WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Study of End Users

Curtis Cook, Brent Stevenson, Justin Schonfeld, Karen Rothermel, Margaret Burnett

Department of Computer Science
Oregon State University
Corvallis, OR 97331

{cook, brents, schonfju, rotherka, burnett}@cs.orst.edu

ABSTRACT

This paper reports on an empirical study involving end users that addresses the question of whether it is possible to provide the benefits of formal testing within the informal spreadsheet paradigm. We have developed a "What You See Is What You Test" (WYSIWYT) methodology that brings the benefits of formal testing in a way that does not require the users to learn the underlying software engineering theory about testing. Our WYSIWYT methodology attempts to do this by supplementing the immediate visual feedback about the values in cells with feedback about their "testedness". A previous controlled experiment using computer science students showed that the subjects who used the methodology were significantly more effective and more efficient in their testing and significantly less overconfident about the correctness of their spreadsheets than subjects who did not use the methodology. The results were obtained without providing any training in the theory of testing or test adequacy that the methodology implements. In this paper we report the results of a similar experiment that investigated whether spreadsheet end users with little or no programming experience would obtain the same benefits using the methodology.

Keywords: spreadsheets, testing, end users, visual programming, empirical studies

1. INTRODUCTION

Today, the most widely used programming paradigm is the spreadsheet paradigm, which includes not only the popular personal computer spreadsheet packages such as Excel and Lotus 123, but visual programming languages as well. Its user community ranges from experienced computer programmers to end users with little or no programming experience. Spreadsheets play a prominent role in business decision making and modeling. Yet, in spite of their widespread

use, almost no work has been done to help with the software engineering tasks involved in creating and maintaining spreadsheets.

Spreadsheet languages differ from most of the other commonly used programming languages in several ways. They provide a declarative approach to programming, characterized by dependence-driven, direct-manipulation working model [2]. Spreadsheet programmers create cells and define formulas for these cells. These formulas reference values in other cells and use them in their calculations. As soon as a cell is defined, the underlying evaluation engine immediately calculates its value and the values of cells that use its value and displays the new results.

Experiments and audits of operational spreadsheets have shown that they often contain errors [21] and that users have unwarranted confidence about their correctness [3, 6, 21, 23, 33]. In four reported audits of operational spreadsheets [21], errors were found in an average of 20.6% of the spreadsheets; in 11 experiments where participants created spreadsheets, errors were found in an average of 60.8% of the spreadsheets; in four other experiments where, the participants inspected spreadsheets for errors, the participants missed an average of 55.8% of the errors.

Studies have shown massive developer overconfidence about the accuracy of spreadsheets. In [3], all nine experienced spreadsheet developers were "very confident" that their spreadsheets were correct even though all of the developers had made at least one error. Panko [21] found the same high level of confidence even though the subjects made many errors. In spite of finding errors in 4 of the 19 spreadsheets audited, Davies and Ikin [6] reported developers were extremely confident about accuracy of the spreadsheets. Even though larger spreadsheets have a higher probability of containing errors, subjects in a study by Reithel [23] rated the accuracy of well-formatted large spreadsheets higher than plainly formatted large spreadsheets and higher than small spreadsheets. All of these studies suggest developers have an unwarranted high level of overconfidence about the correctness of their spreadsheets.

To address the reliability problem associated with spreadsheets, we are investigating methods of bringing some of the benefits of formal testing to the spreadsheet paradigm in a way that does not require the users to learn the underlying software engineering theory about testing. Our theory is that by providing feedback about the "testedness" of the spreadsheet, as opposed to values, we can cause programmers to have less overconfidence about the correctness of their spreadsheets, motivate them to test their spreadsheets more thoroughly, and provide guidance that helps them test more efficiently. The net result would be more reliable spreadsheets even in this informal programming domain.

To this end we have developed a "What You See Is What You Test" (WYSIWYT) methodology for testing spreadsheets [27]. The methodology is designed to accommodate the declarative environment of spreadsheet formulas, the incremental style of development, and the immediate visual feedback expected of spreadsheet languages. The methodology does not assume the programmer has any formal testing background.

The approach used in the methodology tracks the outputs that users have validated and the dependencies that have been exercised for the current set of cell formulas. But for the methodology to be of practical benefit, there are three questions that need to be answered: Can the methodology be implemented efficiently? Will it uncover errors in programs? Will it help programmers become more effective and efficient and less overconfident than programmers who do not use the methodology? Previous work [24, 26] has supported positive responses to the first two questions.

We have embarked on a series of empirical studies with human subjects to answer the third question. The first experiment in the series [27] showed that computer science students who used the methodology to test two spreadsheet programs were significantly more effective and efficient, and significantly less overconfident than computer science students who did not use the methodology. This showed that the methodology could benefit programmers with testing experience. In this paper we investigate whether the methodology helps end users that typically have no programming or testing experience. Spreadsheet end users are a large and important class of spreadsheet users because of the simplicity of the spreadsheet paradigm and the easy-to-use environment. They range from people who create spreadsheets for their own use (calculate stock portfolio) to people who use or create spreadsheets at work (accounts payable) to people who create spreadsheet templates for others to use (business models).

2. BACKGROUND

Most of the literature on testing programs is for imperative programs [7, 9, 11, 15, 22, 32]. There are also a few papers on testing functional or logic programs [2, 12, 14, 18]. The development of a testing methodology for spreadsheets is highly influenced by the differences between the imperative and spreadsheet language paradigms and between imperative and spreadsheet program developers. A first difference is that data dependencies between cells and explicit control flow only within cell formulas drive the evaluation engine for spreadsheet languages. This gives evaluation engines flexibility in the scheduling algorithms and optimization devices they employ in performing the calculations. Hence the testing methodology must not depend on a particular order of evaluation. Second, since spreadsheets are developed incrementally and there is

immediate feedback after each modification of a cell formula, the testing methodology must be able to operate on partially completed programs and to respond quickly and efficiently. A third, and probably the most important difference, is the testing methodology must not require formal training in testing principles. Professional programmers develop most imperative programs whereas spreadsheet programs are developed by a wide variety of users many of who have no formal training or experience in developing or testing programs [27]. Our spreadsheet testing methodology takes these factors into account.

The following gives an overview of the foundations of our testing methodology. For a more detailed presentation see Rothermel, Li and Burnett [26], and Rothermel, Li, DuPuis and Burnett [27]. Our methodology is centered on code-based adequacy criteria. Code-based adequacy criteria aid in the selection of test data and deciding when a program has been tested "enough" by relating the testing effort to coverage of program components. For example, in imperative languages a test suite satisfies program statement coverage criteria if execution of the test cases in the test suite exercises each statement in the program. The usefulness of code-based adequacy has been demonstrated in the considerable research of code-based adequacy criteria for imperative programs [9, 15, 22] and several empirical studies [8, 11, 32].

For our testing methodology we adapted the *output-influencing-All-du* dataflow adequacy criterion originally defined for imperative programs [7]. This criterion, which we call du-adequacy for short, focuses on the definition-use associations (du-associations) in a spreadsheet. A du-association links an expression (in a cell formula) that defines a cell's value with expressions in other cell formulas that reference or use the defined cell. There are two parts to the definition of the *du-association adequacy criterion for spreadsheet languages.* First, each executable du-association in the spreadsheet must be exercised by some test data; second the result of execution of the du-association must contribute to the display of a value that is subsequently pronounced correct (validated) by the programmer.

