AN ABSTRACT OF THE THESIS OF

Christine A. Wallace for the degree of Master of Science in Computer Science presented on August 23, 2001. Title: End-User Assertions in Forms/3: An Empirical Study.

Abstract approved: Redacted for Privacy

Spreadsheets are arguably the most widely used programming language in use today, yet spreadsheets commonly contain errors. Research shows that regardless of the experience of the end user, an alarming number of spreadsheets contain errors (91% in recent field audits). Most spreadsheets are created by end users with little or no programming experience. Unfortunately, software engineering research has largely ignored these users. In an attempt to reduce this high error rate, our research is aimed at bringing the benefits of software engineering to end users without requiring that they first learn software engineering principles. One mechanism for creating error-free programs is assertions. An assertion is a program property that always holds. It provides a way to attach more of the specification to the program. We have developed an assertion tool for spreadsheet languages that extends Microsoft Excel's validation scheme and includes capabilities such as assertion propagation. This work describes an empirical study done to assess how well end users understand and use the information provided by the assertion tool as they perform maintenance tasks. The study also provides information about end users' testing behavior.

End-User Assertions in Forms/3: An Empirical Study

by
Christine A. Wallace

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirement for the
degree of

Master of Science

Presented August 23, 2001
Commencement June 2002

Master of Science thesis of Christine A. Wallace presented on August 23, 2001.

APPROVED:

Redacted for Privacy

Major Professor, representing Computer Science

Redacted for Privacy

Chair of Department of Computer Science

Redacted for Privacy

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for privacy

Christine A. Wallace, Author

ACKNOWLEDGEMENT

This work was supported in part by a NSF ITR grant (#0082265). Graduate student Jay Summet designed and implemented the Lisp side of Forms/3 assertions and provided the team with leadership and entertainment. His work on assertion propagation can be found in his Master's thesis, End-User Assertions: Propagating their Implications. My appreciation goes out to the entire Forms/3 group, past and present. Josh, Dan, and Jay deserve special recognition for putting up with my teasing and for their many helpful comments on the GUI interface. Special thanks to Dr. Burnett, originator of Forms/3, for her endless encouragement and guidance. And, of course, I want to thank my major professor Dr. Curtis R. Cook who always made time for me. He guided both the implementation of assertions in Forms/3 and the empirical study. It was truly my good fortune to end up on his research team.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

LIST OF APPENDIX TABLES

DEDICATION

This thesis is dedicated to the loving memory of my father John A. Lytle, to my mother Ruth E. Lytle, and to my husband James M. Wallace.

END-USER ASSERTIONS IN FORMS/3: AN EMPIRICAL STUDY

1. INTRODUCTION

Spreadsheets are arguably the most widely used programming language in use today. They are easy to use possibly because of their lack of abstractions. Many end users avoid abstraction devices due to the associated learning costs [Green 2000]. But, a novice can quickly pick up spreadsheet basics and begin automating a variety of computational tasks.

Spreadsheets are often created and maintained by end users, people with no formal training in software engineering. These spreadsheets are used in businesses, homes, and governments around the world to order parts, to calculate payroll taxes, for personal record keeping, to compare manufacturing assembly line yields, and to make government policy decisions. Few people's lives are unaffected by the results of corporate and government spreadsheets.

1.1 SPREADSHEET ERRORS AND END-USER SOFTWARE ENGINEERING

The problem is that spreadsheets contain an unexpected and unacceptable number of errors. We know this from the results of both field audits of actual business spreadsheets and from laboratory experiments [Panko 2000]. Field audits since 1997 of 54 spreadsheets found that 91% contained errors. Most of these were classified as "major errors" or "spreadsheets were off by at least 5%." Eighteen laboratory experiments collectively had about a thousand subjects develop spreadsheets from word problems. These word problems had known solutions,

allowing 100% error detection. The different studies used subjects with a wide variety of spreadsheet experience: undergraduates, MBA students with little or no spreadsheet development experience, and MBA students with at least 250 hours of spreadsheet development experience. There was little difference in error rates among these subjects. Across these experiments, 51% of all spreadsheets contained errors, despite the fact that most spreadsheets were only 25 to 50 cells in total size. Cell error rates for whole spreadsheets were at least 1% to 2%. This is in line with the error rates of other human cognitive activities.

Our goal in developing the Forms/3 assertion tool is to assist people, especially those with no software engineering training, in developing more error free spreadsheets. Our assertion tool is designed to act as a layer of abstraction above software engineering practices. Most end users do not first write down an explicit specification document. Most likely, they simply start creating their spreadsheet from the implicit specification in their heads [Nardi 1991]. So whatever information doesn't find its way into a formula is probably lost. Our assertion tool allows an end user to capture such information and make it explicit. It will guide the end user to attach this specification information to the spreadsheet cells and to use the information as an aid in finding and correcting errors.

In this thesis, we describe the assertion tool we developed and an empirical study that investigated whether end users could understand and use the information provided by the assertion tool in a spreadsheet task.

1.2 ASSERTIONS

An assertion on a spreadsheet cell is a property that always holds. For example, a cell that represents area codes must always contain three digit integer numbers. A cell that holds the US Postal abbreviations of states must always contain text with exactly two letters. Likewise, a cell that contains a percentage might be expected to always be a decimal number between 0 and 1 inclusive. There are as many kinds of

assertions as there are properties of cells. We chose to start this research project with just one kind of assertion, a range assertion, and expand from there.

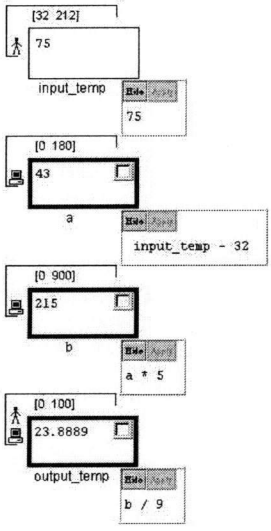
Range assertions are most useful when the task the spreadsheet is designed to solve has restricted input ranges. For example, many input quantities such as weights, lengths, and part counts must be non-negative. Therefore, an assertion on such a cell might be "must be greater than or equal to zero". Some input cells have both lower and upper limits. A class size can be limited to 30 students and obviously cannot be negative. The temperature of liquid water at sea level is between 32 and 212 degrees Fahrenheit. Range assertions are not limited to input cells. An end user may have domain knowledge about an intermediate cell or an output cell. For example, the result of a series of student grade calculations may always fall between 0 and 100. By restricting a cell to contain only certain range(s) of numeric values, we are attaching a piece of the program specification to the spreadsheet.

1.3 A FORMS/3 SPREADSHEET

Let's look at a Forms/3 spreadsheet using range assertions. The spreadsheet in Figure 1b illustrates our scheme of displaying assertions. An assertion is represented in Forms/3 by a guard cell. We call them guard cells or guards because they watch over their assigned cell and alert us with red ovals when the assertion is violated.

This spreadsheet converts a temperature from Fahrenheit to Celsius. The spreadsheet's specification, see Figure 1a, states that only Fahrenheit values between the freezing and boiling points of water are expected in *input_temp*. Therefore, there is a guard cell sitting behind the *input_temp* cell. The guard's stick figure icon tells us that an end user created this guard cell. The string, [32 212], is an abbreviated notation that tells us that *input_temp* is expected to always fall between 32 and 212 inclusive. Look at the guard on the next cell, cell *a*. Forms/3

- The spreadsheet must convert temperatures from Fahrenheit to Celsius.
- The allowable inputs are values between 32 and 212 inclusive.
- The allowable outputs are values between 0 and 100 inclusive.



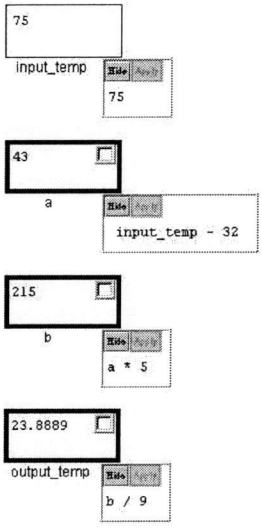


Figure 1a (above): Spreadsheet Specification
Figure 1b (lower left): Forms/3 Spreadsheet with Guards
Figure 1c (lower right): Forms/3 Spreadsheet without Guards

generated this guard as indicated by the computer icon. Forms/3 used *a*'s formula, *input_temp* - 32, as well as *input_temp*'s guard information to generate the range [0 180]. We say that Forms/3 propagated the guard from *input_temp* onto cell *a*. In a similar fashion, Forms/3 propagated guards onto cell *b* and onto *output_temp*. Note that *output_temp*'s guard has both a stick figure and a computer icon. This is because an end user placed a guard onto *output_temp* and Forms/3 propagated a guard onto *output_temp*. In this example, the two guards agree with each other that the expected range of values is [0 100]. If they did not agree, there would be a red oval around the stick figure and computer icons and a red "Conflict" message would appear in place of the range information. There will be more detailed information on Forms/3 implementation of assertions in Chapter 3.

It is important to note that the information contained in an assertion is new information independent of the information contained in the formulas. The restricted range of the input cell is new information not represented by the spreadsheet in Figure 1c. Here there is no information about the range of input, intermediate, or output cells. Thus, the point of the simple spreadsheet in Figure 1b is to demonstrate how information in the spreadsheet's specification that is not contained inside the formulas can be attached through assertions.

1.4 SPECIFICATIONS EXPRESSED AS ASSERTIONS

The value of including more of the specification is two-fold. First, it makes implicit information explicit. Obviously it is rarely beneficial to lose data during a translation. Thus, to drop out information during the translation from specification to spreadsheet creation cannot help achieve a goal of correctness. The original designer may keep those untranslated parts in his head and may always use the spreadsheet as intended. On the other hand, spreadsheets are often modified and/or used by others who might not have the complete picture of the program. Such a person could unwittingly input 30 into a cell to represent 30%, while the spreadsheet was designed to expect 30% represented as 0.30. The result is a value two orders of magnitude too large. The second reason for including as much of the specification as possible is the value of independent verification. A reporter tries to verify a big story through independent channels. A patient in a hospital requests a second opinion. A student adds up a column of numbers forwards and backwards to help assure correctness. These are examples of the value of independent verification. When the independent sources agree with each other we gain more assurance that the result is correct. In the Temperature spreadsheet of Figure 1b, the end user knows that the *output_temp* cell must stay within the freezing point and the boiling point in Celsius. Thus, the user entered guard, [0 100], was placed on the *output_temp* cell. Independent of that, Forms/3 used the cells' formula

information and the *input_temp* guard information to propagate a guard onto *output_temp*. This guard agreed with the user entered guard. The formulas have been double-checked.

Forms/3 is not the first spreadsheet language to implement assertions. Some commercial spreadsheets, such as Microsoft Excel, have cell validation features that allow a user to specify a simple range and detect violations. However, there is no propagation of ranges to other cells. Our work builds upon some of Excel's assertion features and handles others completely differently.

In summary, assertions are a way to attach more of the specification to the spreadsheet. They help guard against erroneous inputs during actual use of the spreadsheet. They can also provide a means of independent confirmation that the spreadsheet is working as intended during the spreadsheet's creation and modification. End users with no software engineering training can use assertions to incorporate more information into their spreadsheets. They may not have a formal problem specification document and the information may be only their own domain knowledge. But if assertions can be made simple and intuitive to use, then end users' spreadsheets can capture that domain knowledge and thereby use it to promote correctness of their spreadsheets.

1.5 OVERVIEW OF THIS WORK

In Chapter 2, we present background information on assertions as used by professional programmers, but we concentrate on how assertions are used by end users in the commercial spreadsheet application Microsoft Excel. We detail how the assertions are created, how violations are signaled, and the limitations of Excel's assertions. In Chapter 3, we describe Forms/3, a spreadsheet-based visual programming language. The Forms/3 testing methodology is detailed as well as its implementation of assertions including how they are created, how violations are signaled, and what we did to solve the limitations of Excel mentioned earlier.

Chapter 4 deals with the design of our protocol analysis experiment. Our experiment results are discussed in Chapter 5. Finally Chapter 6 explores future work and contains our conclusions. Appendices containing our experiment materials and relevant details of each experimental subject's work follows Chapter 6.

2. BACKGROUND

Assertions have been around for years. Some programming languages such as C allow programmers to insert assertions into their code to detect runtime errors. In C, they take the form of *assert* statements. Typically they are used to detect situations that should never occur and that could produce disastrous results. There is ongoing research on how invariants can be generated dynamically from execution traces so that assertions can be automatically inserted into C programs [Ernst 1999]. Assertions may very well aid professional programmers, but our attention is on end users. Specifically, we are trying to help end users create and maintain more correct spreadsheets. Assertions, as used in the C language, are inappropriate for most end users because of their required syntax and complicated Boolean expressions. We want to adapt assertions in a way appropriate to this new audience, one without any software engineering training. In addition, we needed to present assertions in the context of a different programming paradigm, spreadsheet languages.

There have been some attempts at introducing assertions to end-user programmers. For example, Microsoft Excel, a popular commercial spreadsheet application, has data validation, their version of assertions. We examined their implementation and found aspects we wanted to adopt and aspects we wanted to do differently or add.

2.1 EXCEL ASSERTIONS

Excel allows users to specify minimums and/or maximums for integer ranges, decimal ranges, dates, times, and text lengths. In addition, a list of items can form an assertion. The user can program an input message that will appear when the cell is selected. Excel allows the user to select between alert styles and to provide the

error message that will appear in the case of an assertion value violation. In the following sections, we discuss how an assertion is entered into Excel, how Excel signals violations, and some limitations of Excel's assertion mechanism.

2.1.1 Defining an Excel Assertion

To apply an assertion in Excel, first the user must select a cell(s), pull down the "Data" menu, and select "Validation." See Figure 2. Then a pop up window appears. See Figure 3. The user defines or changes types of values, minimums, and/or maximums from the lists of criteria. Excel presents the appropriate boxes to fill in for each data type. In Figure 3, the user has selected "Decimal" numbers "between" 32 and 212.

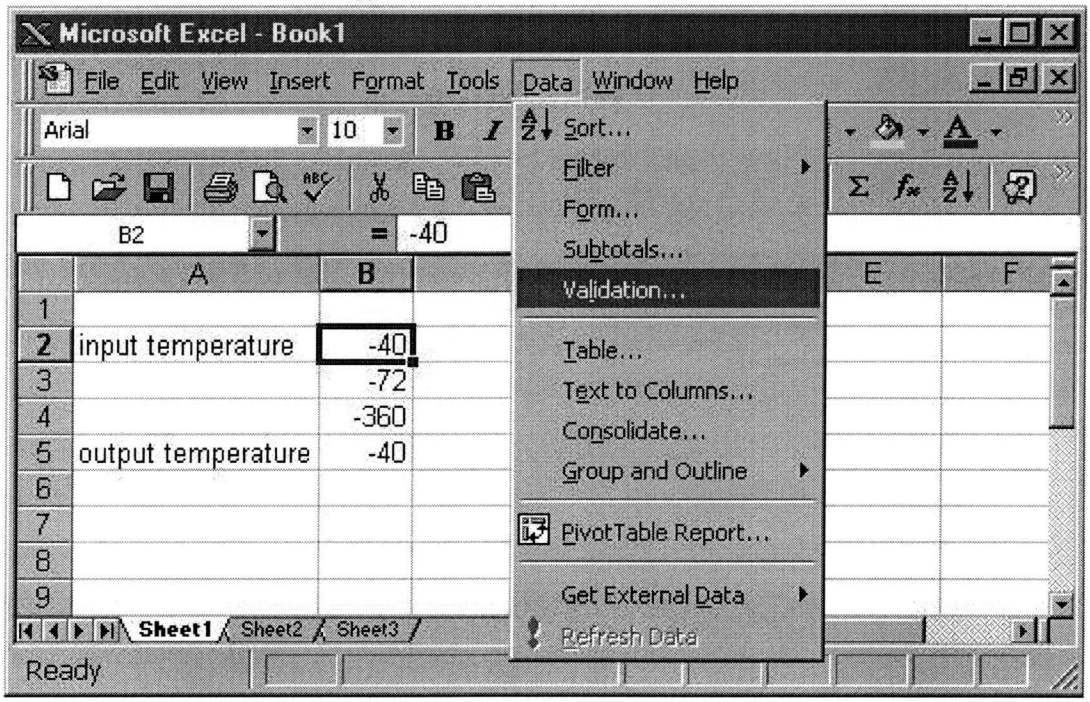


Figure 2: Getting to Excel's Assertion Dialog Box

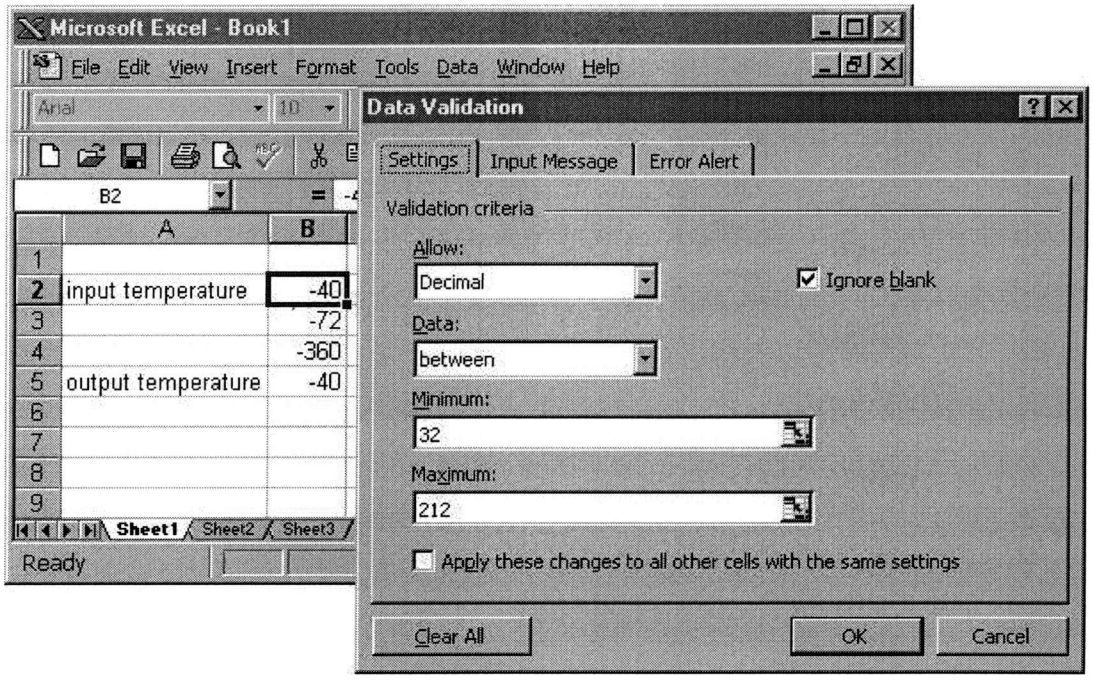


Figure 3: Excel's Assertion Dialog Box

Excel has several choices in the "Allow:" list: Any value, Whole number, Decimal, List, Date, Time, Text length, and Custom. Depending upon the choice, the "Data:" list gives the possible ranges: between, not between, equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to.

