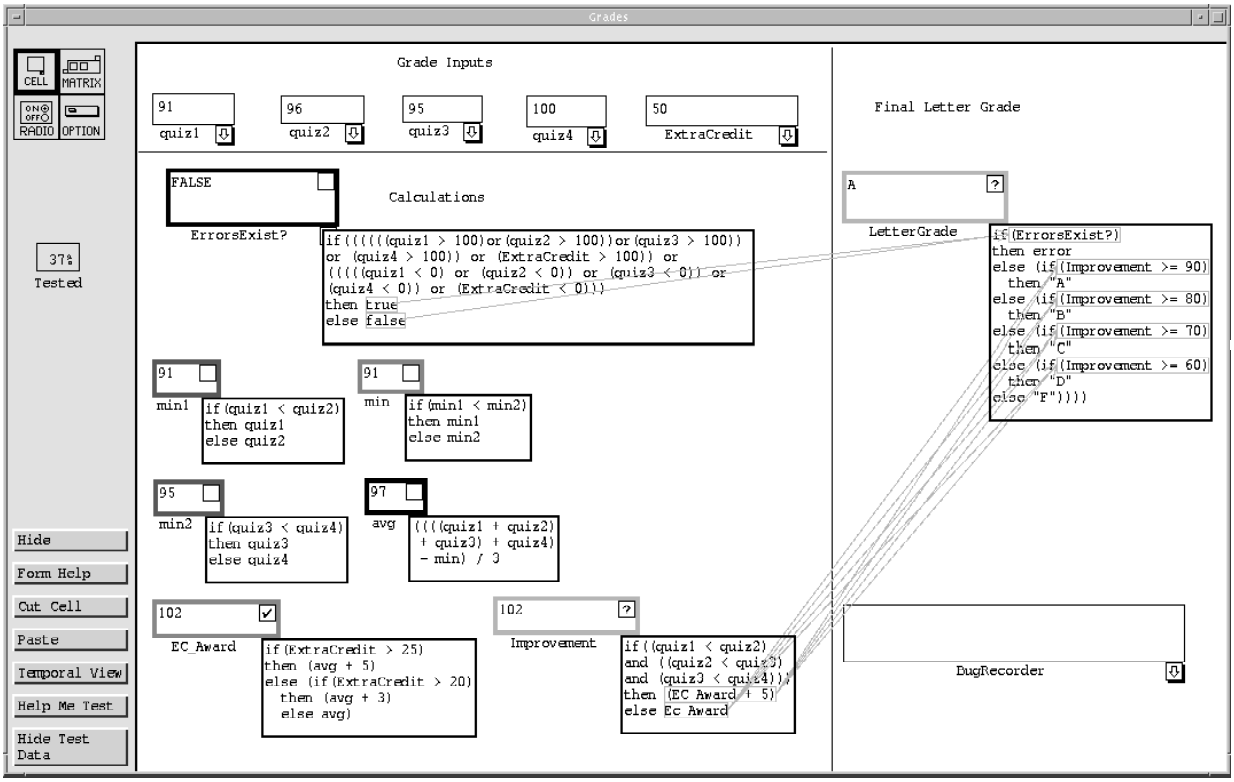

Figure 1: A Forms/3 spreadsheet that calculates a student's grade. Cell relation arrows in and out of formulas are colored in the same way borders are colored: from red (untested) to blue (tested), light gray to black in this paper.

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 purples indicate partial "testedness". Users record decisions that values are correct by checking off the checkboxes in the upper right corner of cells (some currently containing ?'s in the figure). Users can also invoke other visual devices, such as dataflow arrows (between subexpressions or entire cells) colored in the "testedness" colors.

4. Before and after the walkthroughs

Our first experiment with the WYSIWYT methodology gave mediocre results. In this section we describe the first experiment, discuss our use of the CW to refine it, then describe the refined experiment.

4.1 Before: the first WYSIWYT study

The aim of this 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 treatment groups. 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 access to the WYSIWYT methodology. The experiment started with a 25-minute 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 counter-balanced.

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.

4.2 Three Cognitive Walkthroughs

We used the process described in Section 2.1 to perform three walkthroughs of the experimental procedure. In the first walkthrough, looking ahead to future studies, we chose our anticipated user population to be end users, even though our actual participants would initially be computer science students (who presumably have greater background knowledge). A portion of this walkthrough is shown in Figure 2 and Figure 3.

The walkthrough revealed four types of experimental design issues (see also Table 1):

- 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, introducing uncontrolled variables.
- 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 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. Since the redesigned study would first be administered on computer science students, who are familiar with such complexities, Problem Design changes were not made. However, the end user study for which we are now preparing makes these changes a priority.

A second walkthrough was subsequently performed, to test an interface detail's redesign that resulted from the first walkthrough. This second walkthrough addressed solely that one issue (the undo feature in Table 1), but it took into account a range of possible testing strategies that participants might adopt (note the divergence from the typical HCI usage, which focuses on a single method for a single task).