It may not be possible to develop test data that exercise all du-associations. A du-association that cannot be exercised is called *nonexecutable*. It is provably impossible in general, to determine whether a du-association is executable for either imperative or spreadsheet languages [9, 32]. Hence the dataflow adequacy criterion is typically restricted to executable du-associations. It should be noted that in our experience with spreadsheets, most of the nonexecutable du-associations we have encountered have involved direct contradictions that were relatively easy to identify [28].

The du-adequacy criterion is well suited to spreadsheet programs because interactions between the definitions and uses of cell values are a main source of

faults in spreadsheets. Since the criterion requires exercising of all du-associations, it will help expose many of these faults. As mentioned earlier, the spreadsheet evaluation engine does not impose any particular cell order execution. This does not impact du-associations since they are independent of cell order execution. Recall that the du-association criterion requires the display of a value that the programmer validates. Hence linking test coverage to cell validation avoids the problems of du-associations influencing only values hidden or off-screen. Finally, the criterion facilitates incremental testing of spreadsheets as it allows entering values in a subset of the possibly large number of input cells followed by validation of multiple cells.

3. EXPERIMENT DESIGN

The objectives of our study were the same as our previous study [28]. The major differences are the subject population (CS students vs. end users) and the problems used in the experiment. The research questions are:

RQ 1: Do end users who use our testing methodology create test suites that are more effective in terms of du-adequacy than end users who use an Ad Hoc approach?

RQ 2: Do end users who use our testing methodology create test suites more efficiently than end users who use an Ad Hoc approach?

RQ 3: Are end users who use our testing methodology less overconfident about the quality of their testing than end users who use an Ad Hoc approach?

These translate directly into hypotheses for our empirical study. In addition we exercised care in the design of our experiment so that it would provide insight into the following question about the training needed. This was especially important for this study since the participants were end users who had little or no programming experience.

Is training in the underlying test adequacy criterion and its relationship to the visual devices needed in order for spreadsheet programmers to gain testing effectiveness or efficiency from using our methodology?

We conducted a controlled laboratory experiment to address these questions. In the experiment, the subjects tested two spreadsheet programs. Half of the subjects used a spreadsheet environment with the WYSIWYT methodology for both programs while the other half used the same environment without the WYSIWYT methodology.

During the experiment we collected the actions of the subjects as they tested each spreadsheet program. These actions were automatically recorded in transcript files from which we derived the test suites created by the subjects. Other information included a background and post-testing session questionnaires in which the subjects rated how well they thought they tested the spreadsheet. On the second post-testing questionnaire, the WYSIWYT subjects answered additional questions about their use and understanding of our methodology's feedback. The questionnaires are included in Appendices 1 and 2.

3.1 Experimental Environment

We have prototyped our WYSIWYT testing methodology in the research language Forms/3 [4]. This choice was motivated by two factors. First, we have access to its implementation and thus can implement and experiment with various testing technologies within that environment. Second, and most important is that this is the experimental environment we used in our previous experiment [28].

Forms/3 spreadsheets are a collection of cells and each cell's value is defined by a formula. After a cell formula is entered, the programmer receives immediate visual feedback as the cell value is updated and the values of all cells impacted by the new cell values are updated. Figure 1 (Purchase Budget Spreadsheet) shows a Forms/3 spreadsheet that computes the total cost of a purchase order for office supplies, checks to see if sufficient supplies are ordered and compares the total cost to the budget amount. Other features of Forms/3 are the ability to reposition cells, hide or display cell formulas, and perform graphics computations. Figure 2 shows a grades spreadsheet that computes the average for 4 student quiz scores, calculates an extra credit bonus, and assigns an appropriate letter grade based on the average quiz score and extra credit bonus. In this figure some of the cell formulas are displayed.

Next we describe how WYSIWYT methodology can aid in testing a spreadsheet by indicating the testedness of the various parts of the spreadsheet. Recall that du-associations link expressions that define the cell's value with all other cell formulas that reference or use the defined cell. The du-association adequacy criterion specifies that the test data exercise each du-association and that the programmer validates each of the values as correct. For convenience we say that a cell is *tested* if all du-associations whose uses are in the cell have contributed to a value that the programmer pronounced correct. WYSIWYT methodology conveys information about the testedness of a cell through colors. A cell border is colored blue if the cell is fully tested, colored red if not tested, and purple if partially tested. Figure 2 has an example of all three. (In the figures, the colors are gray, black, and darker gray respectively for the different levels of "testedness".) Another way of showing dependency relations between cells is

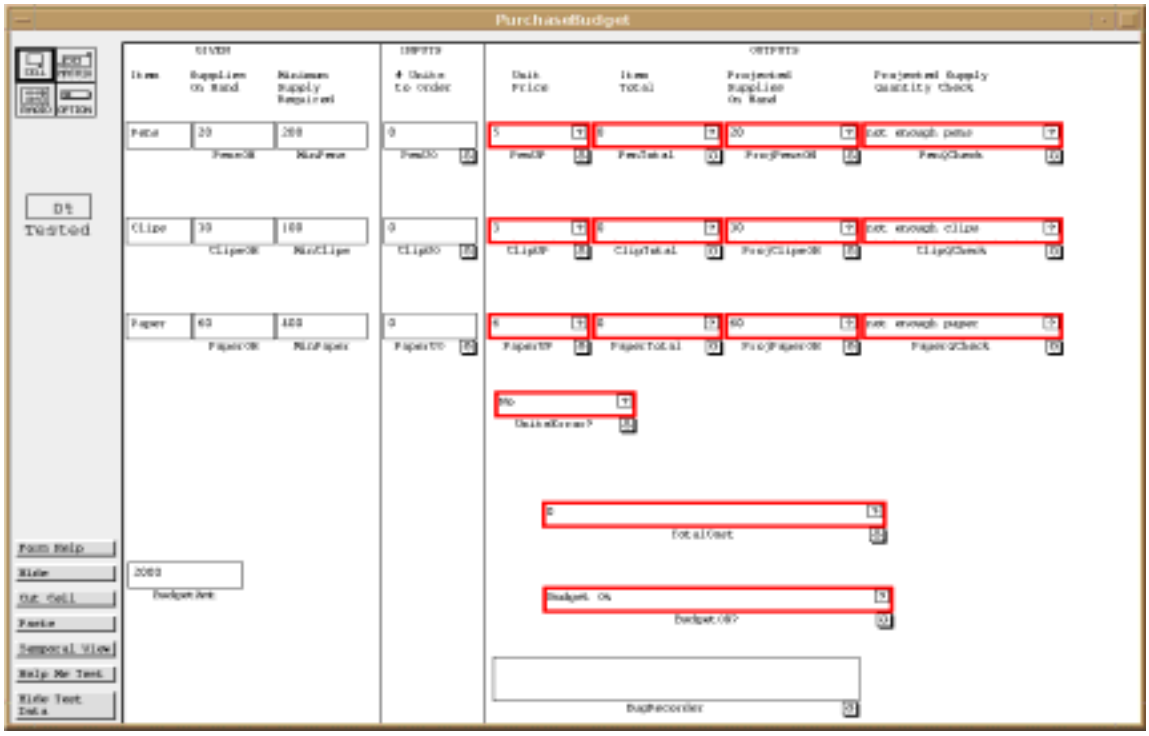


Figure 1: Purchase budget spreadsheet.