The "Input Message" tab allows the user to program a text message such as "Please enter a Fahrenheit temperature between 32 and 212." that will appear whenever the cell is selected. The "Error Alert" tab is where the user may select from the "Stop," "Warning," and "Information" error alert styles. The style of error alert determines what choices the error message presents in the pop up window following an invalid entry. For example, if invalid data is entered into a cell with the "Stop" style, a pop up window with a user programmed text message will appear along with "Retry" and "Cancel" buttons. Continuing on with the invalid data is not an option with the "Stop" style. The other two styles are more lenient. The "Warning" and "Information" styles also allow user programmed text

messages. Unlike the "Stop" style they present the option of keeping the invalid value, the "Information" style by default and the "Warning" style by clicking the "Yes" button.

2.1.2 How Excel Signals Violations

Excel signals invalid values by circling them with a red oval. But, it doesn't display the ovals automatically. In order to see which cells contain invalid values, the user must first bring up the Auditing Toolbar. This is done by going through the "Tools" pull down menu, selecting "Auditing," then choosing "Show Auditing Toolbar." See Figure 4. The Auditing Toolbar will be displayed until the user closes it. The rightmost two buttons on the Auditing Toolbar pertain to assertions.

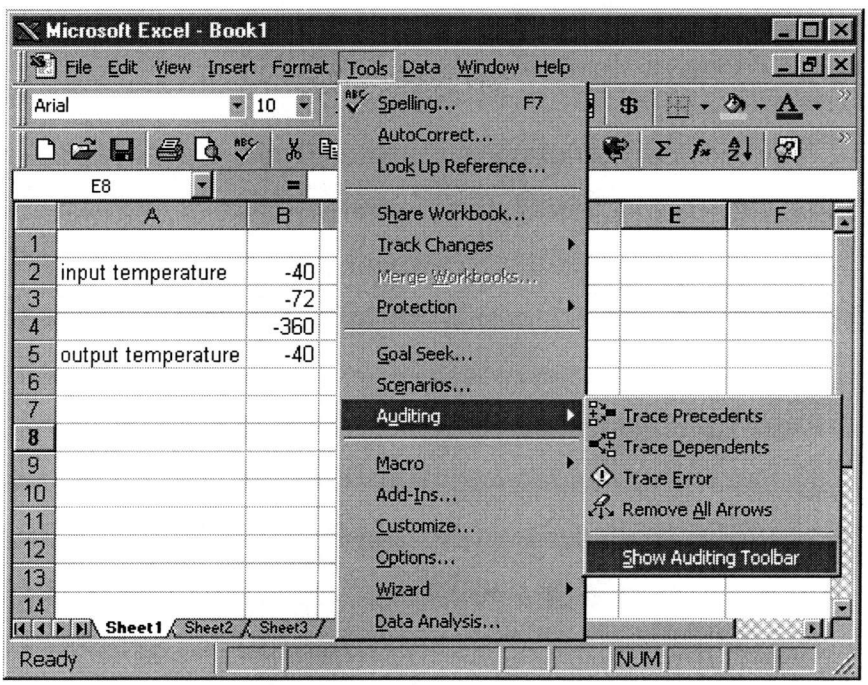


Figure 4: Getting to Excel's Auditing Toolbar

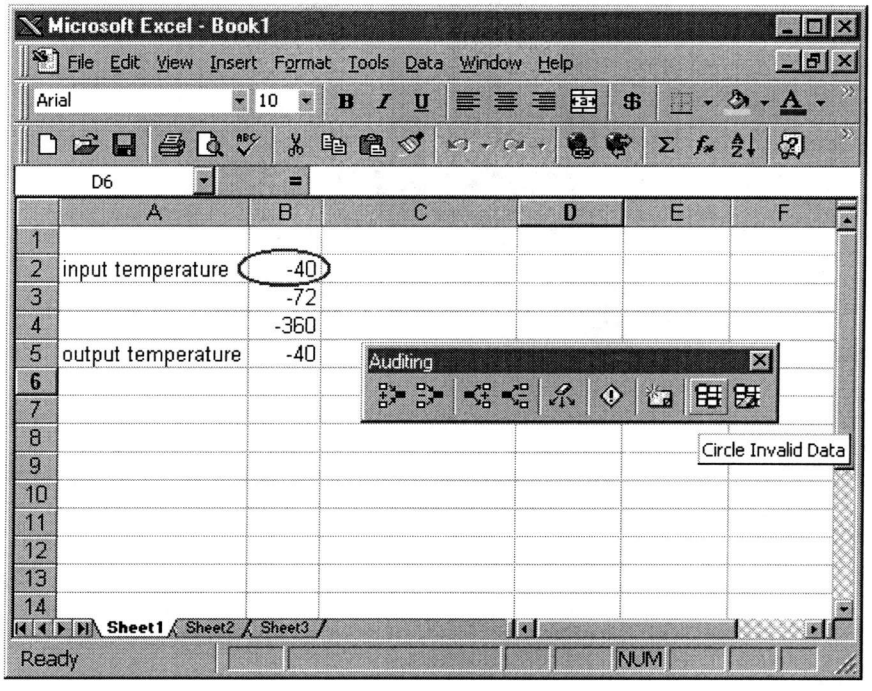


Figure 5: Excel's Auditing Toolbar

See Figure 5. The first of the two is the "Circle Invalid Data" button. The second is the "Clear Validation Circles" button.

From the Auditing Toolbar, the user clicks the "Circle Invalid Data" button. At this point, Excel circles all invalid data and removes any circles from valid data. In Figure 5, cell B2 is circled because -40 is not between 32 and 212, the assertion shown in Figure 3.

2.2 EXCEL ASSERTION LIMITATIONS

To help describe the limitations of Excel's presentation of assertions, we will use some of the Cognitive Dimensions developed by Dr. Green at the University of Leeds [Green 1996]. The Cognitive Dimensions are a broad-brush tool developed

explicitly for non-specialists to access usability. They are "discussion tools" that give names to concepts of usability. Three of the Cognitive Dimensions appropriate to our evaluation of Excel's assertion feature are given in Table 1 below.

Cognitive Dimension	Definition
Visibility	Ability to view components easily.
Juxtaposability	Ability to place any two components side by side.
Consistency	Similar semantics are expressed in similar syntactic forms

Table 1: Three Cognitive Dimensions

2.2.1 Poor Visibility and Juxtaposability

We found Excel to have poor assertion visibility and juxtaposability. A user cannot, for example, look at an Excel spreadsheet and quickly identify all the cells with assertions. A cell that is uncircled and has no input message may still have an assertion. A user must select the cell and go through the "Data," "Validation," and "Settings" sequence described earlier to see the assertion, if any. Many users are accustomed to using Copy and Paste techniques. A normal Paste action also pastes the assertion from the copied cell. Therefore, a user can easily paste assertions all over her spreadsheet without realizing it. It is impossible to view two assertions at once, i.e. poor juxtaposability. Therefore, assertion comparisons are difficult.

2.2.2 Poor Consistency

Excel has poor assertion consistency as well. For example, assertion violations on input cells are handled differently than those on formula cells. When an input

cell assertion is violated, the user gets a pop up window. A formula cell assertion violation does not cause a pop up. This is the case even if a user has gone to the trouble of specifying what message is to appear in the pop up window. Another example of inconsistency is when the user changes the circled data inside an input cell such that it becomes valid, the red oval goes away automatically. But, if the user changes an input that causes the circled value of a formula cell to become valid, the red oval persists on the formula cell. In other words, in a formula cell, valid data may remain circled. But in an input cell, valid data is never circled. Red ovals on valid formula cells will go away if the user clicks on the Auditing Toolbar's "Clear Validation Circles" or "Circle Invalid Data" button.

In general, we didn't like the fact that the user had to ask to see which data was invalid or the fact that this information wasn't "live." In this context, "not live" means that only the data that was invalid at the time of the "Circle Invalid Data" button press is circled. Therefore, the display of red ovals is in sync with the true condition of the spreadsheet only immediately after the button is pressed. After inputs are changed, it is possible that not all circled cells are invalid and not all invalid cells are circled. In other words, after changing input values the display of red ovals may not be in sync with the true condition of the spreadsheet. We found this inconsistent and potentially confusing.

Another source of inconsistency is inherent in the way Excel's assertions are entered. There are three tabs in the Data Validation window. See Figure 3. Only one is visible at a time. There is no consistency check between the three tabs' contents. For example, nothing prevents a user from entering the assertion "Decimal numbers between 32 and 212" in the "Settings" tab, the text "Enter an integer greater than 0." in the "Input Message" tab, and the text "You should have entered a 4 digit number." in the "Error Alert" tab. A user is very unlikely to create an assertion this way. But, on the other hand, it is very easy to change the contents of the "Settings" tab without checking the contents of the other two tab windows, and thus potentially putting the collection in an inconsistent state.

2.2.3 Limited Range Choices

In addition, Excel's method of entering and displaying the assertions is very limited. A user cannot, for example, enter "greater than 0 and less than 100." The closest thing Excel has to that is "between 0 and 100," which includes both endpoints. In Excel, you cannot combine assertions such as "between -1 and 1" and "not equal to 0." A user may consider it one assertion, "between -1 and 1 not including 0." But, Excel considers that two assertions and only one assertion is allowed per cell. Likewise, Excel would not allow multiple subranges such as "between 0 and 1" or "between 100 and 101" to be on the same cell.

2.2.4 Lack of Assertion Propagation

We felt that the most severe limitation of Excel was the lack of assertion propagation. The user must enter each assertion, even if all of the cell's references have assertions. We wanted to generate assertions from the information supplied by the user in the cell formulas and user entered assertions. This way we can inform a user of the possible valid values a formula cell can have. The user can use the propagated assertions as an independent check on his formulas. This and other improvements will be discussed in the next chapter where we describe Forms/3 and our assertion tool.

3. FORMS/3, WYSIWYT, AND OUR ASSERTION TOOL

Forms/3 [Burnett 2001] is a declarative, spreadsheet-based visual programming language under continuous development at Oregon State University. It is primarily used as a research tool to study end-user language and software engineering devices. Users of Forms/3 create cells and define their formulas. These formulas may reference the values of other cells. As in other spreadsheet languages, when a formula is entered, the Forms/3 evaluation engine calculates the cell's value and recalculates the values of other affected cells and displays the results. Thus, the values of all cells are kept up to date.

In this chapter we describe Forms/3's testing methodology, What You See Is What You Test. Then we will walk the user through creating an assertion in the Forms/3 Spreadsheet environment. We will contrast how and when assertion violations are displayed in Forms/3 as opposed to in Excel. We will also show how we addressed Excel's limitations: poor visibility, juxtaposability, consistency, limited ranges, and lack of propagation.

3.1 FORMS/3 TESTING METHODOLOGY

Forms/3's testing methodology, What You See Is What You Test (WYSIWYT) provides feedback about the "testedness" of the spreadsheet as a whole and on individual cells. The WYSIWYT methodology has been designed to work incrementally as formulas are added, deleted, or modified on a spreadsheet. This section gives a brief overview of the WYSIWYT methodology. For a more complete description, see [Rothermel 1998].

3.1.1 Behind the Scenes

Our WYSIWYT methodology operates on the principles of code-based test adequacy criteria. Our test criteria is adapted from the *output-influencing-all-du-pairs* dataflow adequacy criterion originally defined for imperative programs [Duesterwald 1992]. This criterion, which we call du-adequacy for brevity, 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 use (reference) that cell. The criterion requires that each executable du-association be exercised by test data in such a way that the du-association directly or indirectly contributes to the display of a value that is subsequently pronounced correct by the end user. In the next section, we will discuss how Forms/3 communicates "testedness" to the user and how the user communicates testing decisions to Forms/3.

3.1.2 Testing Tools

The WYSIWYT methodology communicates testedness information in two ways: cell border colors and a "percent tested" indicator. Forms/3 uses colored borders to communicate testedness of individual cells to the user. Red means untested, shades of purple mean partially tested, and blue means fully tested. There is also a "percent tested" indicator at the top right of the spreadsheet to display information about the spreadsheet as a whole. See Figure 6. The percentage indicates the portion of the total du-associations in the spreadsheet that have been tested. In this paper, blue borders appear as thick black lines, red borders are thick gray lines. When a formula cell is created or its formula modified, the cell border turns red because the cell is untested. In addition, cells that reference the modified cell will also turn red. This communicates to the user the impact of changing a formula on the rest of the spreadsheet.

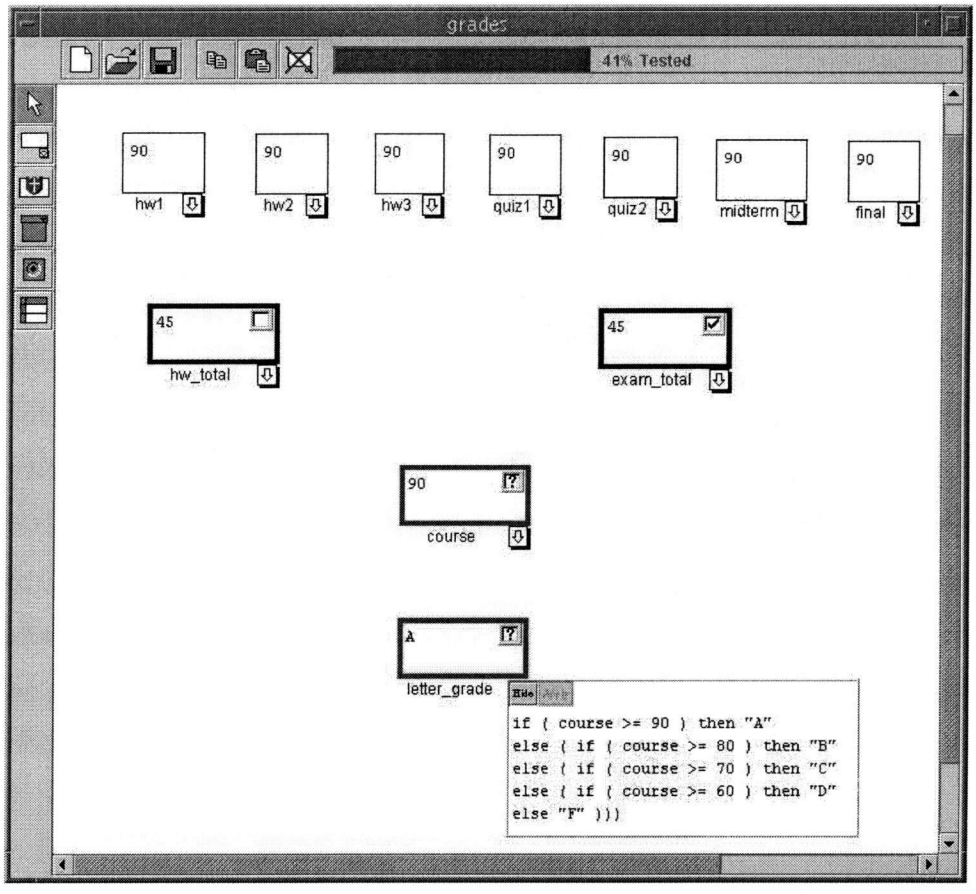


Figure 6: Testing the Grades Spreadsheet

The user tests a cell by deciding if its value is correct for the spreadsheet's inputs. He then communicates this decision to Forms/3 by way of the cell's decision box located in the upper right corner of the cell. A left click in this box tells Forms/3 that the cell's value is correct. A right click tells Forms/3 that the cell's value is wrong for the spreadsheet's inputs. Forms/3 allows testing on any formula cell at any time, but it also gives advice to the user concerning whether or not testedness progress can be made. For example, if the present test case is similar to a prior test case, Forms/3 may determine, based on the testing adequacy criteria, that the spreadsheet's testedness will not increase by using this test case. In this case, Forms/3 would present a blank decision box. The user may still make a decision

and click in the decision box, but progress will not be made toward a 100% tested spreadsheet. On the other hand, if the present test case can provide progress (e.g. test previously untested du-associations), then Forms/3 presents a decision box with a question mark inside. The question mark is advice from Forms/3 to make a decision here.

When a user left clicks on a decision box with a question mark, the percent tested indicator increases, the question mark is replaced with a check mark, and the cell's border turns more blue. Recall, that red means that the cell is untested. A blue border means that the cell has been fully tested. The testing adequacy criteria determine the minimum number of test cases (du-associations) required to turn a cell blue. For example, many formula cells require just a single test case to be fully tested, while a cell with an "if" expression will require more than one. If more than one test case is required, the cell border changes from red, to a shade(s) of purple, to blue on its way to being fully tested. Purple is considered "more blue" than red, i.e. more tested.

Consider the spreadsheet of Figure 6. Suppose a user decided that cell *hw_total* was correct when the inputs were all 80. Now the inputs are all 90. Cell *hw_total* has a blue border (black in this paper) because it has been fully tested. Its decision box is blank because the new test case is similar, by the testing adequacy criteria, to the old test case of all 80's. Cell *exam_total* has just been tested and found to be correct as indicated by the check mark in its decision box. Its border is also blue. However, the borders of cells *course* and *letter_grade* are still red (gray) and there are question marks in their decision boxes. If the user determines that either of these cells is also correct, progress will be made in the spreadsheet's testedness. The formula of *letter_grade* is displayed to show a multi-situation "if" expression. There is a du-association for each of these situations (letter grades). Therefore, it will take a minimum of five test cases to fully test the *letter_grade* cell. Its border will go from red, through four shades of purple, and finally to blue as testing progress is made.

3.2 DEFINING A FORMS/3 ASSERTION

To put an assertion onto a cell in Forms/3, the user selects the Create Guard Cell icon located on the left side of the spreadsheet and then clicks on the cell. A window with a number line appears. See Figure 7. Above and to the right are three tools: a "Selection" tool (arrow), a "Point" tool (filled circle), and a "Range" tool (horizontal line). The following example is the same one used in section 2.1.1 to describe how to define an assertion in Excel. To place the assertion "between 32 and 212," the user clicks on the number line to place a point. The "Point" tool is selected by default. A point will appear on the number line and the user can label it "32" in the field below. Similarly, a second point is added to the right and labeled "212." Next the "Range" tool is selected. When the user clicks between his two points, a thick black line is drawn to indicate all the in-between points as shown in Figure 7. Now the user can click "Apply My Guard" and the assertion takes the form of a guard cell. We will revisit the Forms/3 assertion number line in section 3.4.3 when we discuss the definition of complex ranges.