After implementing the revisions in Table 1, we performed a third walkthrough. The revisions to the experiment design could have themselves introduced new variables. For example, during each of the previous Walkthroughs, a number of questions on the Walkthrough forms were answered positively under the assumption that the user would retain all of the knowledge

Cognitive Walkthrough Form: Start up sheet.

Experiment: *WYSIWYT experiment*
Task: V*alidate 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 spread 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. Italicized text represents a summary of the notes the evaluators took during the walkthrough.

Cognitive Walkthrough Form: A Single Step

Task: *Validate output of a set of inputs on the Grades spreadsheet*
Action #: *5*

…

2. Choosing and executing the action.

2.1 Availability. Is it obvious that the correct action is a choice here? If not, what percentage of users might miss it? (0, 25, 50, 75, 100)

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?

…

2.5 No Label. If there is no label associated with the correct actions, how will users relate to this action to the current goal? What percentage of users might have trouble with this? (0, 25, 50, 75, 100)

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.

Figure 3: Excerpts from our first walkthrough, showing the accumulation of ideas resulting in the last sentence.

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.	Either 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.*
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.*

Table 1: Specific issues identified during our first walkthrough. Italicized text represents changes that were made as a result. Solutions marked with an asterisk (*) indicate changes that were discussed before the CW was done, but were also revealed by the CW.

contained within the tutorial. Yet, we had since made changes to the tutorial. Furthermore, some of the changes identified in the first walkthrough, 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 either had no practical solution, or were not considered necessary for the audience of computer science students. The third walkthrough was intended to catch any problems that our changes might have introduced.

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.

4.3 After: the redesigned WYSIWYT study

We administered the redesigned study 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 the original and refined ("before" and "after") experiments.

5. Results

Table 3 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 our major hypotheses with strong statistical results. Although there was one measure—the number of edits—that moved from significance "before" to non-significance "after," this measure was dependent on the

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 minutes, After: about 20 minutes.
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 tools 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.
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).

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

participants' relative success. That is, in the "before" experiment, the WYSIWYT groups achieved approximately the same coverage as did the Ad Hoc Group in significantly fewer edits (more coverage per edit), whereas in the "after" experiment, 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.

Bare comparisons of main-effect significance levels in Table 3 hide a great deal of information that further supports the differences in results; for instance, in the first row, the power computed for the test of the simple main effect, WYSIWYT versus Ad Hoc, was improved from 0.11 to 0.84. Space forbids extensive statistical discussion here; details of the "after" version are given elsewhere [15].

6. Discussion

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

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

Table 3: Major results from the "before" and "after" experiments. *** 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.)

6.1 CWs in the domain of VPL experiments

Our experimental design required participants, prompted by the VPL interface display, to explore and reason. Understanding such behavior is the strong point in the CW's evaluative abilities. This may explain why the technique was so useful in this type of VPL experiment.

Still, 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 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 were issues in our experiment. Designers of VPL experiments for which these are potential issues might need to supplement the CW with other evaluative techniques, such as Green's Cognitive Dimensions [7-8].

Thus, while the results of our case study can be taken as strong support within a circumscribed area, the exact definition of its boundaries require further study. Also, outside that circumscribed area, other techniques may and probably will be needed.

6.2 Advantages

We observed several advantages of evaluating the experiment with the CW, as opposed to pilot studies and protocol analyses, which are other experiment evaluation mechanisms we have used.

- It was relatively quick; with a small group of evaluators, the walkthrough spanned a total of about four hours. This is much faster than preparing an "almost polished" version of the experiment, as is required for 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 3, question 2.5 suggested either user interface or tutorial changes.

However, the CW should not be viewed as a replacement for other VPL experiment design devices, but rather as an additional tool. For example, we have recently used both a pilot and "think aloud" analyses to help prepare for our upcoming end-user version of the experiment.

6.3 Effectiveness

Any experiment can be improved with hindsight. Our observations with the CW suggest that it moves the hindsight to where it belongs—before the experiment.

In hindsight, it is easy to think of reasons for the improvement from "before" to "after". 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 were obvious—afterwards. 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.

It would be natural to ask why the experimental deficiencies relating to the user interface deficiencies had not been noticed before the experiment. In fact, the team had already been aware that there were imperfections in the user interface. However, the CW helped ascribe importance to user interface problems that the team had noticed but had considered unimportant in designing a successful experiment. For example, the team had always realized that editing a cell formula was unintuitive and physically tricky. Because Forms/3 is a research prototype, these types of imperfections were expected, and presumed unrelated to a successful experiment design. The walkthrough dispelled this idea by illustrating their impact on the experiment task.

The improvement of our WYSIWYT experiment from the "before" version to the "after" version was not limited to sharpening the statistical significance levels; we also gained a better understanding of what we really wanted to know and how to obtain the answers, allowing us to use two experimental groups instead of three.