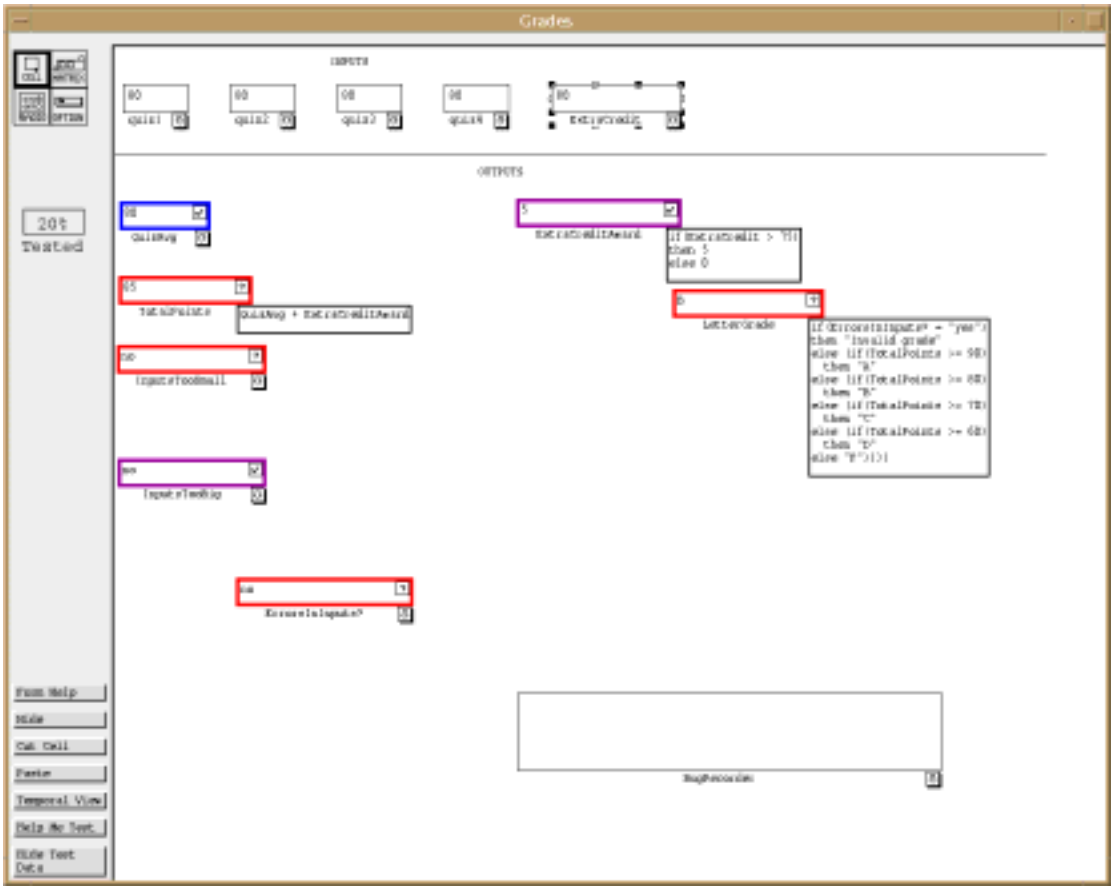


Figure 2 - Grades Spreadsheet with Formulas Displayed

arrows. A programmer may choose for each cell whether or not to display the arrows for that cell. There is arrow from an expression or cell to all other expressions or cells that use the value defined by that expression or cell. Arrows follow the same coloring scheme as for cell borders. See Figure 3.

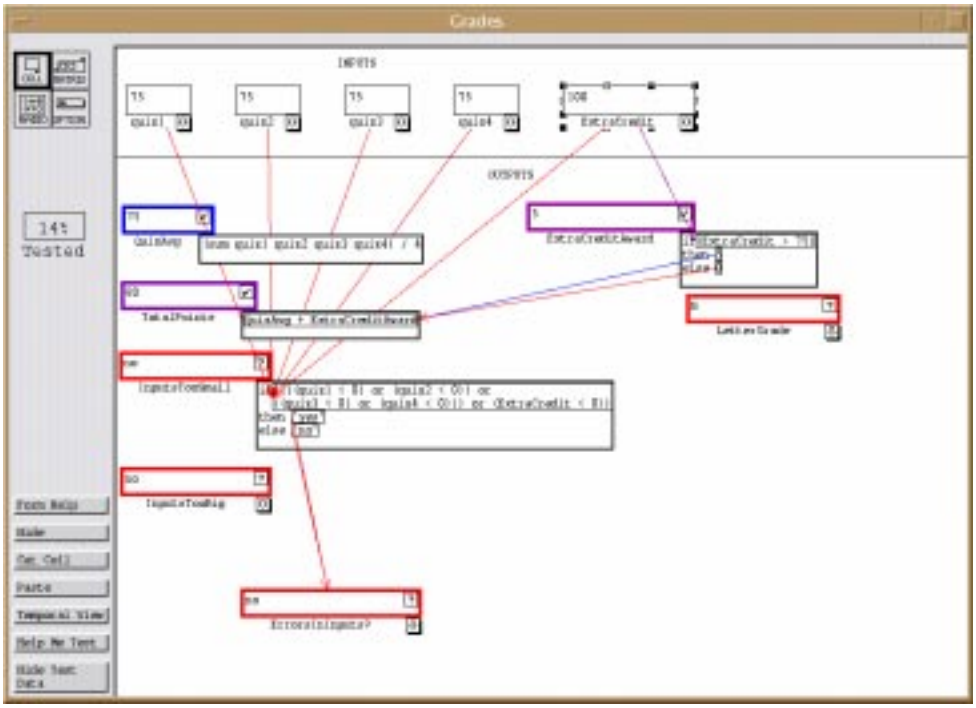


Figure 3: Grades Spreadsheet with Percent Tested, Colors and Arrows displayed.

When a programmer decides a cell value is correct for the given inputs he validates it by clicking in the check box in the upper right corner of the cell. This click causes the underlying validation algorithm [27] to recurse back through the du-associations that affect, directly and indirectly, the currently computed values of the cell and marks all of these cells tested (covered). The system responds with immediate feedback by changing colors of each (affected) visible cell and arrow to reflect the new testedness status. Cells and arrows are initially red or become red when a cell formula is changed which makes it easy to identify untested sections of the spreadsheet. Marks in the check box provide additional testing information. A question mark indicates that validating this cell's value will increase the spreadsheet's testedness since it will validate a previously unvalidated du-association; a blank indicates validating will not increase testedness; and a check mark indicates the user's validation was recorded. In the left column there is also a Percent Tested box that displays the percent of du-associations in the entire spreadsheet that have been tested. It is updated after each validation.

For the experiment half of the subjects used the environment described above. The other half used the same environment but without the colors, arrows, question marks, check marks, and percent tested indicator. In the latter environment, a check box is still present and used to communicate testing decisions. When the user clicked on the check box, the system responded with a quick flash to indicate the decision that the output value for the current inputs was recorded. The check boxes contain blanks during the experiment.

3.2 Subjects

Our previous experiment [28] investigated the impact of the testing methodology on subjects with programming experience – computer science juniors, seniors and graduate students. The goal of this experiment was to investigate the impact of the methodology on spreadsheet end users who work with spreadsheets in their jobs and who typically have little or no programming experience. Our previous study showed that subjects using the methodology were more effective and more efficient in their testing and less over-confident about the reliability of their spreadsheets. In this experiment we investigate whether this is also true for spreadsheet end users.