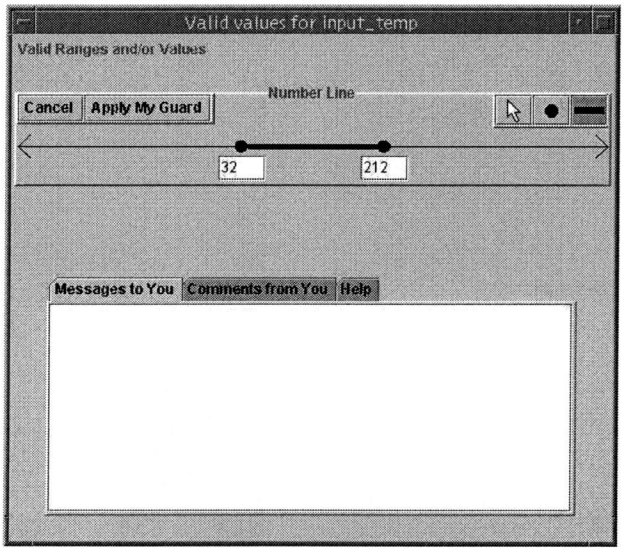


Figure 7: The Forms/3 Assertion Number Line Window

3.3 HOW FORMS/3 SIGNALS VIOLATIONS

The user never has to ask to see which data is invalid in Forms/3. Like Excel, invalid data are circled with red ovals. However, unlike Excel, adding and removing violation ovals is always automatic on all cells. See Figure 8. The *input_temp* cell on the Temperature spreadsheet has a circled value. Its value, -40, is not within the 32 to 212 range. We call this a value violation to distinguish it from an assertion conflict. Assertion conflicts are defined in the Propagation section, 3.4.4.

3.4 FORMS/3 ADDRESSES EXCEL LIMITATIONS

3.4.1 Visibility and Juxtaposability

A quick glance at a Forms/3 spreadsheet reveals which cells have assertions (guards). Figure 8 shows the guard placed on *input_temp* by the user's actions described earlier. Guard cells sit behind formula and input cells. Inside a guard cell, there is space for a textual representation of all or part of the range. The range 32 to 212 inclusive is represented as [32 212]. When an endpoint is not included, the range is written with a parenthesis, such as (0 10], [25 50), or (-1 1). The user

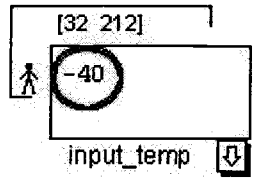


Figure 8: *input_temp* Cell of Temperature Spreadsheet with a Guard

has placed a guard on the *input_temp* cell. This guard contains a small stick figure to indicate that it was user entered. A system generated guard, also called a propagated guard, has a computer icon. There will be a complete discussion of these guards in the Propagation section.

The guards, if present, are always visible. The user can easily tell which cells are "guarded" and what the valid values are. This is how we solved Excel's poor visibility problem. The user can return to the number line representation on any guard simply by double clicking on the guard to open it. From there they can make changes or just view the ranges in the number line representation. The number line windows from two guards may be placed side by side for easy comparison, unlike in Excel. This is how we handled the juxtaposability issue.

3.4.2 Consistency

The display of red ovals is always in sync with the true condition of the spreadsheet. In Forms/3, invalid data is always circled and valid data is never circled. Recall that this is unlike the way Excel works. In Excel, only input cells have automatic red oval removal. Formula cells in Excel may contain valid, yet circled values. And either type of cell in Excel can have uncircled, yet invalid, data.

We chose not to use pop up windows at all for value violations. Recall that Excel issues a pop up window when an input cell has a value violation, but not when a formula cell does. Instead, we immediately circle the invalid data on either type of cell, input or formula. We feel that the pop up windows are too distracting and don't fit into our emphasis on Attention Economics [Blackwell 1999]. Attention Economics analyzes the value of an action, its cost in attention, and the associated risk if the action is not done. For example, while typing this thesis, my main task is writing and formatting. But periodically I stop progress toward that goal and divert my attention to clicking the "Save Document" button. I do this because the value of this action (a saved file) exceeds the cost (diverting my

attention for a moment) and the risk (having to retype several pages in case of a program crash) is too high. We thought that the cost of diverting a user's attention from their main task is too high with a pop up window for the value of making sure the user sees the violation. This is especially true when the violation disappears as the user makes more changes to the spreadsheet.

3.4.3 Complex Range(s), Multiple Ranges

A Forms/3 assertion can contain an arbitrary number of ranges. While the user cannot express assertions such as "between -1 and 1" and "not equal to 0" in Excel, they are easy to represent in Forms/3. Figure 9 shows a number line displaying the ranges [-1 0) (0 1]. The hollow point labeled 0 denotes that it is not included in the -1 to 1 range. A point toggles from filled to hollow (included to not included) when the Point tool is selected and the point's circle is clicked.

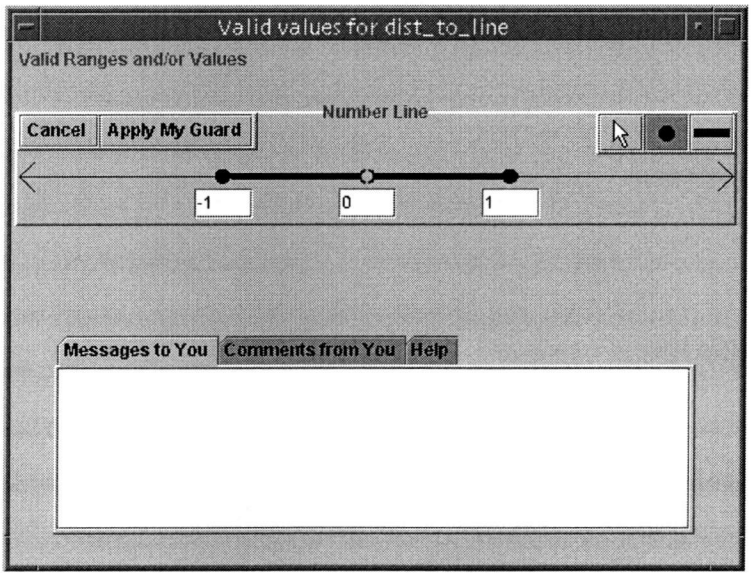


Figure 9: How to Enter the Assertion "between -1 and 1, but not 0" in Forms/3

With the Point tool selected, the user simply clicks on the number line to place as many points as desired. They may be left as discrete points or ranges may be added between them. If a point is added inside a range, it will automatically appear as a non-included (hollow) point. If a point is added outside a range, it will automatically appear as filled. Once placed, points may be grabbed and slid along the number line to make more room. If they are endpoints to a range, the range automatically resizes itself accordingly. Points and/or ranges may be deleted. Error checking is done to make sure that the user has given valid labels to all the points. For example, the label must be a number. The number must be less than all numbers to the right and greater than all numbers to the left. The tabbed window below the number line is a multifunctional area. Error messages such as "That was not a number." appear in the "Messages to You" tab area. Users may document their assertions in the "Comments from You" area. The "Help" tab reminds users how to use the number line. The number line was intended to simulate the number line concepts learned in grade school. With it, our assertion ranges are much more extensive than Excel's.

3.4.4 Propagation

Earlier in chapter 1 we discussed the propagation of guards. Propagated guards are the ones with computer icons. See Figure 10 which contains the entire Temperature spreadsheet. Forms/3 has used the user entered assertion on *input_temp*, combined it with the other cells' formulas, and propagated three system assertions. For example, the formula for cell *a* is "*input_temp* - 32." Since the range on *input_temp* is [32 212], Forms/3 has put a system generated guard onto cell *a* with the range [0 180]. Propagated guards will automatically update their ranges if formulas or user entered guards are changed. User entered guards are not modified by the system as they belong solely to the user.

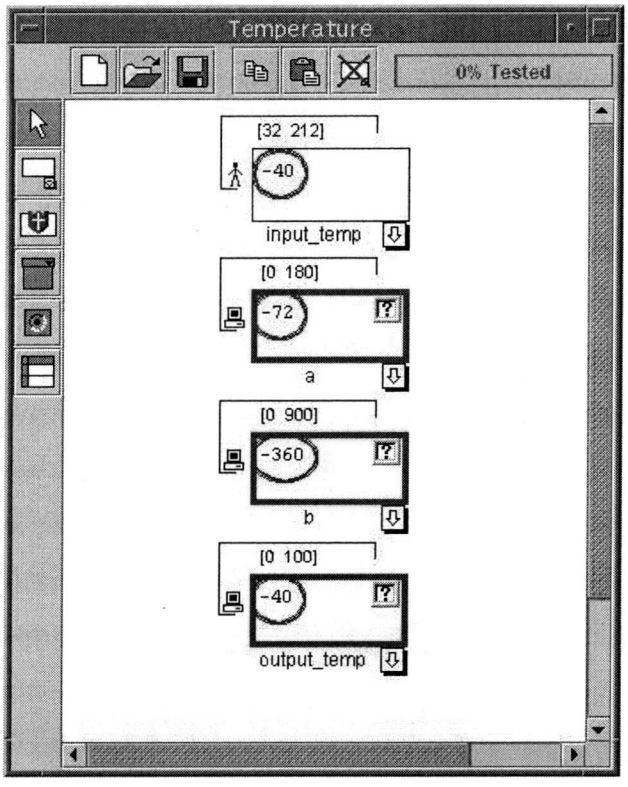


Figure 10: The Entire Forms/3 Temperature Spreadsheet with Guards

When a user opens a system generated guard, they see two number lines: one is theirs and one is Forms/3's. See Figure 11. The user has opened *output_temp*'s guard. The lower number line shows the system generated range on this cell of [0 100] which makes sense since this cell represents the Celsius values. The upper number line belongs to the user. If the user has never indicated a range assertion on this cell, it starts out containing the Forms/3 assertion. The user may keep this information or make changes. If the user clicks "Apply My Guard," they are applying only what is on their own number line. A stick figure will join the computer icon to indicate that both system generated and user entered guards are present. When the two assertions are identical, the guards are said to "agree" and the agreed upon range is displayed in the guard. See Figure 12a. When they are not identical, the guards "disagree" or have an assertion conflict. In this case, the stick

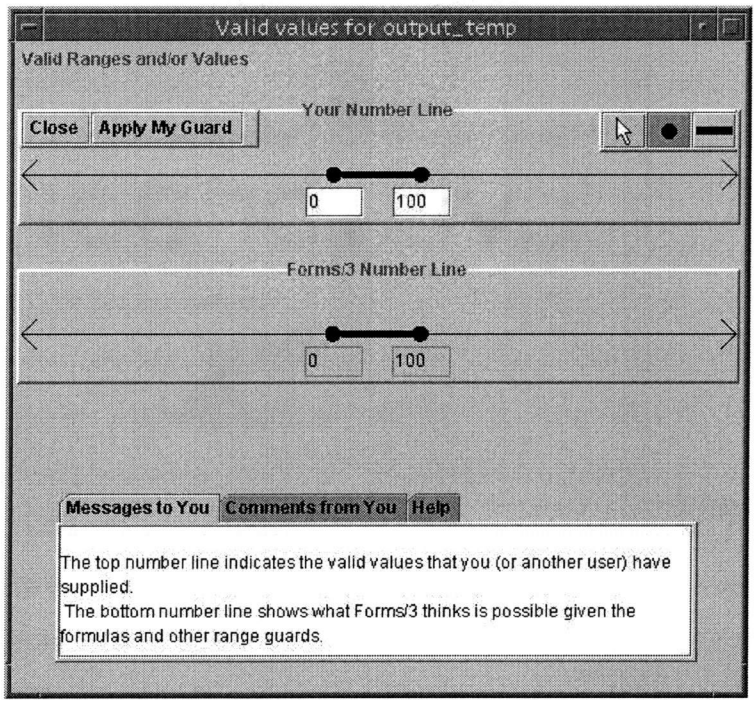


Figure 11: The User's Number Line and the Forms/3 Number Line

figure and the computer are circled in red and a "Conflict" message replaces the range information. See Figure 12b.

To clarify the difference between a value violation and an assertion conflict, note that the left version of *output_temp* in Figure 12a has a value violation because -40 is outside the range of [0 100]. It does not have an assertion conflict since the two guards agree on the range of between 0 and 100. The right version in Figure

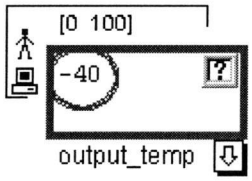


Figure 12a: No Assertion Conflict

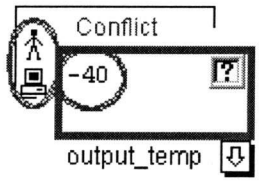


Figure 12b: An Assertion Conflict

12b has both a value violation and an assertion conflict. To see the details of the disagreement, the user would double click on the guard to open it. The two number lines would reveal the nature of the conflict. It might, for example, reveal that the user expects this cell to range from 0 to 100. While Forms/3, due to an error in a formula, declares the range to be from 0 to 324. This would actually happen if the formula multiplied by 9/5 rather than by 5/9.

Because Excel has no propagation, there can be no disagreement between a system and a user guard to point out possible formula errors. Similarly there can be no independent verification when the guards agree. But in Forms/3 a user can place an assertion on input and output cells and Forms/3 will compute its own range on the output cell. If they don't agree, as in the cell of Figure 12b, the user has strong evidence that a formula is wrong. If they do agree, the user receives reassurance that the formulas are correct.

4. EXPERIMENT DESIGN

The objectives of our study were to investigate the following hypotheses:

H1: End users can interpret the guard feedback correctly.

H2: End users can understand guard propagation.

H3: End users are not overly distracted by the assertion conflicts or by the value violations.

H4: End users with guards will modify spreadsheets more correctly than those without guards.

H5: End users with guards will test their spreadsheets as thoroughly as those without guards.

We decided on using a protocol analysis technique because of the nature of the above hypotheses. Protocol analysis allowed us to get at what the subjects were thinking at different points, what their goals were, and what their strategies were. Getting this data was critical to our understanding of what and especially why the subjects did what they did.

We could have studied whether or not end users would add guards to a spreadsheet that they were creating. But, this being the first empirical study on assertions, we chose to instead investigate how end users worked with and understood Forms/3 guards on existing spreadsheets. If we found out that our subjects didn't understand guards and couldn't work with existing guards in a meaningful way, it would indicate that we had to fix a fundamental problem in our tool before proceeding.

4.1 PROCEDURE

To investigate our hypotheses, we conducted a protocol analysis on ten subjects. Each of the subjects modified two spreadsheets, a temperature conversion

and a grades spreadsheet. Five of the subjects used the WYSIWYT methodology ("Control" group). The other five used the WYSIWYT methodology plus guards ("Guard" group).

The experiment was conducted one subject at a time on a Sun workstation. Each subject actively participated in a tutorial by modifying a practice spreadsheet. The investigator would describe Forms/3 features and the subject would perform the subtask described. Since the investigator was working one-on-one, the tutorial could be paced for the individual. The overall goal was that the subject understand the material. There is further information about the tutorial in section 4.3.

Subject data was collected via electronic transcripts and audio recordings of the user during the two modification tasks. In addition, all students filled out a background questionnaire to determine if the two groups were basically homogenous. The Guard group filled out a post session questionnaire that contained questions about the guards. Due to the small number of subjects, we were not looking for statistical significance. We wanted to spot trends.

4.2 SUBJECTS

Our hypotheses involved end users not programmers. Therefore, we decided to draw our subjects from a pool of business school students at Oregon State University. They would be representative of an end user with no software engineering training, yet have experience in modifying spreadsheet formulas. All of our subjects were enrolled in a sophomore business course in which they worked with spreadsheets. They were randomly divided into the two groups.

All the background data was self reported and collected on the background questionnaire. See Appendix A for the questionnaire. The two groups were basically homogenous with a slightly higher GPA and level of programming experience in the Control group. The differences were not statistically significant. Below is a table that summarizes their backgrounds.

	Control Group	Guard Group
Count	5	5
College Major	All Business majors	All Business majors
Year in College	2 Sophomores 2 Juniors 1 Post -Bac	3 Sophomores 2 Juniors
GPA	3.43 (0.43)	3.19 (0.34)
College Math (through college Algebra)	All	All
Programming Experience	1 A college C class 2 A high school class 2 None	1 A college C++ class 4 None
Spreadsheet Experience	All	All
Forms/3 Experience	None	None
Years Speaking English	4 English as a 1st language 1 With 10 years experience	4 English as a 1st language 1 With 5 years experience

Table 2: Subjects' Background Summary

4.3 THE TUTORIAL

The format of the tutorial was that the investigator would describe a Forms/3 feature and the subject would then use the feature on the practice spreadsheet. The tutorial introduced basic Forms/3 skills such as creating, naming, moving, editing, and deleting cells. The investigator was watching and assessing the subject's progress and understanding at all times. Material was repeated if the subject hadn't mastered the skill.

The tutorial also included basic instruction on how to use the WYSIWYT tools. This included colored borders, decision boxes, and the "percent tested" indicator. We made no reference to du-associations as one of the goals of our testing methodology is that the end user does not require formal software testing training. We said only that red means "untested," purple means "partially tested," and blue means "fully tested."

Testing a cell was described as the process of recording in the decision box whether or not the cell's value was correct for the spreadsheet's inputs. In the tutorial, the subjects tested cells named *total_sum* and *ave_milk_intake*. They were told the meanings or functions of the cells without opening their formula boxes. They were asked to test each one before they opened the formula boxes. This is important because we didn't want to teach them that testing was "looking at the formula and plugging in numbers." One of the cells was correct and one was incorrect to give them experience with recording each type of decision. The subjects also experienced testing cells with "if" expressions which required multiple decisions to become fully tested.

The Guard group received additional training on Forms/3 guards. The Control group, of course, did not. The Guard subjects learned how to create, open, and modify guards. They added a guard onto an input cell and saw that Forms/3 immediately propagated its own system guards. As they modified the practice spreadsheet, it was pointed out how the system generated guards updated themselves automatically. The Guard subjects experienced and resolved both a value violation and an assertion conflict.

The two groups used the same examples on the tutorial except for the addition of the guards. Since the tutorials were individually monitored by the investigator, we could determine when each subject had mastered each skill. Tutorial material was repeated as necessary. At the conclusion of the tutorial, the subjects had equivalent Forms/3 skills.