6.4 Specialists not needed

With one exception (the first author, an HCI expert) our team consisted of computer scientists, with no previous experience using the CW. At the start of the project, the first author briefly taught the team the method (in about 90 minutes), introducing the ideas behind it and presenting example walkthroughs. The walkthroughs of the experiment were made by the rest of the team, with the first author acting as observer during the first walkthrough only.

Although a team of HCI experts might have extracted more from the walkthroughs, our team certainly used it successfully enough to obtain useful results.

6.5 General lessons

With the use of the CW, four types of uncontrolled variables emerged (given in Section 4.2). Rephrasing them as "four helpful maxims" may be useful to designers of other empirical studies aimed at evaluating benefits of VPL features:

- 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.
- Design the tutorial with close scrutiny to participants' ability to master the material, considering not only content, but also length and practice time.
- If there is a user interface component *in* the feature under evaluation, make sure it has helpful action choices and labeling.
- If there is a user interface component present in the experiment but *unrelated* to the feature under evaluation, eliminate distracters that could impact ability to perform the experiment's task.

We propose the above maxims derived from our experience with applying the CW to this VPL evaluation experiment as a partial answer to the last question posed in the introduction.

7. Conclusion

Like John and Packer [10], 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. 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.

We also found that the use of the CW can improve the design of a VPL-focused empirical study. It was *effective*: 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 hours for a walkthrough in front of a VPL; it was *concretely focused*, requiring very little working out how an abstract theory or model could be applied to the specific situation; and it was *constructive*, leading directly to suggestions for improving the experiment design.

References

[1] B. Bell, W. Citrin, C. Lewis, J. Rieman, R. Weaver, N. Wilde and B. Zorn. Using the Programming Walkthrough to Aid in Programming Language Design. *Software Practice and Experience*, 24, 1994, 1-25.

[Blackwell, Metacognitive Theories of Visual Programming: What Do We Think We Are Doing? 1996 *IEEE Symp. Visual Languages*, Boulder, CO, Sept. 1996, 240-246.

[3] M. Burnett and H. Gottfried. Graphical Definitions: Expanding Spreadsheet Languages through Direct Manipulation and Gestures. *ACM Trans. Computer-Human Interaction* 5(1), Mar. 1998, 1-33.

[4] M. Burnett, J. Atwood, R. Djang, H. Gottfried, J. Reichwein and S. Yang. Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm. *J. Functional Programming*, (to appear).[5]

S. Card, T. Moran and A. Newell. The Keystroke-Level Model for User Performance Time With Interactive Systems. *Comm. ACM* 23(7), 1980, 396-410.

[6] W. Gray and M. Salzman. Damaged Merchandise? A Review of Experiments that Compare Usability Evaluation Methods. *Human-Computer Interaction*, 13(3), 1998, 203-261.

[7] T. R. G. Green. Cognitive Dimensions of Notations. In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V*. Cambridge Univ. Press. 1989, 443-460.

[8] T. R. G. Green and M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *J. Visual Languages and Computing,* 7, 1996, 131-174.

[9] J. Gurka and W. Citrin. Testing Effectiveness of Algorithm Animation. *1996 IEEE Symp. Visual Languages*, Boulder, CO, Sept. 1996, 182-189.

[10] B. John and H. Packer. Learning and Using the Cognitive Walkthrough Method: A Case Study Approach. *ACM CHI '95,* Denver, CO, May 1995, 429-436.

[11] J. Nielsen and R. Molich. Heuristic Evaluation of User Interfaces. *ACM CHI '90,* Seattle, WA, Apr., 1990, 249-256.

[12] P. Polson, C. Lewis, J. Rieman and C. Wharton. Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *Intl. J. Man-Machine Studies* 36, 1992, 741-773.

[13] J. Rieman, M. Franzke and D. Redmiles. Usability Evaluation with the Cognitive Walkthrough. *ACM CHI '95*, Denver, CO, May 1995, 387-388.

[14] G. Rothermel, L. Li, C. DuPuis and M. Burnett. What You See is What You Test: A Methodology for Testing Form-Based Visual Programs. *20th Intl. Conf. Software Engineering*, Kyoto, Japan, Apr. 1998, 198-297.

[15] K. J. Rothermel, C. Cook, M. Burnett, J. Schonfeld, T. R. G. Green and G. Rothermel. WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation. *Intl. Conf. Software Engineering*, Limerick, Ireland, June 2000, 230-239.

[16] A. Sears and D. Hess. The Effect of Task Description Detail on Evaluator Performance with Cognitive Walkthroughs. *ACM CHI '98*, Los Angeles, CA, Apr. 1998.

[17] C. Wharton, J. Bradford, R. Jeffries, M. Franzke. Applying Cognitive Walkthroughs to More Complex User Interfaces: Experiences, Issues, and Recommendations. *ACM CHI '92*, Monterey, CA, 1992, 381-388.

[18] M. Williams and J. Buehler. Comparison of Visual and Textual Languages via Task Modeling. *Int. J. Human-Computer Studies*, 1999, 51, 89-115.