We advertised in the newspaper, campus publications and appropriate mailing lists for subjects. The advertisements indicated we were interested in spreadsheet users and developers to participate in a study, and that they should have a basic knowledge of simple spreadsheet formulas. Nearly 50 people responded but only 45 were available during the week the seven experimental sessions were scheduled. Subjects were randomly assigned to sessions that fit their time preferences. However, only 35 showed up for their scheduled sessions. Subjects were paid ten dollars for participating. The sessions were randomly assigned to Ad Hoc or WYSIWYT subject to balancing the number of subjects in each group. The control group (Ad Hoc) did not have access to our WYSIWYT methodology and represent end users that test in an Ad Hoc fashion. The treatment (WYSIWYT) group did have access to our methodology.

Of the 35 subjects, 17 were in the Ad Hoc group and 18 in the WYSIWYT group. The size of the Ad Hoc group was decreased by one when we decided to omit data for subjects whose measurable activity was zero (no test cases as revealed by the transcripts of the sessions). The removal of this subject reduced the number of subjects in the Ad Hoc group to 16.

To ascertain whether the subjects in the two groups had reasonably similar backgrounds and spreadsheet experience, we administered a questionnaire

(Appendix 1) to each subject and analyzed the results. Spreadsheet experience included the number of years using spreadsheets, whether used in work or for personal use, and whether they have created spreadsheets or have modified spreadsheets created by other people. Other background information included whether they had ever written computer programs and highest level of education. Table 1 gives a summary of the results. Our analysis showed the two groups were homogeneous.

	No. of Subjects	Average Years Exper	Use in Work	Personal Use	Created	Modified	Never Program	AA or BA/BS
Ad Hoc	16	6.69	88%	75%	81%	81%	50%	50%
WYSIWYT	18	10.11	94%	89%	89%	78%	56%	56%

Table 1 Subject group demographics.

3.3 Tutorial

The experiment was conducted in two different labs in multiple sessions in order to accommodate the participant's schedules. One lab was an open computer lab with no partitions separating 24 workstations while the second lab was more of an "office-like setting" with low partitions between 4 workstations. For the experiment the subjects were seated one per workstation using Forms/3.

The experiment began with a twenty-five minute tutorial of Forms/3 and testing in which each subject actively participated by working with example spreadsheet following instructions given by the lecturer. Throughout the tutorial the subjects had access to a quick reference sheet (Appendix 4) of the spreadsheet features they were being taught. They could make notes on this sheet that remained available to them throughout the experiment. The tutorial introduced the subjects to the basic language features (e.g. basic syntax) and environment features (e.g. how to edit a cell formula) of Forms/3. Testing was described as the process of trying various input values and recording decisions about the correctness of the cell values. All subjects were instructed to record a decision that a cell value was correct by clicking on the cell's checkbox and to record information about incorrect cell values in a special cell named BugRecorder. Because the subjects were end users, with little if any programming experience, the tutorial also provided basic information and hints about choosing test cases (e.g. try all input combinations, look at the formulas or problem description). This instruction in testing may have overly affected subject's representativeness of end users, and we will examine it closely in future experiments.

The WYSIWYT subjects version of the tutorial included information on the methodology's feedback - colored cell borders and arrows, the percent tested indicator and question marks, blanks, and check marks in the cell checkbox. The subjects were told that red means untested, blue tested, and purple partially tested. Question marks were described as "recording a decision here will help test a part of the spreadsheet that has not been tested", check marks as "you have recorded a decision here", and blanks as "you have previously recorded a decision that covers this case." There was no mention of the underlying concepts of du-associations and no mention of nonexecutable du-associations. Note that when the Ad Hoc subjects validated a cell by clicking in its checkbox, the checkbox remained blank, but the cell flashed to indicate the decision was recorded.

The total time given for the tutorial and unstructured practice was the same for both the Ad Hoc and WYSIWYT groups. Since the extra material on the methodology's feedback made the WYSIWYT tutorial part longer, the Ad Hoc subjects were given more unstructured testing time.

3.4 Task and Materials

The subject task was to test two spreadsheets, Grades and Purchase Budget. See Appendix 3 for descriptions of each spreadsheet. The Grades spreadsheet (Figure 2) was similar to the one used in our previous experiment, but simplified to reduce the number of conditional expressions and the use of a summation function. It computed the final grade of a student based on homework and quiz scores. The Purchase Budget (Figure 1) was designed to be similar to the accounting type spreadsheets most of the end users had encountered. Both spreadsheets were designed to not require any special domain knowledge and to be of reasonable complexity given the limited amount of time for testing. In developing the Purchase Budget we examined many spreadsheets used by accounting people at our university to confirm that our problem selection was in the proper domain. It is interesting to note that in our examination of these spreadsheets we rarely found a cell formula with a condition and the most complicated formulas computed row and column summations.
The subjects had 15 minutes to test each spreadsheet. The experiment was countered balanced with respect to problem type as half the subjects tested the Grades spreadsheet first and Purchase Budget second. The order was reversed for the other half. At the start of each testing session the subjects were instructed to read the problem description for detailed information about what the spreadsheet did, the range of input and output values, and the error messages for out-of-range inputs.

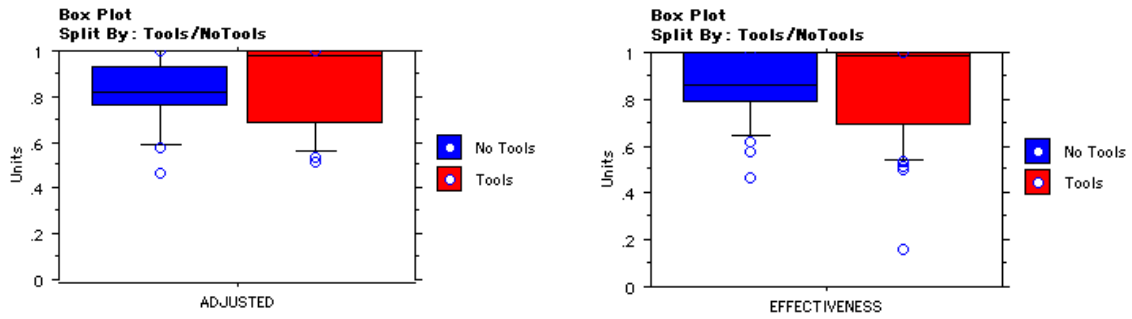
4. RESULTS

4.1 Effectiveness

The first research question is whether the end users who used the WYSIWYT methodology tested more effectively than the end users who did not use the methodology. As in our previous experiment we measured effectiveness in terms of du-adequacy: given a spreadsheet what percent of the executable du-associations were exercised by the subject's test suite.

The data given in Figure 4 shows evidence of skewness. We analyzed effectiveness using the ANOVA since it is robust against moderate skewness and also used non-parametric tests on the two problems treated separately. The two factors for the ANOVA were environment (WYSIWYT and Ad Hoc) and problem (Grades or Purchase Budget). Since each subject used one environment and tested both problems, the environment factor was treated as independent groups and the problem factor was treated as a having repeated measures.

The ANOVA showed no significance difference in effectiveness between the WYSIWYT and Ad Hoc groups. However, there was a significant interaction effect between the two problems (F=5.045, df=1.32, p=0.317) which reflects that the WYSIWYT group was more effective on the Grades problem while the Ad Hoc group was more effective on the Purchase Budget problem. Even so, independent non-parametric tests (Mann-Whitney) on each problem separately showed no significant differences between the WYSIWYT and Ad Hoc groups.



Grades Purchase Budget

Figure 4: Box plots of testing effectiveness for each problem.