4.4 TASKS AND MATERIALS

Each subject was given the same two tasks in the same order: the Temperature Conversion task first followed by the Grades task. For each task, the subjects received a problem description handout. The exact handouts are in Appendices B – D. They worked as long as needed and indicated when they were finished.

4.4.1 Temperature Conversion Task

The subjects were given a working and fully tested (all cells were blue) Forms/3 spreadsheet that converted a temperature between 32 and 212 in degrees

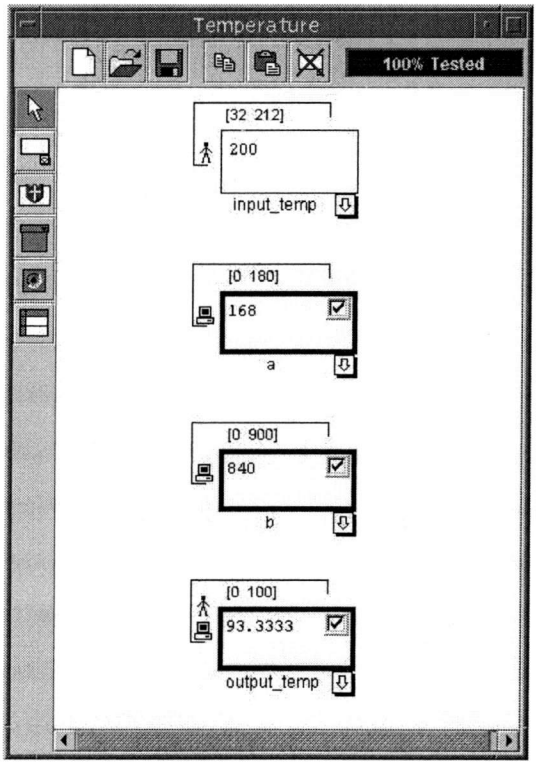


Figure 13: Temperature Spreadsheet as First Seen by the Guard Subjects

Fahrenheit to degrees Celsius. The Guard subjects received the spreadsheet shown in Figure 13. The Control subjects' spreadsheet contained no guards. The task for both groups, as described on their problem description handout, was to modify the spreadsheet so that it converted a temperature between 0 and 100 in degrees Celsius to degrees Fahrenheit with the answer appearing in the *output_temp* cell.

The Temperature Conversion task was designed to see how our Guard subjects dealt with value violations. Figure 13 shows the original Temperature spreadsheet given to the Guard group. The value 200 is in the *input_temp* cell. Since this cell represents Fahrenheit values, its guard has the range [32 212]. During the task, though, the subjects would see a value violation if they edited the guard to reflect the new range of [0 100] before they lowered the value of *input_temp*. Then we could observe their reactions to the value violation.

The Temperature Conversion task also provided an opportunity to observe our Guard subjects deal with an assertion conflict. With the exception of changing the value of *input_temp*, any formula or guard edit causes an assertion conflict on *output_temp*'s guard. Therefore, it was likely that our Guard subjects would see a value violation and an assertion conflict at the same time immediately after they edited the guard on *input_temp*. The user entered guard on *output_temp* had to be changed from [0 100] to [32 212] in the course of the Guard subjects' modifications. Therefore, the conflict on *output_temp*'s guard did not go away automatically even if the subjects completed their modifications correctly. This is not the case with all spreadsheet modifications as the Grades task will demonstrate.

4.4.2 Grades Task

The subjects were given a working and fully tested (all cells were blue) Forms/3 spreadsheet that calculated a student's course grade. See Figure 14 for the Guard version. The problem description listed the homework and exam spreadsheet inputs as having the course weights shown in Table 3.

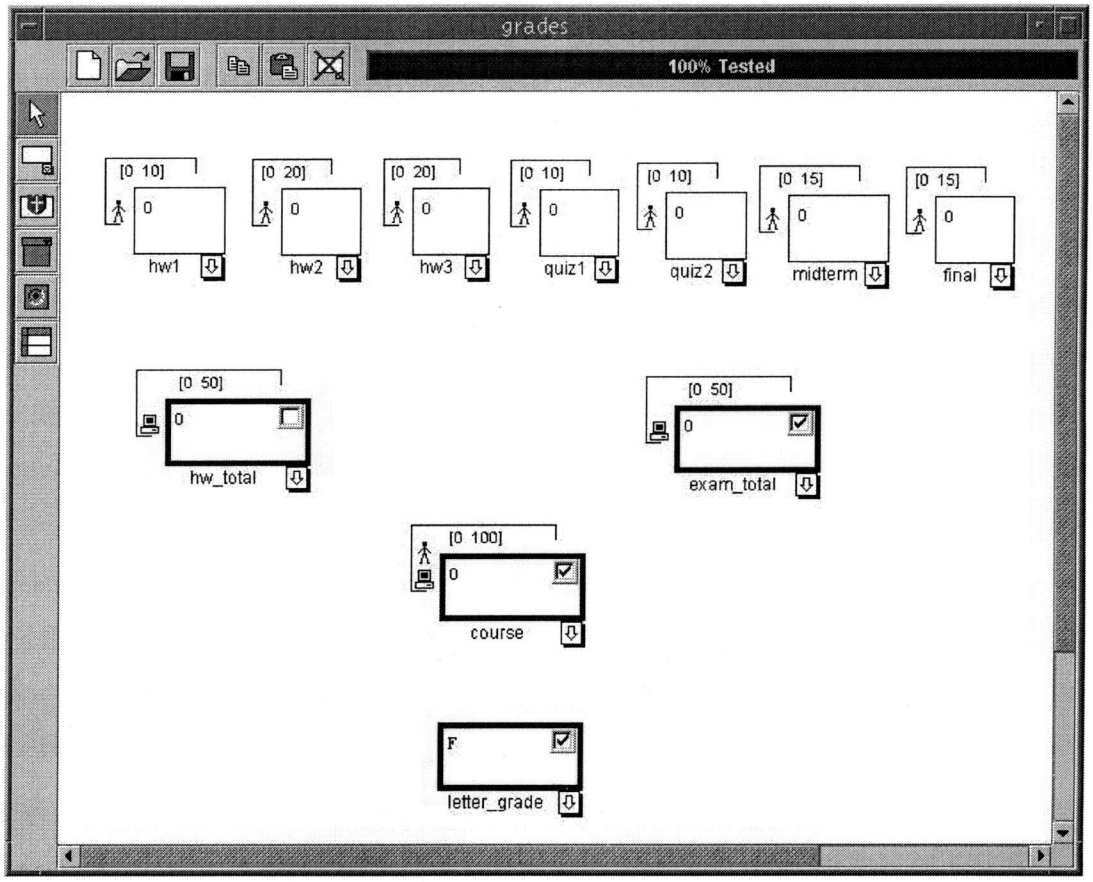


Figure 14: Grades Spreadsheet as First Seen by the Guard Subjects

Input Cell	Course Weighting	Original Point Values
hw1	10%	worth 0 - 10 points
hw2	20%	worth 0 - 20 points
hw3	20%	worth 0 - 20 points
quiz1	10%	worth 0 - 10 points
quiz2	10%	worth 0 - 10 points
midterm	15%	worth 0 - 15 points
final	15%	worth 0 - 15 points

Table 3: Homework and Exam Weighting in Grades Problem

In the original spreadsheet, all the formulas were simple sums. The *hw_total* cell summed the homework scores, *exam_total* summed all quizzes and tests, and *course* computed the total sum. All the subjects were told in the problem description that this resulted in *hw_total* ranging from 0 to 50, *exam_total* ranging from 0 to 50, and *course* ranging from 0 to 100. The letter grade criteria was also listed. For example, a *course* value between 90 and 100 resulted in an "A," 80 to 89 a "B," ...

The subjects were told to change the spreadsheet so that all homeworks, quizzes, and tests were each worth between 0 and 100 points. But the course weighting for each of these had to remain the same. For example, the final still had to be 15% of the course score. They were told that *hw_total* and *exam_total* must continue to range from 0 to 50 and that *course* must continue to range from 0 to 100. The letter grade criteria was to remain unchanged. Because the pilot subjects in our study had difficulty with this problem, we provided an extra handout that described two scenarios of three assignments with unequal course weighting. In the first scenario, the assignments were worth 25, 30, and 45 points respectively. The course grade was calculated by simply adding up the scores. In the second scenario, all three assignments were each worth 100 points. This scenario showed course = 0.25 * Assignment 1 + 0.30 * Assignment 2 + 0.45 * Assignment 3.

We didn't want lack of basic algebra knowledge to interfere with the completion of this task. The subjects were shown, in the extra handout described above, how to do the calculations on paper on a simpler problem. They still had to translate that knowledge into the Forms/3 spreadsheet language and use it on a more difficult problem.

The Grades task was designed to cause an assertion conflict as the Guard subjects began their modification. Unlike the Temperature Conversion task, however, the assertion conflict would automatically disappear if the subject completed the task correctly. The reason is that the range of the user entered guard on the *course* cell should remain at [0 100]. During the modification, the system generated guard will deviate from that range, but will return if the modification is

done correctly. This modification task let us observe the Guard subjects react to the appearance and disappearance of assertion violations.

4.4.3 Post Session Questionnaire

Only the Guard subjects were given the post session questionnaire. A copy is in Appendix E. The questions were designed to assess their knowledge of guards: their graphical representation, propagation, value violations, and assertion conflicts.

5. RESULTS

In this chapter, we refer to the five Control group subjects as C1 - C5, and the five Guard subjects as G1 - G5. Appendices F through O contain the data taken on each individual subject. The first five sections of this chapter give our results for the five hypotheses. Following that, we summarize other findings such as input selection and testing behaviors.

5.1 HYPOTHESIS 1 RESULTS

H1: End users can interpret the guard feedback correctly.

Guard feedback comprises all the guard graphics that the end user sees on the screen. This includes the red value violation ovals, red assertion conflict ovals, the stick figure and computer icons, and the abbreviated range information. We wanted to make sure that the end users could interpret the correct meanings behind these graphics. This was measured two ways: the post session questionnaire, see Appendix E for the complete questionnaire, and observations while the subjects worked on the two problems.

5.1.1 Post Session Questionnaire

These questions refer to Figure 15:

Q1: What does the red oval on *cellA* mean?

Ans. - The value falls outside the range.

Q2: What does the little stick figure in the *cellA* guard mean?

Ans. - The guard was supplied by the user.

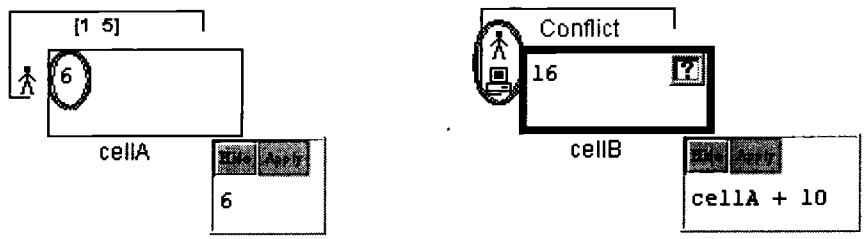


Figure 15: Post Session Questionnaire Graphic

Q3: Why are a stick figure and a computer on *cellB*'s guard?

Ans. - The guard was supplied by both Forms/3 and by the user.

Q4: What does the red oval on the *cellB* guard mean?

Ans. - The user and Forms/3 disagree on the valid range(s) for this cell.

Q6: What can be done to get rid of the red oval on *cellA*?

Ans. - Change the value of *cellA* to be within the range 1 to 5 inclusive.

All five Guard subjects correctly answered all of these questions. Therefore, our subjects demonstrated that they could correctly interpret value violation ovals and assertion conflict ovals and that they understood the meaning of the source icons, the stick figure and the computer. Through question 6, we also know that they could read and understand the abbreviated range information displayed on the unopened guard and knew how to resolve a value violation.

5.1.2 Observations While Subjects Performed Tasks

Our observations of the subjects told us the same thing. The subjects had no trouble understanding the guard's graphical information. They demonstrated in the Temperature problem that they understood a value violation oval and knew how the

resolve it. They also reacted appropriately to assertion conflict ovals in both problems, frequently saying something like, "the ranges disagree." Then, typically, they would open up the guard to see the conflicting ranges.

Only one incident came up that pointed to some confusion in the graphical feedback. This incident involved a combination of guard and testing feedback that confused one subject, G5. The *exam_total* cell was blue because the user had previously tested it. But the value inside the cell was circled in red because it was -7. Recall that the range assertion on *exam_total* was [0 50]. This combination of feedback confused our subject. He said, "The cell's value is wrong, kind of. I mean, it's right, but it's wrong, I guess." At the same time he was checking the decision box, undoing the check, X'ing the decision box, undoing the X. He added, "But why is it a blue cell? That's kind of weird." This subject was the only one to encounter this combination of feedback because he was the only Guard subject to enter negative values into the input cells. This subject's reaction is discussed further in the Future Work section of Chapter 6 and in Appendix O.

5.2 HYPOTHESIS 2 RESULTS

H2: End users can understand guard propagation.

Our subjects were evaluated three ways with respect to guard propagation. They were asked to propagate a guard themselves, to resolve an assertion conflict, and to predict the effects of removing a user entered guard.

5.2.1 Propagate a Guard

First, they were asked in question 5 of the post session questionnaire to propagate a guard themselves onto *cellB* of Figure 15. Four out of five subjects

used the formula and the user entered guard to correctly calculate the answer, [11 15]. The remaining subject used the formula and the user entered guard, but forgot to add 10 to the lower limit. This subject quickly corrected his mistake when questioned. Clearly, they all understood the basic propagation algorithm.

5.2.2 Resolve an Assertion Conflict

Second our subjects were asked in question 7 of the post session questionnaire to resolve the assertion conflict on *cellB* of Figure 15. The results of this question were less conclusive. Two subjects correctly said that the formula must be modified. One of them offered the correct formula. The other three subjects had different initial reactions. One wanted to open the guard and modify the Forms/3 number line to agree with the correct user number line. Note that the subjects were answering this question on paper, not actually using Forms/3 to solve the problem. Thus, the subject wanting to change the Forms/3 number line did not have the benefit of viewing the Forms/3 number line. She couldn't see the grayed out text fields of the Forms/3 number line that beep when a user clicks on them. When she was reminded that she could not directly modify the Forms/3 number line, she decided that the formula must be modified. The remaining two subjects needed to be reminded to think about the process they themselves used to propagate a guard. They eventually arrived at changing the formula. One of these two subjects, G2, encountered a very similar situation during the Temperature task. He opened a guard with an assertion conflict, recognized that the Forms/3 number line was incorrect, and immediately said aloud that a formula must be wrong. Indeed, one of his formulas was in error. Therefore, this individual responded appropriately to the conflict information during the modification task, but not during the test on paper. For this reason and for the Forms/3 number line feedback mentioned above, I believe question 7 should have been given "on-line" rather than on paper. The subjects may have figured out more on their own in a more realistic situation.

5.2.3 Predict the Effects of Guard Removal

Third, the subjects were asked to predict or explain what would happen to the guard on the *exam_total* cell if the guard on the *midterm* cell were removed. Note in Figure 16 that *exam_total*'s guard is propagated in part from the guard on the *midterm* cell. Also note that this was the first time the subjects had ever removed a guard so they had no experience with this scenario. Three out of five subjects made a correct prediction. They varied in detail. One didn't think Forms/3 could calculate *exam_total*'s guard. One thought that maybe it would say "Undefined" on *exam_total*'s guard. One said that maybe *exam_total*'s guard would "go away." The other two subjects didn't predict what would happen, but could explain it once they saw *exam_total*'s guard disappear. Our conclusion is that no one was surprised

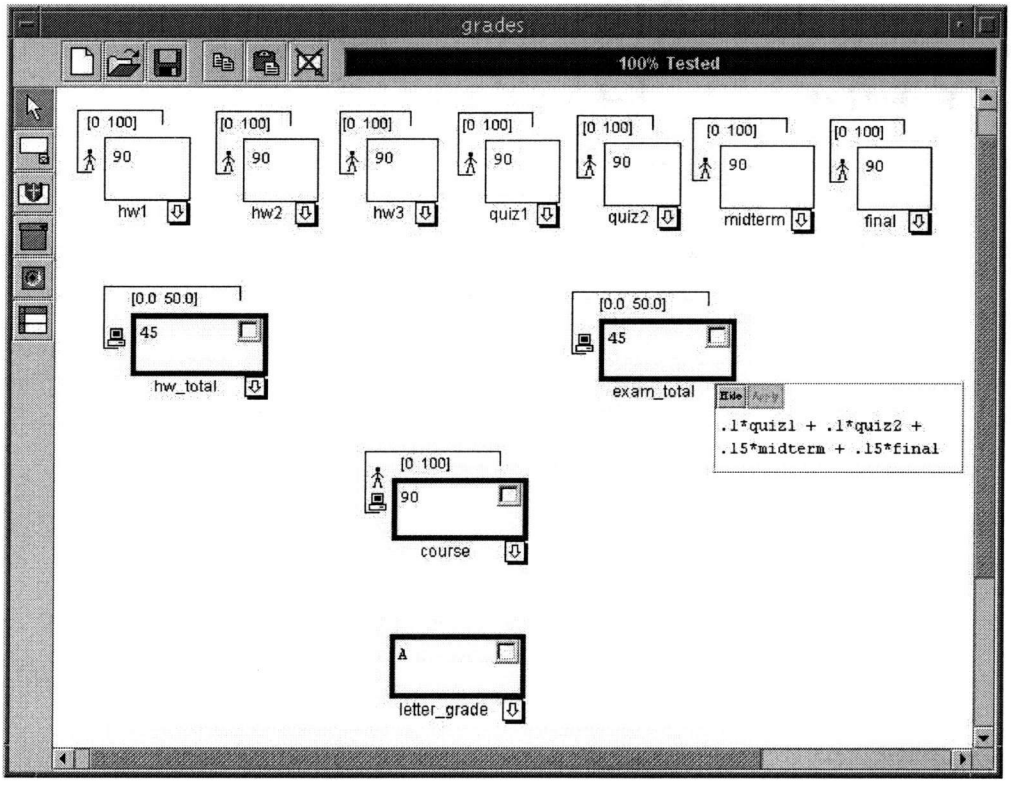


Figure 16: Grades Spreadsheet

(caught off guard?) when Forms/3 couldn't propagate a guard. Even better, most had enough of an understanding of basic propagation that they could anticipate a reaction with no prior experience.

In summary, it appears that our subjects had little trouble understanding guard propagation. What they couldn't predict, they could understand once they saw it happen.

