AN ABSTRACT OF THE THESIS OF

Cory Kissinger for the degree of Master of Science in Computer Science presented on May 19, 2006.

Title: Supporting End-User Debugging.

Abstract approved:

_____

Margaret M. Burnett

Although researchers have begun to explicitly support end-user programmers' debugging by providing information to help them find bugs, there is little research addressing the right content to communicate to these users. The specific semantic content of these debugging communications matters because, if the users are not actually *seeking* the information the system is providing, they are not likely to attend to it. This thesis reports a formative empirical study that sheds light on what end users actually want to know in the course of debugging a spreadsheet, given the availability of a set of interactive visual testing and debugging features. Our results provide insights into end-user debuggers' information gaps, and further suggest opportunities to improve end-user debugging systems' support for the things end-user debuggers actually want to know. Following up on those suggestions, we then present the design and implementation of a solution aimed at helping to close some of those information gaps.

Supporting End-User Debugging

by

Cory Kissinger

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 19, 2006

Commencement June 2007

Master of Science thesis of Cory Kissinger presented on May 19, 2006

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

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.

Cory Kissinger, Author

ACKNOWLEDGEMENTS

I would like to first thank my major professor Dr. Margaret Burnett. Without her encouragement, wisdom, and vision this work would not have been successful. Her positive outlook and quality as a researcher were an inspiration.

Additionally, chapters two through five were part of a joint effort to produce the work presented in [21]. The researchers who helped to produce that paper put in a tremendous effort in each of their roles. I thank Dr. Simone Stumpf and Dr. Mary Beth Rosson for their expert opinion that guided so much of the experiment design and for their immaculate writing ability. For dutifully completing many of the less appealing tasks and following up on every lead I thank Neeraja Subrahmaniyan and Dr. Sherry Yang. I would also like to thank Valentina Grigoreanu and Vaishnavi Narayanan for assisting with the experiment. Lastly, thanks to Laura Beckwith, for her commitment to research, expert advice and sense of humor.

For their consultation outside of the research group and their help in fostering some of the original thoughts that sparked this work I would like to recognize Andrew Ko and Dr. John Pane.

Other members of our research group assisted my research in countless ways, providing everything from a smooth running implementation to a few laughs—thank you Flora Tan, Joey Lawarance, Jason Daggit, and Robin Abraham.

Finally, thanks to my other two committee members, Dr. Curt Cook and Dr. Rajeev Pandey for providing an open door where I knew I could come for answers.

This work was supported in part by the National Science Foundation under Awards ITR-0325273 and ITR-0405612.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDIX TABLES

Supporting End-User Debugging

1 Introduction

Research on end-user programming has, in the past, concentrated primarily on supporting end users' creation of new programs. But recently, researchers have begun to consider how to assist end users in debugging these programs (e.g., [1, 10, 22, 25, 36, 37]).

Support for end users in debugging tasks is often *problem-oriented*; the system tries to discover candidate bugs, communicate these to the users, and provide user interaction mechanisms to explore and correct the bugs. The communication about the bugs may be delivered through diagrams, color highlighting, or similar devices.

Because the user may not understand how to respond to such information displays, a debugging support system may also provide *feature-oriented* information that explains how to interpret or use the debugging features. Feature-oriented information is typically provided by user interface mechanisms that are tied to the feature in question, such as pop-up tool tips, linked help pages, video demonstration snippets, and so on. In this paper, we refer to the collection of support communications, both problem-oriented and feature-oriented, as the system's *explanations*.

Some existing debugging explanation techniques for end-user programmers have been empirically linked to debugging success [10, 22, 36, 37]. However, many of these empirical results are so focused on the success of particular features, they do not provide much general guidance to future designers of end-user debugging support, such as the semantic-content of what needs to be explained, when, and in what context.

However, a few studies do provide some general guidance for end-user debugging explanations. Natural Programming studies for event-based Alice programs [12] revealed that 68% of the questions asked by the participants (HCI students with varying amounts of programming background) during debugging in that language were "why did" or "why didn't" questions [22]. The Surprise-Explain-Reward strategy [40] has been studied in the spreadsheet paradigm; the work on this strategy

provides general guidance regarding interruption styles for communicating about end-user debugging situations [35] as well as effective reward communications in these situations [36]. Finally, because end users may not have experience with debugging support tools, they may be forced to learn about these features as they work, suggesting that studies of what online learners want to know may be helpful (e.g., [3, 30]).

Determining the semantic content needed by an end user when debugging might seem straightforward. For instance, a system could simply describe all visible features and feedback; this is a common approach to information system design. In our research prototype designed to support end-user debugging, explanations such as these have indeed been created for all visible features. We have put significant research into the semantic content of the explanations, refining them on the basis of both theory (the model of Attention Investment [9] and Minimalist Learning Theory [11]) and empirical work. Despite these efforts, it appears that the explanations are not answering what users want to know. For example, one user in a recent study commented as follows [4]:

> Interviewer: "Weren't the tool tips helpful?"
> S3: "Yeah, they were good but sometimes I didn't find the answer that I wanted…I needed more answers than were present."