4.2 Efficiency

Whether the methodology aided subject testing efficiency was the second research question. One way to measure efficiency is in terms of wasted effort. A

redundant test case is one that does not exercise a previously uncovered du-association, and thus redundant test cases can be considered as wasted effort. Efficiency was measured in three ways: the percent of the total test cases that were not redundant, the number of test cases and how quickly the subjects attained coverage.

The percent of non-redundant test cases may be overly simplistic [24], but it is commonly used with structural coverage criteria. The redundant test case data is given in Table 2. An ANOVA of the data showed that the WYSIWYT group created a significantly lower percent of redundant test cases than the Ad Hoc group ($F=55.770$, $df=1,32$, $p<0.0001$). There was a significant difference between the percentage of redundant test cases on the two problems ($F = 20.890$, $df=1,32$, $p <0.0001$) but no significant interaction effects; that is, the WYSIWYT subjects showed the same pattern of lower redundancy percentage on both problems. An independent, parametric (Mann-Whitney) test on each problem separately confirmed the significant differences between the WYSIWYT and Ad Hoc environments.

	Grades		Purchase Budget	
	# Tests	Redundancy	# Tests	Redundancy
Ad Hoc	46.0	71%	58.4	53%
WYSIWYT	16.6	20%	26.8	12%

Table 2: Mean number of test cases and redundancy percentages.

Grades	0-5 min	5-10 min	10-15 min
Ad Hoc	37%	65%	82%
WYSIWYT	37%	72%	87%
Purchase Budget	**0-5 min**	**5-10 min**	**10-15 min**
Ad Hoc	33%	73%	90%
WYSIWYT	31%	72%	85%

Table 3: Mean coverage percentages at 5-minute intervals.

A second measure of efficiency is the number of test cases. ANOVA showed the WYSIWYT subjects ran significantly fewer test cases than the Ad Hoc subjects ($F = 23.866$, $df=1,32$, $p< 0.0001$) and significantly fewer test cases on the two problems ($F = 19.176$, $df=1,32$, $p=0.0001$). Again Mann-Whitney tests on each problem separately confirmed the significant differences in number of test cases between the WYSIWYT and Ad Hoc environments. In our previous experiment [28] where there was no significant difference in the number of test cases for either group or either problem. However, the WYSIWYT subjects were significantly more effective in their testing. Thus we can consider the WYSIWYT

subjects in our previous more efficient than the Ad Hoc subjects because they achieved higher du-adequacy with the same number of test cases.

A third way to measure efficiency is the speed with which the subjects attained coverage. The 15-minute testing session was divided into three 5-minute intervals. Du-adequacy was computed at the end of each interval. As Table 3 suggests there was no significant difference in the speed data at any interval for either problem between the Ad Hoc and WYSIWYT subjects.

4.3 Over-confidence

As mentioned previously in this paper, several studies reported spreadsheet programmers expressed unwarranted confidence in the correctness of their spreadsheets. The third research question investigates whether the WYSIWYT subjects were less overconfident about the correctness of their spreadsheets than the Ad Hoc subjects. This is an important question for our methodology. Our methodology would be of little practical use if it caused our subjects to stop testing early and not use the feedback it provides.

At the end of each testing session the subjects rated to how well they thought they tested the spreadsheet (see Appendix 2) by answering the following question:

How well do you think you tested the spreadsheet?
a) Really, really well. (If you graded it, I'd get an A)
b) Better than average. (If you graded it, I'd get a B)
c) About average. (If you graded it, I'd get a C)
d) Worse than average. (If you graded it, I'd get a D)
e) Poorly. (If you graded it, I'd get an F)

To determine whether a subject was overconfident we compared the subject's self-rating to our "grading" based on their du-adequacy. We assigned letter grades A-F (A if 90-100% coverage, B if 80-89% coverage, C if 70-79% coverage, D if 60-69% coverage, and F if 0-59% coverage). A subject was categorized as "overconfident" if his self-grade was higher than our grade. Table 4 shows that a higher percentage of the Ad Hoc subjects were overconfident compared to the WYSIWYT subjects. However, only for the Grades program was this difference significant (Fisher Exact Test, $p = 0.0229$). In our previous experiment significantly more of the Ad Hoc subjects were overconfident for each problem.

5. DISCUSSION

In our previous experiment the WYISYT subjects were significantly more effective and more efficient than the Ad Hoc subjects while the Ad Hoc subjects were

significantly more overconfident about the correctness of both of their spreadsheets. However, in this experiment we only found the WYSIWYT subjects significantly more efficient and the Ad Hoc subjects significantly overconfident for only one of the two spreadsheets.

	Grades	
	Overconfident	Not Overconfident
Ad Hoc	8	8
WYSIWYT	2	16
	Purchase Budget	
	Overconfident	Not Overconfident
Ad Hoc	5	11
WYSIWYT	3	15

Table 4: Overconfidence: Number of subjects in each group categorized according to overconfidence.

The lack of a difference in the testing effectiveness was a disappointment. Possible reasons include the subjects were unable to use the methodology effectively, the two spreadsheet programs were so simple that the methodology was not needed, or the Ad Hoc group was not truly representative of "normal" spreadsheet users since we taught them testing skills in the tutorial. The questionnaire the WYIWYT subjects completed after testing the second spreadsheet asked them to rate the helpfulness of the various visual devices (colored cell borders, colored arrows, question marks, percent tested indicator, etc.) and their understanding of the information communicated by these devices. Helpfulness ratings are given in Table 5 and subject understanding in Table 6.

Subject's ratings of the helpfulness of the visual devices could be misleading if they did not understand what information was being communicated by these devices. This is especially important given the background of our subjects and the fact that they had only 25 minutes to learn to use the Forms/3 environment and our methodology. Table 6 gives the percent of correct subject responses to the three multiple-choice questions designed to assess the subject's understanding of our methodology.

Subject ratings suggest they did find the visual devices helpful and they seemed to understand their meaning. In fact their responses and ratings followed similar, but slightly lower, pattern to the subject responses in our previous study.

The two spreadsheets in this experiment were designed to be simpler than the ones in our previous experiment. However, the 90% median du-adequacy for the

How helpful were:	Very Helpful	Helpful	Not Helpful
Colored cell borders	78%	22%	0%
Question marks	72%	28%	0%
Clicking to validate	67%	33%	0%
Check marks	50%	39%	11%
Colored arrows	33%	56%	11%
"Tested" indicator	28%	56%	17%
Blanks	22%	56%	22%

Table 5: WYSIWYT subject's helpfulness rating: Percent of subjects who rated the device in each of the possible categories.

A red cell border indicates that the cell is (not tested)	89%
A question mark in a cell's checkbox indicates that (the current input tests part of the spreadsheet not previously tested)	61%
What does a blue arrow between cells indicate? (the relationship between the two cells is fully tested)	72%

Table 6: Subject's opinions about meaning of visual devices. Percentages reflect number who selected correct response (shown in parentheses).

two problems was about 10% higher than the median du-adequacy for the two problems in our previous experiment. It suggests that the problems may have been too simple. To look into this, we conducted short telephone interviews with 13 (5 WYSIWYT and 8 Ad Hoc) of the subjects. This non-scientific follow-up occurred one to two weeks after the experiment. Subjects were asked questions about the difficulty of the two spreadsheet problems compared to spreadsheets they normally worked with, the degree of testing they did in the experiment compared to what they normally do, and whether the environment in the experiment interfered with their testing. Most of the subjects phoned felt that the two problems were simple compared to spreadsheets they normally work with. Others said the problems were simple, but different because the spreadsheets they work with have considerable cross-footing and column and row summations. All of the subjects found the environment easy to use and said it did not interfere with their testing. Almost all of them felt they did more testing in the experiment than they normally would do. Hence it seems more plausible that a combination of the problems being too simple and teaching testing in the tutorial, rather than the subjects not being able to use our methodology, contributed to the lack of a significant difference in testing effectiveness between the two groups.