5.3 HYPOTHESIS 3 RESULTS

H3: End users are not overly distracted by the assertion conflicts or by the value violations.

It has always been our goal that we not distract the user from his / her spreadsheet task. Unlike Excel, we avoided pop up windows that demand the user's attention. We like the analogy of the telephone and e-mail systems. One interrupts whatever you are doing and demands immediate attention. The other waits until you wish to see if you have messages. So, in Forms/3 when a value violation occurs, the value is immediately circled in red. The user notices, but nothing is asked of him / her. They can deal with it on their own time. Microsoft Excel pops up a window demanding immediate attention whenever an input cell has a value violation. Excel doesn't have assertion conflicts since it does not propagate assertions. Forms/3 reacts to assertion conflicts by circling the stick figure and computer icons in red and by replacing the abbreviated range notation with "Conflict" written in red. Still, our concern with attention economics caused us to investigate whether or not these red ovals and "Conflict" messages distracted our subjects' attention.

We looked at this issue by examining the electronic and audio transcripts that reveal the order of the subtasks. We also asked the Guard subjects on the post session questionnaire if they felt distracted by the red ovals and "Conflict" messages.

5.3.1 Value Violations in Temperature Problem

Each of the five subjects with guards began the Temperature task by editing the guard on the *input_temp* cell to reflect the new Celsius range. The value previously placed in the *input_temp* cell was 200. Therefore, a value conflict confronted each of the five subjects after their guard edit. Four of them immediately changed the value of *input_temp* to something within range, thus removing the red violation oval. One subject lived with the red violation oval while completing all her edits. Only after she completing all her formula edits and her guard edit on *output_temp* did she do something about the violation. So, four out of five Guard subjects addressed the value violation before going on to their formula edits. This proved to be not uncommon as four out of five Control subjects also changed the value of *input_temp* before editing any formulas. All the subjects were asked about the order in which they did the maintenance task. Almost all mentioned something about choosing to work from the top down or from input to output. It makes sense, then, that end users with guards and those without tended to change the value of the input cell before working on formula edits. From their comments there was no evidence that the Guard subjects were distracted by the value violations. Most mentioned that the red ovals were expected or easily explainable. Subject G5's response was typical. "Now they are all mad." "Because that doesn't fit within the range of 0 to 100."

5.3.2 Assertion Conflicts in Temperature Problem

The subjects were also observed as they dealt with assertion conflicts. Each Guard subject experienced an assertion conflict as soon as they modified the original Temperature spreadsheet. The conflict appeared on the *output_temp* cell. All five subjects continued working in a top down fashion and only dealt with the assertion conflict after they had edited all previous cells. Similarly, all the Control

subjects worked with a top down approach as well. Most commented later that they saw the conflict, but decided to deal with it (if it was still there) only when they got to that cell. Clearly, the subjects with guards were not distracted by the conflict enough to go off task in the Temperature problem.

5.3.3 Assertion Conflicts in Grades Problem

There were no instances of value violations in the Grades problem, so we will talk only about assertion conflicts. Initially the Grades spreadsheet is in a state of assertion agreement, but any change in a formula or in a user entered assertion will cause a different system generated assertion to be propagated. In other words, assertion conflicts occur as soon as anything used to propagate assertions is changed. All the Guard subjects began by either editing a formula or a user entered guard. Therefore, an assertion conflict occurred immediately on the *course* cell after each Guard subject's first edit. We will measure the distraction by looking at when the subject dealt with the assertion conflict on the *course* cell.

To complete the Grades modification, a Guard subject had to change all the input guards to reflect the new grading scale of 0 to 100 and they had to edit the formulas in *hw_total* and *exam_total* to reflect the course weighting. If these modifications were done correctly, then the assertion conflict generated with the first edit would vanish. In four out of five cases, the Guard subjects continued working on their modifications after the assertion conflict was signaled. Their modifications were correct and the assertion conflict disappeared. They never had to deal with it at all. The red ovals and "Conflict" message did not distract them enough to divert their attention. In the fifth case, Guard subject G4 changed the *hw1* guard which caused an assertion conflict on cell *course*. She then used the copy / paste technique to place the same guard, [0 100], onto the other input cells. Still working in a top down fashion she said aloud, but did not check the decision boxes, that the *hw_total* and *exam_total* cells were right. Note that the subject had

not yet modified, or even looked at, the formulas in these two cells. At this point, she addressed the assertion conflict on the *course* cell. Forms/3 used the existing formulas to propagate a guard of [0 700]. Subject G4 changed her guard to agree. The assertion conflict disappeared. Later she said that she forgot that *course* "could only go to 100%." So, this subject seemed distracted enough to interrupt her edits. She later went on to complete the formula edits and then change the *course* guard back to [0 100]. In the Temperature problem, this same subject dealt with the assertion conflict only after she had finished all her edits. So, the assertion conflicts were not consistently distracting to this subject.

5.3.4 Distraction Summary

In conclusion, we looked at when each Guard subject handled the Temperature problem's value violation. Their order of edits was essentially the same as those of the Control group. So, we have no reason to believe that the red violation oval distracted our Guard subjects. Assertion conflicts were generated immediately in both problems resulting in ten observations. In nine out of ten times, the subject continued with their task (editing formulas) and only dealt with the conflict (if it was still there) when they got to that cell. Also, their comments during their tasks and during the post session questionnaire all confirm our findings that our violation / conflict feedback is not overly distracting.

5.4 HYPOTHESIS 4 RESULTS

H4: End users with guards will modify spreadsheets more correctly.

5.4.1 Temperature Problem

Only one subject out of ten turned in an incorrect Temperature modification. She was a member of the Control group, subject C1. Figure 17 shows her modified spreadsheet. Her error is seen in cell *a* where she multiplied by 5/9 rather than by 9/5. This error escapes detection even though she has chosen 100 (degrees Celsius)

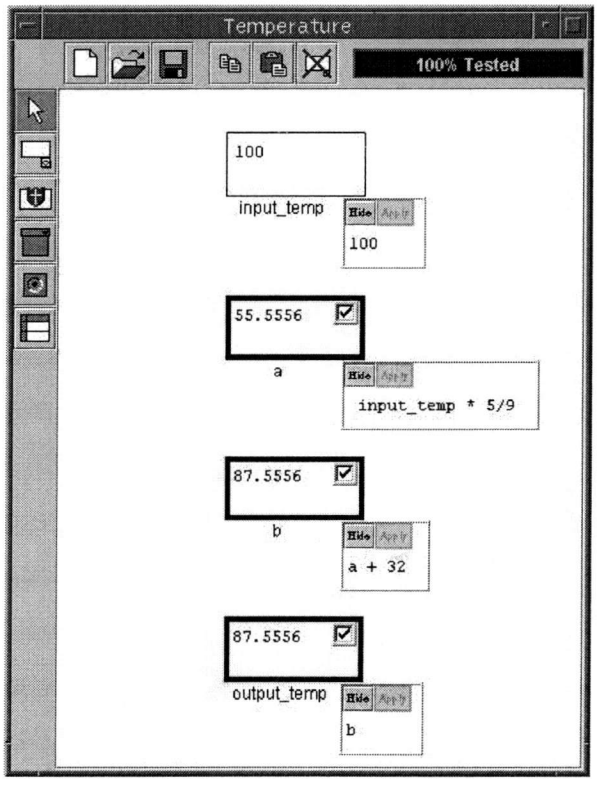


Figure 17: Subject C1's Incorrect Temperature Spreadsheet

as an input value. Recall that the problem description stated in two places that 100 degrees Celsius equals 212 degrees Fahrenheit. She tested her program by doing the arithmetic in her head using her incorrect formula. She said, "5/9 is a little bit more than 5/10." So 55.5556 looked right. Then she added 32 to that and checked off 87.5556 as correct. When asked how confident she felt, she said, "I was 99% confident." She went on to mention that in an actual situation she would check the math with a calculator rather than doing it in her head. The way this subject tested turned out to be pretty typical. She tested the computer's arithmetic by referring to her own formulas. Her mention of using a calculator to do an even better job was also common. We will discuss the subjects' ideas about testing in more detail later in this chapter. For now, the point is that an error survived through the testing phase even though the subject had chosen a value, the boiling point, that provided an independent means of verification.

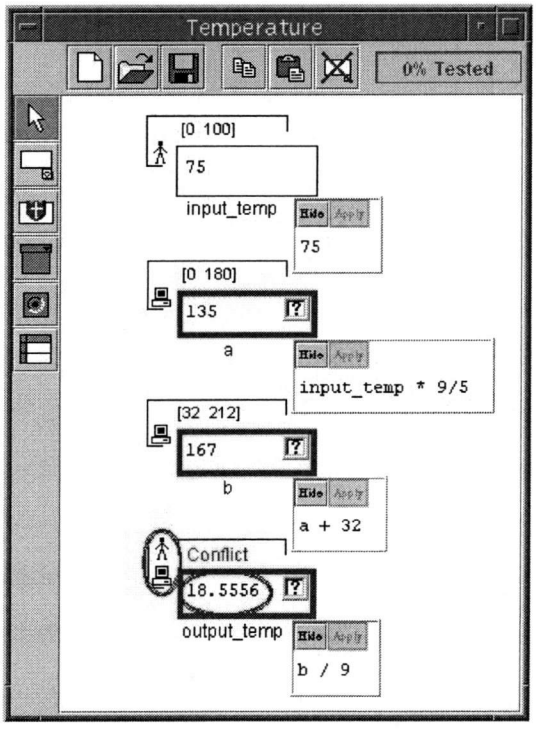


Figure 18: Guard Subject G2's Temperature Spreadsheet

None of our Guard subjects turned in an incorrect Temperature modification. However, three out of five had some sort of error(s) in their spreadsheets at some time. Two subjects entered typos, like saying '+' but typing '*.' Once, a subject entered incorrect formulas due to operator precedence misconceptions. Three subjects entered the conversion formula in two cells rather than in three. Therefore, at a time when they may have felt done, the third cell still contained a remnant of the old conversion formula. See Figure 18. Guards helped to alert the subjects to all but the typos. We describe their actions in more detail in the following:

The initial Temperature spreadsheet used four cells to convert Fahrenheit to Celsius. The conversion formula was spread out over three cells. Guard subjects, G2 and G3, used just two cells for the formula as they put * 9/5 into one cell. Figure 18 shows exactly what subject G2 saw except that we have shown all the cell formulas. Subject G2's habit was to open a formula, edit it, and then close the formula. Subjects G2 and G3 provided some evidence of how Forms/3 guards may help end users. When they got to the final cell, the entire formula had been entered, but there was still a remnant of the old formula left in *output_temp*. The guard on *output_temp* had an assertion conflict. At this point they each opened the guard on *output_temp*. See Figure 19. Their guard still said [0 100] from the original spreadsheet. The Forms/3 guard said [3.5556 23.5556]. Upon seeing this information one of the subjects immediately said, "There's got to be something wrong with the formula." The other subject also immediately said, "And Forms/3 says it should go from something else totally different, so that means that somewhere my formula is wrong. But I know overall that it should be between 32 and 212 because it is supposed to be in Fahrenheit." Each of these two subjects quickly interpreted what the Forms/3 number line was telling them. They knew to examine their formulas for errors. That is exactly what we want a subject to do when the Forms/3 range differs from what they know is the true range. Each of the two subjects replaced the old formula remnant with a reference to cell *b*. Then the Forms/3 number line said [32 212] as they expected.

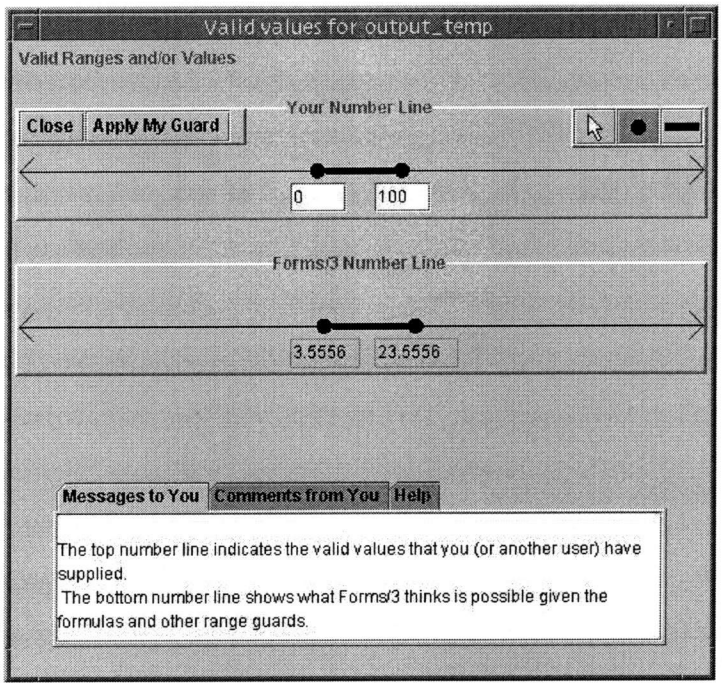


Figure 19: *output_temp*'s Guard as Seen by Subjects G2 and G3

Another subject, Guard subject G4, also used just two cells in her conversion. She, too, then mentioned the conflict. But she chose not to open it. Instead she went right to the third cell and noticed the remnant formula. Then she decided to change everything to use three cells. She knew something was wrong because of the conflict itself.

Guard subject G5 noticed in the original spreadsheet that 200 degrees Fahrenheit was converted to 93.3333 degrees Celsius. So he entered the latter into *input_temp* and then edited all three formula cells to do the conversion to Fahrenheit. But his formulas were incorrect. He said that something was wrong because "it didn't come out to 200 ... and right here there is a conflict." He opened the range guard and changed his number line to [32 212]. The Forms/3 number line said [57.6 237.6]. He introduced another error while troubleshooting before finally fixing both errors. Now the *output_temp* cell held the value 199.99994. He said that Forms/3 rounded off the value so it wasn't exactly 200. He thought the

spreadsheet was working because the output was very close to what he expected and there were no conflicts. While troubleshooting, this subject mainly used the value of *output_temp* to know when his spreadsheet was working correctly, but he also noted that a conflict existed. He correctly edited the guard on *output_temp* and he noticed when the conflict went away. By selecting an input value for which he knew the correct output, he was using an independent verification of correctness. In other words, he didn't just run his input value through his erroneous formulas and check it off as doing the arithmetic correctly. We can't know exactly how much help the assertion conflict provided in telling this subject that his spreadsheet was incorrect since he was also noting the value of the *output_temp* cell. But he did mention the conflict as one of the reasons he knew something was wrong. He also mentioned its absence as a reason why he thought the spreadsheet was correct.

5.4.2 Grades Problem

No one turned in an incorrect Grades modification. In fact, all the Guard subjects modified the formulas perfectly on the first try except one. That subject made a syntax error by forgetting a '*' sign. She got an error message inside the cell. Eventually she tracked down her problem, but since it was syntactic, the guards played no role in helping her.

5.5 HYPOTHESIS 5 RESULTS

H5: End users with guards will test their spreadsheets as thoroughly.

We were concerned that the Guard subjects might gain confidence due to reinforcing guard feedback and therefore test less than their Control group counterparts. We looked at the final percent testedness of each modification problem to determine if the two groups tested their spreadsheets to a similar degree.

	Guard Group					Control Group				
Subject	G1	G2	G3	G4	G5	C1	C2	C3	C4	C5
% Testedness	100	76	88	71	100	100	100	100	75	76
Mean (Std. Dev.)	87 (13.4)					90.2 (13.4)				

Table 4: Grades Problem % Testedness

5.5.1 Temperature Problem

The Temperature problem can be 100% tested with just one test case. All ten subjects reached the 100% testedness level.

5.5.2 Grades problem

To reach 100% testedness, a test case must be run for each of the five grades. See Table 4 which gives the testedness percentage data for the Grades spreadsheets. A Mann-Whitney test showed no significant difference in the testedness percentage between the two groups, ($p = 0.74$).

5.6 CONTROL GROUP TESTING BEHAVIOR

In this section we look at how our Control subjects tested. How did they make their decisions on cell correctness? Would guards have helped them notice errors? What types of test cases did they choose and why?

Two subjects, C1 and C4, liked to see that a cell was correct for more than one test case before clicking on its decision box. It seemed as though they were telling Forms/3 that this cell is absolutely correct when they clicked its decision box.

Subject	Temperature	Errors at some point	Grades	Errors at some point
C1	Checked math	Yes	Predicted output	No
C2	Checked math	No	Checked math	No
C3	Checked math	No	Checked math	No
C4	Predicted output	Yes	Checked math	No
C5	Predicted output	Yes	Checked math	No

Table 5: Control Group Testing Approaches

Subject C1 did this to make sure she understood how Forms/3 expects formulas to be written. She lightened up a bit on her criteria as she became comfortable with Forms/3. Subject C4 just liked verifying the math with two different inputs before pronouncing the cell correct.

Sometimes the Control group subjects tested by just checking the computer's arithmetic inside each cell's formula. Sometimes they attached an intrinsic meaning to the cell with reference to the spreadsheet's inputs. To illustrate the difference, consider subject C1 with the flawed Temperature spreadsheet. She tested it by checking the computer's math using her incorrect formula and she found no errors. But her input value was 100. If she had predicted that *output_temp* should contained 212, then she would have been attaching an intrinsic meaning to the cell. She would have discovered her error. Table 5 summarizes the approach taken by each Control group subject as they tackled each problem.

Three of the Control group subjects typed incorrect formulas into the Temperature spreadsheet. Two of those three knew what to expect out of *output_temp* and discovered their errors. The other subject, C1, checked the math inside her flawed formula and pronounced the spreadsheet correct. With the exception of some syntax mistakes, none of our Control group subjects entered an incorrect formula into the Grades spreadsheet.

Would guards have helped these three subjects find their mistakes? Subject C1, who never found her error, would have seen an assertion conflict on the final cell, *output_temp*. We think she would have at least opened the guard; all the subjects did. There she would have seen that her number line retained the old range, [0 100], and that Forms/3's number line displayed the range [32 87.5556]. If C1 behaved as all of our Guard subjects did and changed her number line to [32 212], then the assertion conflict would have persisted. We believe that she would have at least been aware that an error existed. The other two subjects, C4 and C5, already knew that their spreadsheet contained error(s) because they didn't get their expected output. They, too, would have seen an assertion conflict on *output_temp*. That would have reaffirmed their suspicions. The Forms/3 number line information might have helped them in identifying their mistakes. For example, C5 was using an input value of 38 because she knew that 38 Celsius is a very hot day, but possible. The Forms/3 number line would have given her information on what her formulas did with inputs of 0 and 100. Subject C4 used an input value of 0. With this input, cell *a* can multiply by anything, cell *b* can divide by any nonzero number, and the output will be correct as long as *output_temp* adds 32. Subject C4 didn't make this type of error, the point is that her method of inspecting the final cell for an expected value can miss errors on certain input values. But, a guard will display an assertion conflict if this type of error is made. We believe that in the three cases of flawed formulas seen in the Control group, the use of guards would have alerted or reaffirmed the end users to the presence of an error.