While all of the 5 WYSIWYT subjects in the phone interview felt that the cell border colors and arrows were helpful, they also mentioned becoming frustrated

because they were not able to either turn all of the cell borders or arrows blue or get the Percent Tested indicator to 100%. This was because of non-executable du-associations. They had plenty of time to do all of the testing they wanted but became frustrated when they were unsuccessful in their attempts to generate test cases to exercise these non-executable du-pairs. This may explain why the colored arrows and percent indicator were rated lower than the other visual devices in Table 5.

The significantly higher testing efficiency both in this experiment and our previous experiment strongly suggests our methodology is useful in helping subjects keep track of which conditions (du-associations) their test cases have exercised. This is reflected in the fewer redundant test cases and the generation of fewer numbers of test cases. It would seem that the symbols in the checkbox for the WYSIWYT version are most helpful here. Recall that a question mark in the checkbox informs the user that validating the current inputs will test a du-association that has not been previously validated. A blank or check mark in the check informs the user that validating will not validate a new du-association. Meanwhile an Ad Hoc subject only sees blank check boxes and the screen flashes when he clicks the check box. Hence the checkbox symbols help the WYSIWYT subjects keep track of du-associations that have been exercised while the Ad Hoc subjects were forced to do their own keeping track of which du-associations they have exercised.

6. THREATS TO VALIDITY

As in our previous study we attempted to address threats to internal and external validity. For internal validity the two groups of subjects were counterbalanced with respect to problem type, had equal training time, and the problems selected were from familiar domains. This study overcame one serious threat to external validity of our previous study, namely the subjects being representative of spreadsheet users. Subjects in this study were spreadsheet users and developers. However, as in most controlled experiments, the task of running test cases, limited time to do the testing of the spreadsheets, and size and simplicity of the spreadsheets work against the generalization of the results.

The two spreadsheets used in the experiment did not contain faults because the focus was on the effectiveness and efficiency of testing. Furthermore, the subjects were prevented from changing any of the non-input cells so that no formulas could change. This may be unrealistic, but seeding the spreadsheets with faults, or allowing the users to create uncontrolled formulas would have confounded the experimental data on testing effectiveness and efficiency.

Testing effectiveness was measured by the percent of du-associations exercised by the subject's test cases rather than faults detected. This is supported by several empirical studies of imperative programs [8, 11, 32] that found a correlation between du-adequacy and effectiveness in finding faults. However, we plan to evaluate the fault detection effectiveness of the test suites generated by our subjects using ideas from mutation testing [17]. A *mutant* of a program P is a version of P created by the introduction of a single fault. Examples of mutants are replacing a plus operator with a minus operator, or adding or subtracting one from a constant, or replacing one variable with another variable. A test case *kills* a mutant when the test case causes the program to fail (e.g. fault to manifest itself by producing incorrect output). The goal of mutation testing is to generate a set of test cases that kill all of the mutants. The mutants remaining alive after all of the test cases are either functionally *equivalent* to the original program (output of mutant is same as output of original program) or killable but the test cases are insufficient to kill it. The *mutation score* is the percentage of non-equivalent mutants killed by the test set. We have recorded the test cases generated by each subject during their testing. The mutation score will measure testing effectiveness for this test set.

7. CONCLUSIONS

Our testing methodology and experimentation represents the first attempt to formally develop a software engineering approach for the informal spreadsheet paradigm. Even though we did not show that end users using our methodology were more effective in their testing, we did show they tested more efficiently and were less overconfident. Further, we showed that end users could easily learn to use our methodology.

However, one problem that surfaced was the frustration users of our methodology experienced when they attempted but were unable to move the percent tested indicator to 100% or turn all of the colored cell borders and arrows blue because of non-executable du-associations. We are investigating ways of mitigating this frustration.

9. Acknowledgements

We thank Jennifer DeYoung, Andy Ko and other members of the Visual Programming Research Group for their help with the experiment administration and implementation and their feedback on the testing methodology. We would also like to thank the end users that took the time to participate in the study. This work was supported in part by NFS under EIA-9806821. Patent pending.

REFERENCES

[1] A. Ambler, M. Burnett, and B. Zimmerman. Operational versus definitional: A perspective on programming paradigms. Computer, 25(9):28-43, Sept. 1992.

[2] F. Belli and O. Jack. A test coverage notion for logic programming. In The 6th Intl. Symp. Softw. Rel. Eng., pages 133-142, 1995.

[3] P. Brown and J. Gould. Experimental study of people creating spreadsheets. ACM Trans. Office Info. Sys., 5(3):258-272, July 1987.

[4] M. Burnett and H. Gottfried. Graphical definitions: Expanding spreadsheet languages through direct manipulation and gestures. ACM Trans. Computer_Human Interaction, pages 1-33, Mar. 1998.

[5] E. H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In IEEE Symp. Info. Visualization, Oct. 1997.

[6] N. Davies and C. Ikin, "Auditing spreadsheets", Australian Account, 1987, pp. 54-56.

[7] E. Duesterwald, R. Gupta, and M. L. Soffa. Rigorous data flow testing through output influences. In Proc. 2nd Irvine Softw. Symp., Mar. 1992.

[8] P. Frankl and S. Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. IEEE Trans. Softw. Eng., 19(8):774-787, Aug. 1993.

[9] P. Frankl and E. Weyuker. An applicable family of data flow criteria. IEEE Trans. Softw. Eng., 14(10):1483-1498, Oct. 1988.

[10] D. Gilmore. Interface design: Have we got it wrong? Chapman & Hall, London, England, 1995.

[11] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments on the effectiveness of dataflow and controlflow-based test adequacy criteria. In 16th Intl. Conf. Softw. Eng., pages 191-200, May 1994.

[12] W. Kuhn and A. U. Frank. The use of functional programming in the specification and testing process. In Intl. Conf. and Wkshp. Interoperating Geographic Info. Systems, Dec. 1997.

[13] J. Leopold and A. Ambler. Keyboardless visual programming using voice, handwriting, and gesture. In 1997 IEEE Symp. Vis. Lang., pages 28-35, Sept. 1997.

[14] G. Luo, G. Bochmann, B. Sarikaya, and M. Boyer. Control_flow based testing of prolog programs. In The 3rd Intl. Symp. Softw. Rel. Eng., pages 104-113, 1992.

[15] M. Marre and A. Bertolino. Reducing and estimating the cost of test coverage criteria. In 1996 IEEE 18th Intl. Conf. Softw. Eng., pages 486-494, Mar. 1996.

[16] B. Myers. Graphical techniques in a spreadsheet for specifying user interfaces. In ACM CHI '91, pages 243-249, Apr. 1991.

[17] A. J. Offutt, G. Rothermel, R. H. Untch, and C. Zapf, "An experimental Determination of Sufficient Mutant Operators", *ACM Transactions of Software Engineering Methodology* 5(2) April 1996, pp. 99-118.

[18] F. Ouabdesselam and I. Parissis. Testing techniques for data-flow synchronous programs. In AADEBUG'95: 2nd Intl. Wkshp. Automated and Algorithmic Debugging, May 1995.

[19] R. Panko. What we know about spreadsheet errors. J. End User Comp., pages 15-21, Spring 1998.

[20] R. Panko and R. Sprague. Hitting the wall: Errors in developing and code inspecting a "Simple Spreadsheet" model, *Decision Support Systems*, 1997.

[21] R. Panko and R. Halverson. Spreadsheets on trial: A survey of research on spreadsheet risks. In 29th Hawaii Intl. Conf. System Sciences, Jan. 1996.

[22] S. Rapps and E. J. Weyuker. Selecting software test data using data flow information. IEEE Trans. Softw. Eng., 11(4):367-375, Apr. 1985.

[23] B. J. Reithel, D.L. Nichols and R.K Robinson, "An Experimental Investigation of the Effects of Size, Format, and Errors on Spreadsheet Reliability Perception, Journal of Computer Information Systems, 1996, pp. 54-64.

[24] G. Rothermel, M. Burnett, L. Li, C. DuPuis, and A. Sheretov. A methodology for testing spreadsheets. Technical Report TR: 99-60-02, Oregon State University, Jan. 1999.

[25] G. Rothermel, M. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In Proc. Int'l. Conf. Softw. Maint., pages 34-43, Nov. 1998.

[26] G. Rothermel, L. Li, and M. Burnett. Testing strategies for form-based visual programs. In The 8th Intl. Symp. Softw. Rel. Eng., pages 96--107, Nov. 1997.

[27] G. Rothermel, L. Li, C. DuPuis, and M. Burnett. What you see is what you test: A methodology for testing form-based visual programs. In The 20th Intl. Conf. Softw. Eng., pages 198-207, Apr. 1998.

[28] K. Rothermel , C. Cook, M. Burnett, J. Schonfeld, T.R.G. Green, and G. Rothermel. "WYSIWYT testing in the spreadsheet paradigm: An empirical evaluation", to appear in Proceeding International Conference on Software Engineering, Limerick, 2000.

[29] T. Smedley, P. Cox, and S. Byrne. Expanding the utility of spreadsheets through the integration of visual programming and user interface objects. In Adv. Vis. Int. '96, May 1996.

[30] G. Svendsen. The influence of interface style on problem-solving. Intl. J. Man_Machine Studies, 35:379-397, 1991.

[31] G. Viehstaedt and A. Ambler. Visual representation and manipulation of matrices. J. Vis. Lang. and Comp., 3(3):273-298, Sept. 1992.

[32] E. J. Weyuker. More experience with dataflow testing. IEEE Trans. Softw. Eng., 19(9): 912-919, Sept. 1993.

[33] E. Wilcox, J. Atwood, M. Burnett, J. Cadiz, and C. Cook. Does continuous visual feedback aid debugging in direct-manipulation programming systems? In ACM CHI'97, pages 22--27, Mar. 1997.

APPENDICES

Appendix 1: Background Questionnaire

Appendix 2: Post session Questionnaires

Appendix 3: Statement of 2 problems

Appendix 4: Quick Reference Cards

Appendix 1: Background Questionnaire

All Participants Background Questions

1. Identification Code (from sticker on computer screen) _____

2. Spreadsheet languages used:

__ MS-Excel __ Lotus-123 __ Works __ Corel-Quattro Pro __ Other_____

3. Years of experience using spreadsheets: _____

4. I have used spreadsheets for

__ work __ personal other_____

5. I have

__ created spreadsheets __ modified other people's spreadsheets __ used spreadsheets

6. Programming experience (other than spreadsheets):

I have:

__ never written programs __ written programs for work
__ written programs for class assignments __ written programs for fun

Years of programming experience: _____

7. Highest level of education:

__ High School __ Some College __ AA __ BS/BA __ Other _____

8. Is English your primary language? If not, how long have you been speaking English?

Appendix 2: Post session Questions

QUESTIONS FOR PROBLEM 1 (WYSIWYT)

1. How well do you think you tested the spreadsheet?

   ____ Really, really well. (If you graded it, I'd get an A)
   ____ Better than average. (If you graded it, I'd get a B)
   ____ About average. (If you graded it, I'd get a C)
   ____ Worse than average. (If you graded it, I'd get a D)
   ____ Poorly. (If you graded it, I'd get an F)

2. How much additional testing time would you need before you would feel comfortable giving this spreadsheet to the group that needs to use it?

   ____ You gave us too much time, I really used only the first _____ minutes.
   ____ You gave us just about the right amount of time, I don't need any extra.
   ____ You gave us too little time, I need about _____ more minutes.

3. Of the tests you would have liked to run on the spreadsheet, what percent did you perform?

   ____ 110% or more (I did extra since I had so much time.)
   ____ 100% or more
   ____ 90% or more
   ____ 80% or more
   ____ 70% or more
   ____ Less than 70%

QUESTIONS FOR PROBLEM 2 (WYSIWYT)

1. How well do you think you tested the spreadsheet?

   ____ Really, really well. (If you graded it, I'd get an A)
   ____ Better than average. (If you graded it, I'd get a B)
   ____ About average. (If you graded it, I'd get a C)
   ____ Worse than average. (If you graded it, I'd get a D)
   ____ Poorly. (If you graded it, I'd get an F)

2. How much additional testing time would you need before you would feel comfortable giving this spreadsheet to the group that needs to use it?

   ____ You gave us too much time, I really used only the first _____ minutes.
   ____ You gave us just about the right amount of time, I don't need any extra.
   ____ You gave us too little time, I need about _____ more minutes.

3. Of the tests you would have liked to run on the spreadsheet, what percent did you perform?

   ____ 110% or more (I did extra since I had so much time.)
   ____ 100% or more
   ____ 90% or more
   ____ 80% or more
   ____ 70% or more
   ____ Less than 70%

4. Do you think you found all the bugs in:

   Spreadsheet #1 yes no
   Spreadsheet #2 yes no

The following questions refer to both testing problems:

5. How helpful were the following?

Seeing the colored arrows was:	Very Helpful	Helpful	Not Helpful
Seeing the cell border colors was:	Very Helpful	Helpful	Not Helpful
Seeing a question mark in a checkbox was:	Very Helpful	Helpful	Not Helpful
Clicking a checkbox to record a decision was:	Very Helpful	Helpful	Not Helpful
Seeing a check mark in a checkbox was:	Very Helpful	Helpful	Not Helpful
Seeing a blank in a checkbox was:	Very Helpful	Helpful	Not Helpful
Seeing the % Tested indicator was:	Very Helpful	Helpful	Not Helpful

Please check the last choice if you have to guess at the answer.

6. A red cell border indicates that the cell is

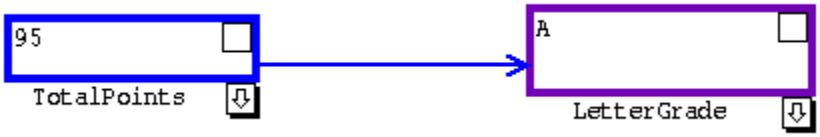
____ Not tested
____ Partially tested
____ Fully tested
____ I'm not sure what it indicates

Please check the last choice if you have to guess at the answer.