Their selection of test cases was interesting. Some chose realistic values. "Something that you might actually see." These test cases were often difficult to predict an outcome from. Other subjects chose minimums or maximums. In Table 6, "Realistic" means that the subject relied on inputs or combinations of inputs that they thought would happen in real life. "Min/Max" means that the subject tested extreme(s). "Predictive" indicates that the subject could predict the value of the output cell given their inputs. "Boundary" indicates that the subject tested a boundary condition of an "if" expression.

	Problem	Realistic	Min/Max	Predictive	Boundary
Subject C1	Temperature		Max	Yes	n/a
	Grades		Max	Yes	
Subject C2	Temperature	Yes			n/a
	Grades	Yes *			
Subject C3	Temperature	Yes			n/a
	Grades	Yes			
Subject C4	Temperature		Both	Yes	n/a
	Grades	Yes			
Subject C5	Temperature	Yes		Yes	n/a
	Grades	Yes			

* Random "realistic" values except for *midterm* going negative.

Table 6: Control Group Test Case Selection

got unmanageable, these subjects used estimation skills rather than a calculator to check the cells' outputs. They favored these realistic values over inputs for which they knew the output and over inputs that made the arithmetic simpler.

5.7 GUARD GROUP TESTING BEHAVIOR

This section concerns the testing of the Guard subjects. As with the control group we examined their decisions on cell correctness, their use of guards in detecting errors and the types of test cases choosen.

Like the control group, the Guard group sometimes tested by just checking the computer's arithmetic or they attached an intrinsic meaning to the cell. In other words, they could predict the value without using the cell's formula. The difference is illustrated by the Temperature spreadsheet of subject G5. At first, his spreadsheet contained formula errors. He added 32 in cell *a* instead of at the end of the

Subject	Temperature	Errors at some point	Grades	Errors at some point
G1	Predicted output	No	Predicted output	No
G2	Checked math	Yes*	Predicted output	No
G3	Checked math	No*	Predicted output	No
G4	Checked math	Yes*	Checked math	No
G5	Predicted output	Yes	Checked math	No

* There was a remnant of the old formula conversion left in cell *output_temp* because the subject attempted to do the new conversion using just two cells.

Table 7: Guard Group Testing Approaches

conversion. He found his error largely because the output value wasn't even close to what he expected, which was 200. He was attaching a meaning to *output_temp* beyond what its formula box contained. If he had been simply checking the math through each formula box, it is highly doubtful that he would have caught his mistake. In fact, this subject clicked cell *a* as being correct even though it was the source of his error. He only recognized that there was a problem when he got down to *output_temp*.

Table 7 summarizes the basic approach taken by each Guard group subject as they tackled each problem.

Three Guard subjects, G2, G3 and G4, attempted to do the temperature conversion in two cells rather than in three. Subjects G2 and G3 opened the range guard that was in conflict, saw Forms/3's number line, and said that there must be an error in the formulas. In both cases the Forms/3 number line indicated a range of 3.5556 to 23.5556. They knew this was wrong. These two subjects found that the last cell was still dividing by 9, which was a remnant of the old conversion. They changed this to just refer to cell *b* and resolved the conflict. The other subject, G4, said "... oh-oh, there's a conflict." She didn't open the guard. She went right to the remnant formula on the same cell and noted that she didn't want to divide by 9

there. She went on to correct the problem by using all three cells to do the conversion. In all of these cases, an assertion conflict on a guard was the first indication of a problem. Also in all of these cases, the subjects had chosen input values for which they didn't know the correct output value. So, they wouldn't have recognized a problem due to an unexpected output. If these three subjects hadn't been Guard subjects, would they have forgotten about the last cell because they were done entering the formula? One Control group subject, C3, also did the conversion in two cells. He remembered the last cell, perhaps partially because his first step had been to open all four formula boxes. One other Control group subject, C5, started by entering the whole conversion formula into one cell. She asked if she had to use all three cells, and was explicitly told to make the answer appear in *output_temp*. So, with some difficulty, she reworked her formula to use all three boxes. We do not have evidence of a Control group member leaving formula remnants in place. All we do know is that it is impossible to leave formula remnants and not be alerted to the problem by an assertion conflict.

Two subjects, G2 and G4, made formula errors which they noticed as soon as they clicked 'Apply.' For example, G4 said "plus," but typed a '*' symbol. She immediately noticed a very large resultant value in the cell and corrected the problem. Subject G2 modified cell *a*'s formula to say "*input_temp* * *input_temp*" "instead of *input_temp* * 9." He, too, found his mistake immediately after clicking 'Apply.' Having guards didn't help with either of these "formula entry" problems, although they could have in the form of assertion conflicts.

Subject G5 predicted an output for the Temperature spreadsheet. He had a true formula error due to improper application of algebraic precedence rules and didn't get his expected result. Plus he noted that he had a conflict. These two events, not getting his expected result and having an assertion conflict, led him to start troubleshooting his spreadsheet. If he had been like the other half of our subjects and used a nonpredictive input, then the conflict would have been his only indication. He mentioned this value.

G5 - 4. "... it had a conflict so you know that something's wrong. "I mean, I knew anyways because I'm looking at the, I knew that I ... what value I wanted to come out. And if it doesn't come out to anything near it, I know it's going to be wrong. But if you didn't know, it would be nice to have that conflict there."

With the exception of some syntax mistakes, none of our Guard group subjects entered an incorrect formula into the Grades spreadsheet.

Another interesting aspect to their testing was their selection of input values. Some chose "more real" values. "Something that you might actually see." These test cases were often difficult to predict an outcome from. Other subjects chose minimums or maximums or boundary values. In Table 8, "Realistic" means that the subject relied on inputs or combinations of inputs that they thought would happen in real life. "Min/Max" means that the subject tested extreme(s). "Predictive" indicates that the subject could predict the value of the output cell given their inputs. "Boundary" indicates that the subject tested a boundary condition of an "if" expression.

	Problem	Realistic	Min/Max	Predictive	Boundary
Subject G1	Temperature		Max	Yes	n/a
	Grades		Both	Yes	3 of 4
Subject G2	Temperature	Yes			n/a
	Grades		Min	Yes	
Subject G3	Temperature	Yes			n/a
	Grades		Both	Yes	
Subject G4	Temperature	Yes			n/a
	Grades	Yes	Min		
Subject G5	Temperature	Yes		Yes	n/a
	Grades	Yes *			1 of 4

* Random "realistic" values except for *midterm* and *final* going negative.

Table 8: Guard Group Test Case Selection

When asked about their "realistic" input values, most subjects said that they were just random values within the prescribed range(s). They were values that could really happen. One subject in the Guard group, G3, said that she didn't like to test the extremes. She thought that they worked by default. She even mentioned that they take less math. On the Temperature problem, only one Guard subject tested her spreadsheet with an extreme value. That was subject G1 using the value 100. But because of the [0 100] guard on *input_temp*, in actuality, all five subjects received information in the form of propagated guards about what their spreadsheets did with extreme inputs. All five Guard subjects correctly changed their number line to the Fahrenheit range of [32 212] on *output_temp*, thus doing a "pseudo" verification for the extreme inputs. Even users like G3, who actively avoided the extreme inputs, did pseudo test cases on the extremes without realizing it.

At least two Guard subjects used the guards as one indication that their modification was correct. Each of them explicitly said that they were reassured as to their spreadsheet's correctness when the Forms/3 guard held the range [32 212]. Conversely, everyone who dealt with a conflict knew that something was wrong.

6. CONCLUSIONS

In this chapter, we discuss what we conclude from our empirical study and we propose future work suggested by our subjects' reactions and responses.

6.1 CONCLUSIONS FROM OUR PROTOCOL ANALYSIS

6.1.1 Working with Guards

Our research study found that end users can understand and can work meaningfully with Forms/3 assertions. In particular we found that:

- End users attached correct meaning to the guards and knew when to edit them as the meaning changed.
- End users easily edited existing guards.
- End users understood Forms/3's propagated guards.
- End users understood the nature of and how to remove value violation ovals.
- End users understood the nature of and how to remove assertion conflict ovals.
- End users stayed on task despite feedback about value violations and/or assertion conflicts.
- End users could use the Forms/3 guards to recognize potential formula errors.
- End users could use the Forms/3 guards as another check to gain assurance of their spreadsheet's correctness.
- End users with guards did not test less than end users without guards.
- Guards helped end users find errors.
- Guards would have helped end users without guards to find errors.

As a result of the above observations, we conclude that our feedback was informative yet unobtrusive. It was useful and easy to understand. The underlying propagation surprised no one and was predictable by most. Our red ovals used to signal value violations and assertion conflicts were noticeable yet undemanding when the user wanted to stay on her task. In short, we received no feedback that our presentation of assertions fell short of our goals. End users can and do use Forms/3 assertions effectively.

6.1.2 Ancillary Findings

The protocol analysis provided insight and information about the way our ten experimental subjects chose their test cases and how they made testing decisions.

6.1.2.1 Selection of Input Values

Random realistic input values were very popular during our subjects' testing. They used these inputs even though they could not predict the resultant output. When the subjects could not predict an output, they opened the formula boxes and plugged values into the formulas. In effect, they checked Forms/3's arithmetic. No one found an error this way. Our subject, C1, who turned in an incorrect Temperature spreadsheet, just checked the arithmetic using her flawed formula and clicked the decision box as correct. As a Control subject, she had no assertion conflicts to warn her of potential errors. She wasn't the only one. Subject G5 started his Temperature spreadsheet modification by incorrectly adding 32 to *input_temp*. He tested this cell by doing the math in his head. He clicked the decision box as being correct. He later found his error when the value of *output_temp* wasn't what he had predicted. He also had the benefit of an assertion conflict to tell him to look for an error.

Some subjects used extreme values in their test cases while others avoided the minimums and maximums. One subject, G3, stated that minimums and maximums work by default and use less math. She thought she was being more thorough by testing the midranges. Our Guard subjects all ran pseudo test cases on the minimums and maximums due to the propagated ranges. Only one subject out of ten intentionally ran at least one boundary condition test case.

6.1.2.2 Local vs. Global Testing

We identified another characteristic of testing: local vs. global testing. Local testing is defined as looking only at the referenced cells when making a testing decision. Global testing is when a user looks at the spreadsheet input cells or uses domain knowledge to decide whether a formula cell's value is correct. To illustrate the difference, consider the *course* cell in the Grades spreadsheet. If the user tests this cell by adding together *hw_total* and *exam_total* (checking the math), then *course* has been locally tested. If the user predicts what *course* should be from the input cell values, then *course* has been globally tested. Subjects mixed these two styles of testing within the same spreadsheet. They may test *output_temp* in the Temperature spreadsheet globally, and test cell *b* of the same spreadsheet locally. In fact, many intermediate cells have no intrinsic meaning beyond their formulas and would be difficult to test globally. Unfortunately, it is quite easy to test cells with intrinsic meaning in a local fashion rather than globally. This is what subject C1 did while testing *output_temp* in the Temperature spreadsheet. She locally tested cells *a* and *b*. Then she tested *output_temp* locally by checking the math. She failed to step back and realize that her spreadsheet was converting 100 degrees Celsius to 87.5556 degrees Fahrenheit.

6.2 FUTURE WORK

With this study behind us, we can now look forward to new questions. These include:

- What other types of guards (besides range guards) would be useful to end users?
- How would multiple types or combinations of guards be displayed to the end user?
- Will end users use guards as they create new spreadsheets?
- How can we integrate assertions together with other Forms/3 tools such as "Help Me Test," regression testing, slicing, and the WYSIWYT methodology tool set?
- Are there combinations of Forms/3 feedback, especially colors, that are confusing? (Such as red circles, blue borders, pink cell backgrounds.) Guard subject G5 was confused by a red value violation oval in a cell with a blue border. "It's right, but it's wrong."
- Should values that violate guards be used "as-is" in formula cells? Or should the user be warned that an out of range value was used in this calculation?
- Are Forms/3 guards equally useful on large spreadsheets or multi-form spreadsheets?

These are questions that may be considered now that the basic usability questions have been addressed. They involve, but are not limited to, issues with multiple types of guards, integration, and scalability. Since this was a protocol analysis with few subjects, larger numbers of subjects are needed in future studies to provide statistical significance on issues of spreadsheet creation and maintenance.

Testing issues, such as the tendency to check the computer's math, could be addressed in future work. Our observations show an overwhelming tendency for the end users to work from inputs cells through intermediate cells to output cells in

data flow order. Hence, they test intermediate cells before final cells. When intermediate cells have no meaning beyond their formulas, there is little alternative to checking Forms/3's math. We did not observe subjects skipping the testing of these intermediate cells. They check the math even though many of them stated that the computer would do the math correctly. Will the subjects tire of checking something as reliable as a computer performing arithmetic calculations? Will this lead them to stop testing altogether, even globally? How can our tools help them test in a global, predictive fashion?

It is possible that our subjects were predisposed to checking Forms/3's arithmetic because they were told that Forms/3 was an experimental software language. Were they testing Forms/3 or were they testing their own spreadsheets? One subject tested the guards by entering values barely out of range just to see if they would be circled. Would he not have done that with Excel's assertions because Excel is a commercial product?

How can our tools encourage test cases that exercise boundary conditions? Our Guard subjects seemed to test minimum and maximum inputs by examining propagated ranges. Is that good enough or should actual test cases be run?

There is much more work to be done to hone our collection of Forms/3 tools to help end users create and maintain error-free spreadsheets. With error rates as they are, though, the research is well worth the effort.

BIBLIOGRAPHY

[Blackwell 1999] Blackwell, A. and Green, T. R. G. Investment of Attention as an Analytic Approach to Cognitive Dimensions. In T. R. G. Green, R. H. Abdullah & P. Brna (Eds.) *Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group (PPIG-11),* pp. 24-35.

[Burnett 2001] Burnett, M., Atwood, J., Djang, R., Gottfried, H., Reichwein, J., Yang, S. Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm, *Journal of Functional Programming* 11(2), March 2001, pp. 155-206.

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

[Ernst 1999] Ernst, M., Cockrell, J., Griswold, W., and Notkin, D. Dynamically Discovering Likely Program Invariants to Support Program Evolution. *International Conference on Software Engineering*, Los Angeles, California, May 1999, pp. 213-224.

[Green 2000] Green, T. R. G. Instructions and Descriptions: some cognitive aspects of programming and similar activities. Invited paper, in Di Gesù, V., Levialdi, S. and Tarantino, L., (Eds.) *Proceedings of Working Conference on Advanced Visual Interfaces (AVI 2000).* New York: ACM Press, pp. 21-28.

[Green 1996] Green, T. R. G. and Petre, M. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *J. Visual Languages and Computing* (1996), 7, pp. 131-174.

[Nardi 1991] Nardi, B. and Miller, J. Twinkling lights and nested loops: distributed problem solving and spreadsheet development. *Int. J. Man-Machine Studies* (1991) 34, pp. 161-184.

[Panko 2000] Panko, R. Spreadsheet Errors: What We Know. What We Think We Can Do. *Proceedings of the Spreadsheet Risk Symposium European Spreadsheet Risks Interest Group* (EuSpRIG) July 2000.

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

APPENDICES

APPENDIX A:
BACKGROUND QUESTIONNAIRE

Code _____

All Subjects Background Questions

1. Major _____
2. Year (Sophomore, Junior, ...) _____
3. Overall GPA _____
4. What were your last 2 college math classes? You may list classes that you are currently taking. List a high school class if you haven't taken 2 in college yet. Be sure to label them "H.S."_____
5. Do you have previous programming experience? (Check all that apply)
   - High school course(s). Number? _____
   - College course(s). Number? _____
   - Professional. How long? _____ years
6. Have you ever created a spreadsheet for... (Check all that apply)
   - A high school course? _____
   - A college course? _____
   - Professional use? _____
   - Personal use? _____
7. Have you participated in any previous Forms/3 experiments? Yes / No
8. Is English your primary language? Yes / No
   If not, how long have you been speaking English? _____

APPENDIX B
TEMPERATURE PROBLEM DESCRIPTION

Given: A correctly working spreadsheet that converts degrees Fahrenheit to degrees Celsius. It was designed to measure the temperature of water used in a cooling system. So, it needs to work from water's freezing point, 32 deg. F (0 deg. C), to water's boiling point, 212 deg. F (100 deg. C).
C = (F - 32) * 5 / 9

Task: Using the existing cells of this spreadsheet, change it to convert degrees Celsius to degrees Fahrenheit. As above, the data will range from water's freezing point to its boiling point.
F = C * 9 / 5 + 32

This spreadsheet will be used by NASA during the next shuttle launch. It must be correct.

Scenario:

The boiling point of water is 212 degrees Fahrenheit or 100 degrees Celsius.

APPENDIX C
GRADES PROBLEM DESCRIPTION

Given: A correctly working spreadsheet calculating a student's grade in a course.

Course	Weighting
hw1	10%
hw2	20%
hw3	20%
quiz1	10%
quiz2	10%
midterm	15%
final	15%
	100%

The way the original spreadsheet is designed, hw1 is worth up to 10 points, hw2 up to 20 points, final up to 15 points,... All points are totaled to arrive at the course point total. A letter grade is assigned based on the following:

Letter Grade	Criteria
90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
< 60	F

Task: This teacher has found that students got confused with some work being graded on a 10 point scale, some on a 15 point scale, and some on a 20 point scale. She has decided to grade everything on a 100 point scale. In other words, every assignment and every type of exam will receive from 0 to 100 points. But, the course weighting will remain the same. In addition, the *hw_total* and the *exam_total* cells must continue to range from 0 to 50. And the *course* cell must continue to range from 0 to 100. This means the weighting will have to happen inside the formulas. So, since hw1 is worth 10% of the course total, and *course* ranges from 0 to 100, then a perfect hw1 score of 100 should contribute 10 points toward the course total. Letter grades are assigned using the exact same criteria as before.

Example: Suppose an assignment was worth 27% of the course total, and the assignment was graded on a scale of 0 to 100. Then, in one of the formulas, the assignment points should be multiplied by 0.27 so that a perfect assignment score of 100 would contribute 0.27 * 100 = 27 points toward the course total.

This spreadsheet might be used to calculate YOUR grade! So, it better be right!

APPENDIX D
GRADES PROBLEM HANDOUT

Scenario I.