7. A question mark in a cell's checkbox indicates that

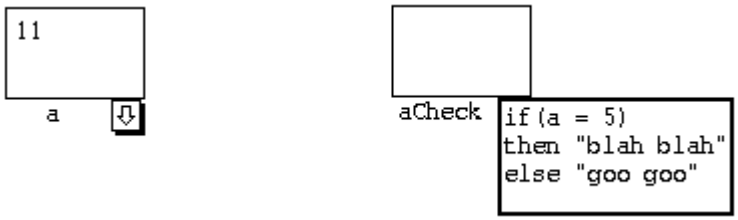
____ the cell's formula is not correct for the current inputs
____ the current input tests part of the spreadsheet not previously tested
____ the cell's value is correct for the current inputs
____ None of the above
____ I'm not sure what it indicates

8. Suppose the arrow in the picture below is blue. What does a blue arrow between cells indicate?



____ the cells at both ends of the arrow are both fully tested
____ the relationship between the two cells is fully tested
____ the cell the arrow comes from is fully tested
____ None of the above
____ I'm not sure what it indicates

9. Given the cells pictured below, what is the correct output value for the cell aCheck?



____ blah blah
____ goo goo
____ 11
____ None of the above
____ I'm not sure what the correct output should be

10. In your previous experience with spreadsheets have you ever:

Tested a spreadsheet to decide if it was working correctly?	yes	no
Found an error in a spreadsheet that was supposed to be correct?	yes	no
Corrected an error in a spreadsheet that was supposed to be correct?	yes	no

QUESTIONS FOR PROBLEM 1(Ad Hoc)

1. How well do you think you tested the spreadsheet?

   ____ Really, really well. (If you graded it, I'd get an A)
   ____ Better than average. (If you graded it, I'd get a B)
   ____ About average. (If you graded it, I'd get a C)
   ____ Worse than average. (If you graded it, I'd get a D)
   ____ Poorly. (If you graded it, I'd get an F)

2. How much additional testing time would you need before you would feel comfortable giving this spreadsheet to the group that needs to use it?

   ____ You gave us too much time, I really used only the first _____ minutes.
   ____ You gave us just about the right amount of time, I don't need any extra.
   ____ You gave us too little time, I need about _____ more minutes.

3. Of the tests you would have liked to run on the spreadsheet, what percent did you perform?

   ____ 110% or more (I did extra since I had so much time.)
   ____ 100% or more
   ____ 90% or more
   ____ 80% or more
   ____ 70% or more
   ____ Less than 70%

QUESTIONS FOR PROBLEM 2 (Ad Hoc)

1. How well do you think you tested the spreadsheet?
   ____ Really, really well. (If you graded it, I'd get an A)
   ____ Better than average. (If you graded it, I'd get a B)
   ____ About average. (If you graded it, I'd get a C)
   ____ Worse than average. (If you graded it, I'd get a D)
   ____ Poorly. (If you graded it, I'd get an F)

2. How much additional testing time would you need before you would feel comfortable giving this spreadsheet to the group that needs to use it?

   ____ You gave us too much time, I really used only the first _____ minutes.
   ____ You gave us just about the right amount of time, I don't need any extra.
   ____ You gave us too little time, I need about _____ more minutes.

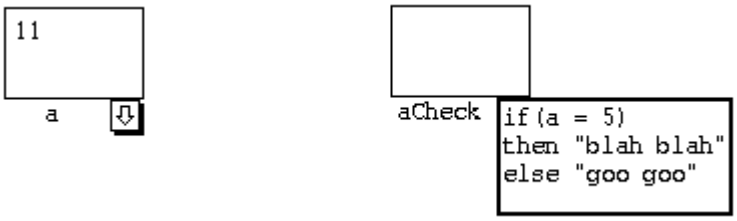
3. Of the tests you would have liked to run on the spreadsheet, what percent did you perform?

   ____ 110% or more (I did extra since I had so much time.)
   ____ 100% or more
   ____ 90% or more
   ____ 80% or more
   ____ 70% or more
   ____ Less than 70%

4. Do you think you found all the bugs in:

   Spreadsheet #1 yes no
   Spreadsheet #2 yes no

5. Given the cells pictured below, what is the correct output value for the cell aCheck?



____ blah blah
____ goo goo
____ 11
____ None of the above
____ I'm not sure what the correct output should be

6. In your previous experience with spreadsheets have you ever:

Tested a spreadsheet to decide if it was working correctly?	yes no
Found an error in a spreadsheet that was supposed to be correct?	yes no
Corrected an error in a spreadsheet that was supposed to be correct?	yes no

Appendix 3: Statement of 2 problems

Grades

The Grades spreadsheet uses 4 quiz scores and an extra credit score as course letter grade of A, B, C, D, or F.

The letter grade is based on the student's total points:
90 and up => A
80 – 89 => B
70 – 79 => C
60 – 69 => D
0 – 59 => F

The student's total points is the sum of the extra credit award and the The extra credit award is 5 points if the extra credit score > 75, othe

The quiz and extra credit scores may range from 0 to 100. For invalid i spreadsheet outputs an error message.

Input Cells	Output Cells	Special Cells
• quiz1	• QuizAvg	• BugRecorder
• quiz2	• ExtraCreditAward	
• quiz3	• TotalPoints	
• quiz4	• InputsTooSmall	
• ExtraCredit	• InputsTooBig	
	• ErrorsInInputs?	
	• LetterGrade	

Purchase Budget

The Purchase Budget spreadsheet computes the total cost of a purchase order, compares that total to the budgeted amount, and checks to see if enough supplies are ordered.

Some cell values in the Purchase Budget spreadsheet are considered to be ``given'', they are neither inputs nor outputs. These cells contain the following information: item names, supplies on hand for each item, minimum supplies required for each item and the total amount budgeted for this purchase order.

Inputs to the spreadsheet are the number of units to order for each item on the purchase order. If a negative number of units are ordered an error message is printed.

The unit price for each item is based on a volume discount:

Pens:	Clips:	Paper:
if less than 50 units are ordered then the cost is $5 per unit otherwise the cost is $2 per unit	if less than 100 units are ordered then the cost is $3 per unit otherwise the cost is $2 per unit	if less than 200 units are ordered then the cost is $6 per unit otherwise the cost is $4 per unit

Given Cells				**Input Cells**		**Special Cells**
Pens	Pens	MinPens		PenUO		BugRecorder
Clips	Clips	MinClips		ClipUO		
Paper	Paper	MinPaper		PaperUO		
BudgetAmt						

Output Cells				
PenUP	PenTotal	ProjPensOH	PenQCheck	UnitsError?
ClipUP	ClipTotal	ProjClipsOH	ClipQCheck	TotalCost
PaperUP	PaperTotal	ProjPaperOH	PaperQCheck	BudgetOK?

Appendix 4: Quick Reference Cards

(WYSIWYT) Forms/3 Quick Reference Card

Record a testing decision and update testedness colors	click on cell's checkbox when it contains a question mark
Red	not tested
Blue	tested
Purple	partially tested
Question mark in checkbox	you can record a decision here
Check mark in checkbox	you have recorded a decision here
Remove most recent testing decision	if the check mark is still on screen, click on the check mark to remove it, otherwise it can't be removed
Change an Input cell's value	1. select the cell 2. click in the edit window 3. make changes 4. click the Accept button
Blank in checkbox	you have already recorded a decision that covers this case
Formula display	click on formula tab 
Formula hide	click on white space at bottom of formula display
Arrows On / Off for a cell	right click on the cell to toggle on/off
Arrows with Detail for a cell	turn Arrows on for the cell and display formulas at sources and destinations of arrows

(Ad Hoc) Forms/3 Quick Reference Card

Record a testing decision	1. select the cell 2. click its checkbox
Edit a cell's formula	1. select the cell 2. click in the edit window 3. make changes 4. click the Accept button
Formula display	click on formula tab 
Formula hide	click on white space at bottom of formula display