Suppose that a course grade is based on 3 assignments.

Assignment 1	25%
Assignment 2	30%
Assignment 3	45%
	100%

The teacher decided to make Assignment 1 worth a maximum of 25 points, Assignment 2 worth 30 points, and Assignment 3 worth 45 points. Then to calculate the final grade, all she had to do is add up the points.

Example 1: Student A turned in 3 perfect assignments for scores of 25, 30, and 45. Therefore, Student A received 25+30+45 = 100 for a course grade.

Example 2: Student B earned the following scores:

Assignment 1	21 out of 25
Assignment 2	27 out of 30
Assignment 3	36 out of 45
Course	84 out of 100

Scenario II.

Same 3 assignments with the same weights as above

Assignment 1	25%
Assignment 2	30%
Assignment 3	45%
	100%

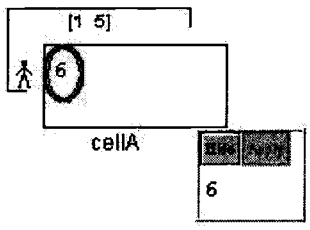
This time the teacher decided to make all 3 assignments worth 100 points, but she wanted to keep the assignment weighting the same. She also wanted the course grade to range from 0 to 100 just like above. So, the teacher calculated the course grade by multiplying Assignment 1's score by 0.25, multiplying Assignment 2's score by 0.30, multiplying Assignment 3's score by 0.45, and adding up the 3 products. Let's re-look at the two example students.

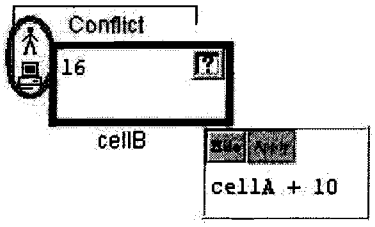
Example 1: Student A turned in 3 perfect assignments for scores of 100, 100, 100. So, course = 0.25*100 + 0.30*100 + 0.45*100 = 100

Example 2: Student B received 84, 90, 80 on the three assignments. Note that these are equivalent to his scores before on different scales.

So, course = 0.25*84 + 0.30*90 + 0.45*80 = 84

APPENDIX E
POST SESSION QUESTIONNAIRE





For questions 1-4, choose the best answer from the list on the right. Answers on the right may be used more than once. Questions 5-7 are fill-in-the-blank.

1. What does the red oval on *cellA* mean? ____
2. What does the little stick figure in the *cellA* guard mean? ____
3. Why are a stick figure and a computer on *cellB*'s guard? ____
4. What does the red oval on the *cellB* guard mean? ____
5. Given the formula in *cellB* and the range guard on *cellA*, what do you think Forms/3 says are the valid range(s) for *cellB*? _____

_____

6. What can be done to get rid of the red oval on *cellA*?_____

_____

7. *cellB*'s user guard is [10 50]. Its Forms/3 guard is [11 15]. Assume all user guards on this spreadsheet are correct. How can this conflict be resolved? _____

_____

8. Did the red ovals or conflict messages distract you or worry you while you were working?_____

A. The user and Forms/3 disagree on the valid range(s) for this cell.
B. The guard was supplied by Forms/3.
C. The user should open this guard.
D. The value falls outside the valid range.
E. The guard was supplied by the user.
F. The guard was supplied by both Forms/3 and by the user.
G. Forms/3 disagrees with the range [1 5].
H. The user and Forms/3 agree on the valid range(s) for this cell.
I. The user must change this guard.
J. The user should not have supplied a guard for this cell.
K. The value is almost outside the legal range.
L. I'm not sure.

APPENDIX F

CONTROL GROUP SUBJECT C1

input_temp	a	b	output_temp
100	Check	Check	Check

Table F.1: Control Subject C1's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
100	0	0	0	0	0	0	Vis			
	100						Vis			
		100					Vis			
50	50	50					Chk			
			100					Vis		
				100	100	100		Vis		
			50	50	50	50		Chk	Chk	Chk (F)
100	100	100	100	100	100	100				Chk (A)
85	85	85	85	85	85	85				Chk (B)
75	75	75	75	75	75	75				Chk (C)
65	65	65	65	65	65	65				Chk (D)

Table F.2: Control Subject C1's Grades Test Cases

F.1 TEMPERATURE PROBLEM

Subject C1 used local testing while checking off each of the Temperature problem cells. Local testing is defined as testing a cell by considering only the cell's

immediate reference(s) rather than the input(s) to the spreadsheet. For example, subject C1 referred only to the value of cell *a* while testing cell *b*.

Subject C1 made her testing decisions by following each cell's formula and doing the arithmetic in her head. See her comments below for details.

1. "So 100 times 5/9, I would check it with a calculator, but, that looks about right because 5/9 is a little bit more than 5/10."
2. "Normally I would check it with a calculator."
3. "Then 55 + 32 is 87. That makes sense."

Since she was just rechecking the computer's arithmetic and using local testing, C1 never found the error in cell *a*'s formula, *input_temp* * 5/9. She should have used the fraction 9/5. Despite the fact that her input temperature was the boiling point in Celsius, her local testing methods failed to reveal that her output temperature was not the boiling point in Fahrenheit. Recall that the Temperature problem description presented the boiling point equivalence information in two different places.

If this subject had been using guards, she would have seen an assertion conflict on *output_temp*. Hopefully, this would have brought to her attention her current range of *output_temp*, [32 87.5556]. All of our Guard subjects changed their number line to display a range of [32 212]. If she had done the same, she may have been reminded of the meaning of the cell; *output_temp* displays Fahrenheit temperatures, not just the value of its formula.

F.2 GRADES PROBLEM

In this problem, subject C1 used a global testing approach. For example, while testing the different letter grades, this subject always referred to the spreadsheet's inputs as producing a specific grade. She didn't check off *letter_grade* because of

the value in the *course* cell, rather she referred to the homework and test input cells. Also, she consistently predicted what a formula cell should contain for each test case.

4. "So then I know that if I switch them to 50's, that I should get 25 if my formula is working." "... and it is 25, so I'm going to say that that box is correct."
5. "Going back with the 100% on everything, this should definitely give me an 'A.' It does."

She was concerned that because she was new to Forms/3, she might not know the exact format that Forms/3 expected. For example, she didn't know whether parentheses were needed or if spaces inside the formula mattered. To convince herself that she entered *hw_total*'s formula correctly, she visually tested it each time she entered 100 into a hw input cell. That is, she watched for change in the *hw_total* cell after each hw input cell edit. Then she changed all the hw inputs to 50, predicted a 25 out of *hw_total*, and then finally checked the *hw_total* decision box. She had done a lot of testing before putting that first checkmark in *hw_total*'s decision box. She had confirmed that each input cell contributed to *hw_total* and that the value was correct for two test cases. It seemed that to her, checking *hw_total* meant that the cell was absolutely correct, not just correct for this test case.

After her success with the formula in *hw_total*, she went to fewer pains in testing *exam_total*. She knew how Forms/3 expected the formula. She commented on how she knew the computer would do the math correctly. She just needed to learn how Forms/3 wanted the formulas written.

6. "We are testing theory. I'm assuming that the computer can actually do the math." "... I am testing more my entry of it." "... I'm not worried about the computer's math."

7. "I was building my confidence in writing the formula, not necessarily my confidence in it doing the formula. Because once I saw that the first one (*hw_total*) did it, I understood how this spreadsheet expects it."

This subject used her domain knowledge about course grading while testing this spreadsheet. She talked about how in any class, all 100's has to be an 'A' and all 50's will be an 'F.' To test the other grades, she entered values in the middle of the range because of grade curves and plus / minus grading.

8. "That's more just how I think of grades. Because teachers like when it gets around 90 or 80, they kind of play with it. Some of them bump it up, some of them bump it down. And so when I'm thinking like grades in my actual schooling and stuff, 100% is an 'A,' 50% is a flunk." "... We work on a plus / minus system. A 90 is not an 'A'. 90 is an 'A-.'" "... I don't think of 90's as being 'A's'."

This subject used local testing on the first problem and global testing on the second. She checked the computer's math on the first problem and said she wasn't worried about it on the second. We don't think that she simply learned through experience not to check the computer's math because during the post session questions she was asked what she would do to be even more sure that her Temperature spreadsheet was correct. She replied that she would check the math on a calculator. She had extensive domain knowledge on the second problem that influenced her choices of test cases.

APPENDIX G
CONTROL GROUP SUBJECT C2

input_temp	a	b	output_temp
90	Check	Check	Check

Table G.1: Control Subject C2's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
64	85	98	0	0	0	0	Chk			
			85	82	72	96		Chk	Chk	Chk (B)
					12					Chk (C)
					100					Vis
					150					Chk (A)
					-50					Chk (D)
					-150					Chk (F)

Table G.2: Control Subject C2's Grades Test Cases

G.1 TEMPERATURE PROBLEM

Subject C2 used local testing and redid the math in her head for each cell. She chose 90 as her input value because it was a "round even number." She entered her formulas correctly on the first try.

1. "90 times 9. That's right."

G.2 GRADES PROBLEM

On this problem, subject C2 also used local testing and compared ratios in her head. Specifically, she said that she tested the *letter_grade* cell by examining the value of the *course* cell rather than the input cells. As shown above, this subject chose real life values for her inputs and varied only *midterm* to test each grade. During the post session question period, she mentioned that it would have been nice if Forms/3 could have helped her find values that would test an 'A.'

2. "84 that would be a 'B.'" "... 75.9% would be a 'C.'"
3. "43 out of 50 (43 was in *hw_total*) is pretty close to these (64, 85, 98) out of 300." "... If it was a little off, I wouldn't have got it."

APPENDIX H
CONTROL GROUP SUBJECT C3

input_temp	a	b	output_temp
57	Check	Check	Check

Table H.1: Control Subject C3's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_grade
70	5	84	0	0	0	0	Chk			
			75	92	85	89				Vis (D)
	90									Vis (B)
	60									Vis (C)
	100									Vis (B)
100										Vis (B)
			85				Chk	Chk	Chk	Chk (A)
		0								Chk (C)
	0									Chk (F)
	50									Chk (D)
		50								Vis (C)
		70								Vis (C)
		85					Chk		Chk	Chk (B)

Table H.2: Control Subject C3's Grades Test Cases

H.1 TEMPERATURE PROBLEM

Subject C3 chose 57, "just a random number", as his input value. He opened all four formula boxes and started to do the conversion in two cells rather than in three. His conversion was correct. When he got to the last (unneeded) cell, he decided to go back and rework his solution to use all three cells. He tested by approximating the computer's arithmetic. He used local testing techniques throughout.

H.2 GRADES PROBLEM

This subject chose realistic values as his inputs. He used estimation skills to test *hw_total* and *exam_total*. He went through most of the grades before remembering to tell Forms/3 of his decisions by clicking in the decision box. Then he went through the grades again, this time checking the decision box of *letter_grade*. He based all decisions about letter grade on the value of the *course* cell; local testing.

1. "(43.8) looks about right." "... you know it's going to be less than 45."

The Forms/3 testing methodology assumes that if a cell is correct, then all the contributing cells upstream are correct as well. The first time subject C3 clicked on *letter_grade*, Forms/3 turned *course* and *exam_total* blue for him. This surprised him a bit.

2. "Because this box worked it changed those for me?"

APPENDIX I
CONTROL GROUP SUBJECT C4

input_temp	a	b	output_temp
150	Check (old)	Check (old)	Check (old)
0	Check	Check	Check
100			Visual

Table I.1: Control Subject C4's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
90 Chk. h1	85 Chk. h2	95 Vis.	50 Vis.	85 Vis.	75 Vis.	100 Vis.	Vis.			
		100 Chk. h3	75 Chk. h4	90 Chk. h5	100 Chk. h6	50 Chk. h7	Chk	Chk	Chk	Chk (B)
Note: This subject created 7 new cells, h1 - h7, to hold the weighted scores.										

Table I.2: Control Subject C4's Grades Test Cases

I.1 TEMPERATURE PROBLEM

Subject C4 was very confused when she started the Temperature problem. She changed the name of the *input_temp* cell to "centegrade" and thought that the spreadsheet should work. She tested the input value 150 with the old formulas, checking the decision boxes as she went.

1. "I'm not seeing how it wouldn't work."
She entered 0 into her "centegrade" cell.

2. "It seems like it will (work) every time."

This subject was confused as to what we wanted her to do. She was reminded of the problem description. Then subject C4 noticed that *output_temp* was -17.7778 and not the 32 she wanted. She decided that she needed to modify the formulas. After modifying all three formulas, she still didn't get 32 in *output_temp* because her order of operations was wrong. She experimented with different orders and stopped when *output_temp* equaled 32. She tested the intermediate cells locally, but knew that *output_temp* was correct because it was correct for the spreadsheet's input of 0 Celsius; global testing. Then she entered 100 into her input cell to see if 212 resulted. It did. She started to do some local testing on the intermediate cells, but stopped because they were already blue.

3. "Oh they are already tested so I don't have to."

She mentioned "testing it (a cell) from three different areas" and she couldn't think of another way to test it. She was remembering what we had done to test the tutorial's "error cells." Those cells had three situations to test. She was trying to test the Temperature spreadsheet's cells three different ways, too, even though they were simple formula cells. It is unknown whether or not she would have tested the Temperature spreadsheet with multiple test cases if she hadn't been mistakenly thinking she had to test everything three times.

I.2 GRADES PROBLEM

This subject created seven new cells, h1 - h7, to hold the weighted scores. Then *hw_total* and *exam_total* summed up the values in the new cells. Subject C4 frequently tried two values in an input cell before checking off the corresponding weighted cell. She used local testing methods throughout this spreadsheet. She tested the cells by double-checking the computer's math by comparing ratios and computing simple sums.

4. "A 17 out of 20 is going to equal the same thing as an 85 out of 100."
5. "I don't have a calculator, but I'm just kind of guessing that that's right."
6. "*hw_total* is going to be 45, 9 plus 17 plus 19."

She tested one letter grade and was confident that the spreadsheet was correct. She ended up with a 75% testedness level.

APPENDIX J
CONTROL GROUP SUBJECT C5

input_temp	a	b	output_temp
38	Check	Check	Check

Table J.1: Control Subject C5's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
70	80	75	80	90	85	55	Chk	Chk	Chk	Chk (C)

Table J.2: Control Subject C5's Grades Test Cases (

J.1 TEMPERATURE PROBLEM

Subject C5 was very familiar with the Celsius system as she had lived under it for most of her life. She chose 38 as her input value because she had a good feeling for how hot it was. She had a good idea of what to expect out in degrees Fahrenheit.

She began by putting the whole conversion formula into the first formula cell, cell *a*. Then she didn't know what to do with the other two cells. She asked if she needed to use them. She was told to use all four cells such that the answer in Fahrenheit was displayed in *output_temp*. So, she tried to split up the formula into the three cells.

This subject repeatedly had trouble remembering to click 'Apply.' Her formulas would be correct but not 'Applied.' Therefore, Forms/3 would be using the old

(invisible) formulas to do the calculations. This caught her time after time. More than once she reworked correct (but "unApplied") formulas because the answer in *output_temp* wasn't what she expected, it was too hot. She modified her formulas until her result was what she expected, about 100. Subject C5 used local testing on the intermediate cells, but global testing on the final cell.

1. "The number in the answer, I like the number."

J.2 GRADES PROBLEM

Subject C5 made many syntax mistakes while typing the formulas. She typed an "el" instead of a one in the cell name *hw1*. She typed a percent sign inside a formula. She also forgot to click "Apply" in this problem as in the last.

She mostly used local testing methods. To be more sure of the spreadsheet, she said she would use a calculator to check the math.

APPENDIX K
GUARD GROUP SUBJECT G1

input_temp	a	b	output_temp
100	Check	Check	Check

Table K.1: Guard Subject G1's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
60	80	90	80	80	80	80	Chk	Chk	Chk	Chk (B)
100	100	100	100	100	100	100				Chk (A)
0	0	0	0	0	0	0				Chk (F)
60	60	60	60	60	60	60				Chk (D)
70	70	70	70	70	70	70				Chk (C)

Table K.2: Guard Subject G1's Grades Test Cases

K.1 TEMPERATURE PROBLEM

Subject G1 used local testing methods on the intermediate cells, but global testing on the final cell. Recall that local testing is defined as considering only the cell's immediate reference(s) rather than the input(s) to the spreadsheet. For example, subject G1 referred only to the value of cell *a* while testing cell *b*. We consider the testing she did on *output_temp* to be global because she related that 100 in *input_temp*, the spreadsheet's input, should result in 212 in *output_temp*. This subject made no errors in her formulas.

K.2 GRADES PROBLEM

For this problem, subject G1 started out using mostly local testing. She based her decision that 'B' was correct for *letter_grade* on the fact that the *course* cell contained 80. For all the rest of the grades, however, she typed in values for which she could predict a letter grade. She tested the minimums, the maximums, and some boundary conditions. So, on four out of five letter grades, she used global testing.

She explained her testing strategy switch:

1. "Because first I am just being realistic because most students will not get all the homework and midterms the same. I am just applying my experience into the computer and I find it is hard for me to say if it is right or wrong." "... I just want to test it and I don't have to be so complicated in the input cells."

This subject mentioned that she was more confident in her Grades spreadsheet than in her Temperature spreadsheet because she tested Grades more. The reason she ran just one test case on the Temperature spreadsheet was:

2. "Because it shows that I am 100% tested and I believe Forms/3."

APPENDIX L
GUARD GROUP SUBJECT G2

input_temp	a	b	output_temp
75	Check	Check	Check

Table L.1: Guard Subject G2's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
0	0	0	0	0	0	0	Chk	Chk	Chk	Chk (F)

Table L.2: Guard Subject G2's Grades Test Cases

L.1 TEMPERATURE PROBLEM

Subject G2 combined two operations into cell *a* so that its formula multiplied by 9/5. When he tested that cell, he very quickly checked it off as being correct. He said that 75 times 9/5 is 135. It was fairly evident that he hadn't even approximated the correct answer, he was trusting the computer's math. It seemed as though this subject equated testing with checking the computer's math and he was willing to trust the computer.

1. "The tested feature is a very nice feature. I like it." "... At the same time I don't know a lot of people who would take the time." "... they blindly trust the computer and say, hey, this is right."

2. "As far as the number being correct, hey, the computer is my calculator. It's going to be right."

When subject G2 got to cell *b*, he correctly edited the formula. As soon as he clicked "Apply" he noticed that the system generated guard on cell *b* changed to reflect the 32 to 212 Fahrenheit range. This may have influenced his testing decision.

3. "... when we do that and 'Apply' it, we get 167. And now we have found that the guard value has changed on *b* from 32 to 212, representing the possible range we could get for Fahrenheit - which is pretty ingenious for this program. I like that. So we click that and say it's OK."

Now, subject G2 has the task completed in two cells. The third cell, *output_temp*, still contains the remnants of the old conversion. Having worked his way down to *output_temp*, he deals with the assertion conflict.

4. "Now we've got a little problem, though, on the output temperature because there's a conflict between myself and the computer. The computer seems to think that - these things are all weird. So, there's got to be something wrong with the formula."

He quickly found the problem in *output_temp*'s formula and corrected it. It is clear that the assertion conflict and the information on the Forms/3 number line effectively alerted this subject to the presence of an error.

Subject G2 was really a textbook case illustrating how to read and use the information displayed by the guards. He noticed guard changes immediately after each formula edit. He quickly interpreted the assertion conflict and the Forms/3 number line and reacted appropriately. When he recognized the 32 to 212 range on cell *b*'s guard and later on *output_temp*'s guard, he was reminded of the meaning

behind the cells. In other words, cell *b* wasn't just a cell that added 32 to cell *a*, it represented the Fahrenheit output temperatures. Reminding yourself of the intrinsic meaning of a cell can only help the testing process. He showed similar awareness of the guards on the next problem as well.

L.2 GRADES PROBLEM

For this problem, subject G2 never changed the values in the input cells. The inputs remained all zeros. A test case of all zeros can hide any number of formula errors, though his formulas were correct. This subject supplemented this minimal testing by noticing the propagated guards each time he clicked 'Apply.' As in the last problem, we believe that the guards might have influenced his testing decisions." Possibly having guards reduced his test cases, although two Control group subjects also tested only one letter grade.

5. "Then we 'Apply' and hide. And basically now our range is between 0 and 50 which is the criteria we wanted."
6. "We 'Apply' it, we hide it. And again we get that homework plus exams will equal, each are on a 0 to 50 scale. And if you were to add those together it would equal 100."
7. "As for the course, we have an 'F,' but the range is going to be *hw_total* plus *exam_total* and so that range is going to be 0 to 100."
8. "I'm going to say that these are all correct." "... because the guards on the *hw_total* and the *exam_total* are each 0 to 50. So if you took the best you could get on homework and the best you could get on exams, 50 plus 50 is 100, that's 0 to 100 scale for the course. The best you can get on any course is 100. Since everything has a zero value you get an 'F' in the course."

The quotes above demonstrate how this subject attributed meaning to the cells beyond what their formulas stated. He used his domain knowledge and the information illustrated by the guards to describe why the *course* cell should range from 0 to 100.

Subject G2 was asked how confident he was in his spreadsheets and why.

9. "Very confident." "... It is very nice to have the second level of reassurance that you put in a range, the computer puts in a range, and then the number is in-between the range. So that's very nice to see that the computer says, well it should be between this and this and it is between this and this." "...I'm just confident with computers."

APPENDIX M
GUARD GROUP SUBJECT G3

input_temp	a	b	output_temp
23	Check	Check	Check

Table M.1: Guard Subject G3's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
0	0	0	0	0	0	0	Chk	Chk	Chk	Chk (F)
	95	80	94	88	70	100				Chk (C)
100	100	100	100	100	100	100	Vis	Vis	Vis	Chk (A)

Table M.2: Guard Subject G3's Grades Test Cases

M.1 TEMPERATURE PROBLEM

Subject G3 completed the task in two cells. After editing the first one, she commented that the computer's guard changed. After editing the second cell, cell *b*, this subject had worked her way down to *output_temp* which still had an assertion conflict. Since the conversion formula has been entered into just two cells, *output_temp* still contained the remnants of the old conversion formula. This is exactly the situation faced by the previous subject, G2. Subject G3 behaved almost identically.

1. "Now I have a problem down here because the ranges disagree. So I say it should go from 0 to 100. And Forms/3 says it should go from something else totally different. So that means that somewhere my formula is wrong."
2. "But I know overall that it should be between 32 and 212, because it is supposed to be in Fahrenheit. So I'll put that in."

She reviewed all her formulas and corrected *output_temp*'s formula and the assertion conflict went away.

3. "And 'Apply' that and, oh, my conflict magically goes away."

While testing, this subject just tested the math in each of her formulas. She couldn't predict an answer in Fahrenheit for her input of 23, so all her testing was local.

4. "23 and then I'm supposed to times that by 9/5. I have no idea what that is off the top of my head, but it is obviously going to be more than 23, but less than 46. So I'll say that that's probably correct."
5. "Then this one is supposed to be $a + 32$. Well that is correct so I'll check that."
6. "I feel comfortable and it's now 100% tested so I feel that it's correct."

The subject was asked how she felt about seeing the conflict after she finished with her first two cells.

7. "I liked it that the computer didn't do everything for me, that it made me think and try and figure out, well lets look and see what the problem is here 'cause it could have been an error in my formula and it was. And I needed to actually go through and figure out, oh, that's just supposed to be *b*." (The formula in *output_temp* was supposed to be just a reference to cell *b*.) "So, I thought that was good."

8. "It was a little disconcerting going through when I was changing each of the boxes that the ranges would change so much. But, I understood that it was still in process, that those weren't the final numbers and it was OK."

Similar to the last subject, this subject used the guard feedback to help her decide when her spreadsheet was correct. The fact that Forms/3 said that her output range was [32 212] contributed to her belief that her spreadsheet was done correctly.

9. "... I went through and I looked at all my little formulas, all of them together. And looking at all of them, I have my C, which is supposed to be between 0 and 100, and it is. So, I have my input temperature of 23. And then it is supposed to be input temperature times 9/5, which is correct with the formula. Then I am supposed to take that total and add 32, which is also correct. And, yes, I feel that it is correct because the end result is supposed to be between 32 and 212, and it is."

At the end of the session, student G3 said:

10. "Looking back really it would have made more sense to put in a number that I could have easily gone through and done the math in my head, probably like 20."

She was asked if she could think of any numbers that she would just know the answer for.

11. "Zero would be a good one to do, too." "... it would have popped out 32."
12. "... a 100 which would have popped out 212. But I always prefer to check more of the midranges because it seems like the minimums and the maximums almost by default that those happen, because it doesn't take as much math."

13. "To me testing the middle makes it seem, 'Does this really work?' Because working on the minimum and maximum kind of seem default to me."

M.2 GRADES PROBLEM

As subject G3 started the Grades problem, she said aloud that the ranges across the input cells would need to change, but the [0 50] ranges on the *hw_total* and *exam_total* cells would have to stay the same. She got this information from the problem description. Then she described how she was going to put some weighting inside the input cells. For example, she wanted the *hw1* cell to "take some number and multiply it by .1." She was told to keep the top row of cells as input cells so that they contained only the score on that assignment or test. She proceeded to put the weighting inside the *hw_total* and *exam_total* cells. When she was done, Forms/3 displayed the range on *hw_total* as [0 9] because she hadn't changed her input cells' guards yet. Forms/3 still thought that *hw1* ranged from 0 to 10, and that *hw2* and *hw3* ranged from 0 to 20.

She struggled to figure out why Forms/3 thought the range was [0 9] on *hw_total*. She also noticed that *exam_total*'s new range was [0 6.5]. She knew that she wanted them both to be [0 50]. She talked about multiplying everything in *exam_total*'s formula by 100. But, before she did that she decided that that would make the range too big. Not knowing what to do, she recalled her actions to date on this spreadsheet. She didn't know why Forms/3 chose the [0 9] range on *hw_total*.

14. "If it just took half of the total, the potential total, that would still be 25. It wouldn't be 9." "... 10 plus 20 plus 20 times .5, if they got 100% on everything."

The 10, 20, and 20 that she referred to were the homework guard upper limits. She said that if they got 100% on homework one, that a 10 would appear in *hw1*. So she was still thinking of the input cells in the scenario one way, they didn't range

from 0 to 100 to her. She had mentioned changing the input guards at the start of the problem, but then got sidetracked with the discussion of where not to weight the inputs. She was told to review the problem description. It didn't take her long to decide to change all her input guards to a range of [0 100].

15. "Now it went back to 0 to 50." (Referring to the guards on *hw_total* and *exam_total.*)

She proceeded to test her spreadsheet, first with all zeros and then with "something that might happen." She based her decision that *letter_grade* was a 'C' by looking at the *course* cell, which was 78.7; local testing. She later said that she had expected the scores she entered to "probably pop out a 'B' or a 'C.'" So, she had some global idea of the end result. Then she tested using an all 100's test case. She predicted the outcomes of all the intermediate cells as well as the final cell, *letter_grade*; global testing.

This subject paid close attention to the changing guard information on both problems. She used this feedback to indicate when she had an error and when she was finished.

16. "The guard showed me that what I had been doing was not what I wanted to be doing."

This subject's main problem was that she had forgotten to change the input cells' guards. Her formulas were all correct on the first try. This is an example of how asking the end user to input extra information can result in complications when they do it incorrectly or inconsistently.

She applied three test cases; a minimum, a middle, and a maximum. Actually, the guards could have been used to know how the formulas would behave under the extreme conditions, except for the *letter_grade* formula. She expressed a strong desire to test nonextremes. It is unknown exactly what she was thinking when she

stated that the minimums and maximums don't take as much math. Could it be that the computer just "knows" that 0 deg. C = 32 deg. F and that 100 deg. C = 212 deg. F and so those inputs don't go through the normal formula path? Is that what she meant by "default?" She tested the extremes in the Grades problem. Maybe they wouldn't work by "default" to her.

Subject G3 also mentioned that to be completely confident in the Grades spreadsheet she would compare the spreadsheet's results to some pre-done results in a teacher's gradebook.

17. "... probably taking a gradebook and just pulling out three samples and going through and putting in the scores. And then doing check figures with pre-done figures."

So, subject G3 would like independent corroboration that her spreadsheet is correct. She wanted to do this with realistic scores that she already knew the results for.

APPENDIX N
GUARD GROUP SUBJECT G4

input_temp	a	b	output_temp
45	Check	Check	Check

Table N.1: Guard Subject G4's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
0	0	0	0	0	0	0	Vis	Vis		
50	50	100	50	100	50	100	Chk	Chk	Chk	Chk (C)

Table N.2: Guard Subject G4's Grades Test Cases

N.1 TEMPERATURE PROBLEM

Subject G4 started out to do this conversion using just two cells. So, she multiplied by 9/5 in cell *a*. In cell *b* she said she was going to add 32, but she typed a '*' symbol instead. She noticed her mistake immediately because the value in cell *b* was very large, 2592. She didn't say anything about the propagated guard range.

1. "Oh, 'cause + 32." "... that looks a little better. OK. And oh-oh, there's a conflict. I don't want that to be 'b' divided by 9."

She decides to use all three cells to do the conversion. All three cells are correctly edited to accomplish the task in three cells.

2. "Oh and I still have a conflict. It says that it should have a range of 32 to 212, which is right. So, I am wrong somewhere. OK, let's see, I still say 0 to 100." "... I only want it to be from 32 to 212."

Subject G4 changes her guard to agree with the Forms/3 guard. The conflict goes away. When she proceeds to testing, she checks the math in each formula with her input value of 45.

3. "45 times 9, that looks right so I am going to click and say that that's right."
4. "... that divided by 5, and that's right."
5. "And this one I said that plus 32, which is also right."

N.2 GRADES PROBLEM

The first thing she does is change the range on all the input cells to be [0 100]. Then she says, but does not click, that *hw_total* and *exam_total* are still right. She says this without ever having opened their formulas. She notices the conflict on the *course* cell. The subject opens the *course* cell formula and then the guard. The Forms/3 number line says [0 700] because no weighting has been done yet.

6. "Oh, because I still say it should be 0 to 100 and it should be 0 to 700."

She modified her guard to agree with Forms/3. Then she got to work on the weighting. First she put the weights in *hw_total*. An error dycon popped up when she was through because she left out a multiplication sign. The error dycon propagated to the *course* and *letter_grade* cells. She didn't notice any of the error messages. Because of the syntax error in the formula, the system could not propagate a guard onto either *hw_total* or *course*. Therefore, there is only a user

entered guard on *course*, i.e. no conflict. She then modified *exam_total* without mistake.

Now she sees the error dycons. After awhile, she notices that the guard is missing from *hw_total*. She discovers her missing '*' sign and corrects the problem. Now Forms/3 can propagate guards and a conflict appears on the *course* cell. Our subject agrees with Forms/3 that the guard on *course* is [0 100].

7. "... because it (*course*) can only go to 100%."

She put values into the input cells that she said were "easy numbers." She checked off the intermediate cells by checking the math and using local testing techniques. When she tested *exam_total*, the math got a little hard to do in her head, so she said she was just going to trust that one.

8. "And 72.5 is a 'C.' Yeah, that's right."

This subject was the only Guard subject who dealt with the assertion conflict before editing the formulas. She didn't jump to the assertion conflict as soon as it happened, though. In fact, she first finished putting guards on the rest of the inputs. She also visually checked the *hw_total* and *exam_total* cells before doing anything with the conflict. Yet, she looked into the assertion conflict before working on her formula edits. After completing the Grades problem she commented on why she changed her guard to the range [0 700].

9. "... because I wasn't thinking in terms of percent, which is what it is now, for the course - 100%. I was just thinking in terms of adding all the homeworks and quizzes and everything up. So I wasn't thinking, I forgot that I was going to have to multiply each homework by whatever percent."

Subject G4 wasn't looking for, and did not notice, the guards changing after she 'Applied' her formulas. The problem description stated the expected ranges for *hw_total*, *exam_total*, and *course*. But she didn't watch for those to appear.

APPENDIX O
GUARD GROUP SUBJECT G5

input_temp	a	b	output_temp
93.3333	Check	Check	Check
100			Check

Table O.1: Guard Subject G5's Temperature Test Cases

hw1	hw2	hw3	quiz1	quiz2	midterm	final	hw_ total	exam_ total	course	letter_ grade
100	0	0	0	0	0	0	Vis			
	100	100					Chk			
			100	100	100	100		Vis		
						0		Vis		
						100		Chk	Chk	Chk (A)
		75			0					Vis
		73								
		74					Chk	Chk	Chk	Chk (C)
					-90	-90				Chk (F)
					29	55				Chk (C)
					0	32				Chk (D)
					100	100				Chk (A)
					91	76				Chk (B)

Table O.2: Guard Subject G5's Grades Test Cases

O.1 TEMPERATURE PROBLEM

Subject G5 entered an input temperature of 93.3333 because when the spreadsheet was given to him, it converted 200 deg. F to 93.3333 deg. C. So he expected a correct conversion to output 200. He didn't do the modification correctly and immediately saw that there was a problem.

1. "It didn't come out to 200 and, I mean there's, right here there's a conflict."

He opened up his guard and changed his number line to 32 to 212. The conflict persisted because with his incorrect formulas, Forms/3 thought that *output_temp* had the range [57.6 237.6]. The problem was that he is added 32 at the beginning of the formula rather than at the end. Later he incorrectly multiplied by 5 and divided by 9, still adding 32 at the beginning. Finally he gets the order of operations right and the result is close to what he is looking for, 199.99994.

2. "It rounded it off. I guess we'll take that."
3. "It's in the ranges and there's no conflict bars."
4. "... it had a conflict so you know that something's wrong. "I mean, I knew anyways because I'm looking at the, I knew that I ... what value I wanted to come out. And if it doesn't come out to anything near it, I know it's going to be wrong. But if you didn't know, it would be nice to have that conflict there."

He tested one more value, 100, and predicted an outcome of 212. Then, he said that he wanted to test one more thing. He entered 100.000000001 to test the guard. He was ready to release his spreadsheet.

At the beginning of this spreadsheet modification, subject G5 started his formula edits by incorrectly adding 32 to *input_temp* in cell *a*. Yet, right after this edit, he tested cell *a* and pronounced it correct because he just checked the math. Yes, it added 32 to the previous cell. He didn't do any more testing until he got

nearly his expected value out. Then, he retested using local testing and checking the math until he reached the final cell, which he tested globally.

O.2 GRADES PROBLEM

He edited *hw_total* first with the correct weights. Then he changed all his input guards to have the range [0 100]. Then he tested the *hw1* guard with inputs -9 and 101. After correctly editing *exam_total*, subject G5 began testing. Some of it was visual, meaning that he would look at the value and say it was right without clicking in the decision box. He happened on a combination of inputs that yielded exactly 80 in the *course* cell and 'B' in the *letter_grade* cell.

5. "That's kinda good right there. It's like right on the cusp."

He lowered the *hw3* score slightly to see if *letter_grade* dropped below a 'B.' He checked off all the formula cells because they looked right.

6. "They look right." "I mean if you look and he got 100, 100, and a 74, that's going to be pretty high. And it's (44.8) pretty high in that range (0 to 50). I figure the computer's going to make the right calculations ..."

He then lowered the *final* and *midterm* cells each to -90. He checks off the 'F' letter grade, but is bothered by what he sees in *exam_total*. It has a value of –7, which is circled with a red oval. The cell itself has a blue border.

7. "Well, I don't understand. It picks out the errors."
8. "The cell's value is wrong, kind of. I mean, it's right, but it's wrong, I guess." "It picks up the fact that these are errors."
9. "But why is it a blue cell? That's kind of weird."

10. "Given the information, it's right. It's wrong, though. It's not possible."

Subject G5 tried to test the *exam_total* cell. But he couldn't make up his mind as to whether or not the cell was correct. He checked, undid the check, X'ed, undid the X, X'ed, then checked. Clearly he was confused by the combination of testing and guard feedback, specifically the red and blue colors on the same cell.

He entered nonnegative values in for *midterm* and *final*. He checked off the 'C' letter grade for the second time. He was asked what made him check off a 'C.'

11. "I'm trusting that these (*hw_total* and *exam_total*) are calculating the right values." "... it's in the range that it's supposed to be." (in-between 70 and 80)

He went through other grades, some of them were repeats of previously tested situations. He stopped testing once the indicator reached the 100% tested level. Basically he would check off a letter grade because of the value of the *course* cell; local testing. He was trusting that *exam_total* and *hw_total* formulas were doing the arithmetic correctly. He could not predict from the inputs what the letter grade would be.

This subject said that he didn't like that Forms/3 propagated an out of range value, referring to the time that *exam_total* contained -7 because *midterm* and *final* were negative. He noted that if he were only looking down at the *course* cell, which had no violation ovals, he wouldn't know that bad data had contributed to that cell. At least he wanted it indicated somehow that out of range data was upstream. He liked the idea of using an asterisk after the value like they do in some sports record books to indicate an irregularity in the accomplishment such as extra games in the season.