Herein lies the problem. Little is known about what information end-user debuggers such as S3 actually want to know.

This research consists of two parts. The first part builds upon previous works to help fill a critical gap in what is known about end-user debugging support: the semantic content of what should be explained to end users to support debugging. To explore this kind of information, we conducted a formative analysis of *information gap instances*—utterances expressing an absence of information—expressed by end-user programmers working on spreadsheet debugging tasks. During the experiment, participants interacted with a spreadsheet environment that contained visual features providing problem-oriented debugging information (e.g., visual colorings of cells that

need testing), but that provided no feature-oriented explanations to help users make profitable use of them. Within this debugging context, we investigated the following research question:

When debugging, what do end-user programmers want to know?

In the second part, we follow up on the first part's results regarding both the content as well as the presentation style of the explanations. Using this information, we present the design and implementation of an explanation approach aimed at complementing a traditional feature-oriented approach, which also involved two steps. First, we generated a list of design constraints that an ideal explanation approach would meet. The constraints were derived from both the results of our formative study and also from related literature. We conclude with the design and implementation of a solution grounded by these constraints.

2 Experiment

The experiment procedure was a think-aloud using pairs of participants. The goal was to allow the participants at least a possibility of succeeding at their spreadsheet debugging, so that they would stay motivated, but without including explanations that might bias the content of the participants' information gaps.

To achieve this balance, we removed all feature-oriented information about how the debugging features worked. We then administered a tutorial to give just enough instruction to our participants to be able to perform basic functions in the particular environment (a research spreadsheet system). Finally, when the participants began their tasks, we removed the only remaining source of support from the room, the researcher himself. The participants were recorded (video and audio) and their screen state was continuously captured along with all instant messenger dialogue (explained below). Figure 1 shows what the researcher observed remotely.

With so little information, participants could have become "stuck," at which time their think-aloud verbalizations would cease to be useful. To avert this situation, we provided mechanisms for the participants to obtain information. Although they had both received the same training, the most accessible information to a participant was his or her partner. This encouraged them to keep talking to each other, which turned out to be the primary way they worked through their information gaps.

A slightly less accessible, but potentially more valuable, source was an instant messenger dialogue between the pair and the researcher, with which the participants could ask questions. Since the researcher was out of the room, the questioner had to include relevant context information, avoiding simple "What's that?" questions. The cost of waiting for the researcher to answer this sort of question (typically 10 seconds), made discussion between the pairs less costly than using the instant messenger, in terms of time and effort. Researcher responses were restricted to the set of feature-oriented explanations that had been removed from the environment for purposes of the experiment. The researcher could also send a hint if the participants

expressed confusion about a particular feature and refused to move on. Pairs typically received one such hint. The participants also had three "wild cards," which could be used as a last resort, to bring the researcher back into the room to provide a hint on how to make progress. (The participants rarely used the wild cards and only occasionally used the instant messenger.)

2.1 Participants

We chose the pair think-aloud protocol because it is particularly well suited to eliciting participants' verbalizations of problem-solving thoughts [19, 20]. This set-up also creates a different social context than for individuals working alone, but since collaborative debugging among spreadsheet users is extremely common [26], it does not introduce validity concerns. Because we wanted participants to feel comfortable talking together, we recruited only *pairs* of participants. This mechanism ensured that each pair already knew each other.

A pre-experiment questionnaire recorded participant background data (Appendix D contains all questionnaires used in the study). Eleven of the fourteen participants were business majors. The other three were in education, industrial engineering, and nutrition, none of whom were paired with each other. None of the participants had programming experience beyond a first level programming course. Gender was distributed equally, with two male-male, two female-female, and three male-female pairs.

2.2 Environment

The debugging features that were present in this experiment were a subset of WYSIWYT ("What You See Is What You Test"). WYSIWYT is a collection of testing and debugging features that allow users to incrementally "check off" or "X out" values that are correct or incorrect, respectively and provide visual feedback [10]. In WYSIWYT, untested cells have red borders. Whenever users notice a correct value, they can place a checkmark (√) in the decision box at the corner of the cell they

observe to be correct. As a cell becomes more tested, the cell's border becomes more blue. (Figure 1 includes many cells partially or fully tested.)

Instead of noticing that a cell's value is correct, the user might notice that the value is incorrect. In this case, instead of checking off the value, the user can X-out the value. X-marks trigger fault likelihood calculations, which cause the interior of cells suspected of containing faults to be colored in shades along a yellow-orange continuum.

In addition, arrows that allow users to see the dataflow relationships between cells also reflect WYSIWYT "testedness" status at a finer level of detail. The optional dataflow arrows are colored to reflect testedness of specific relationships between

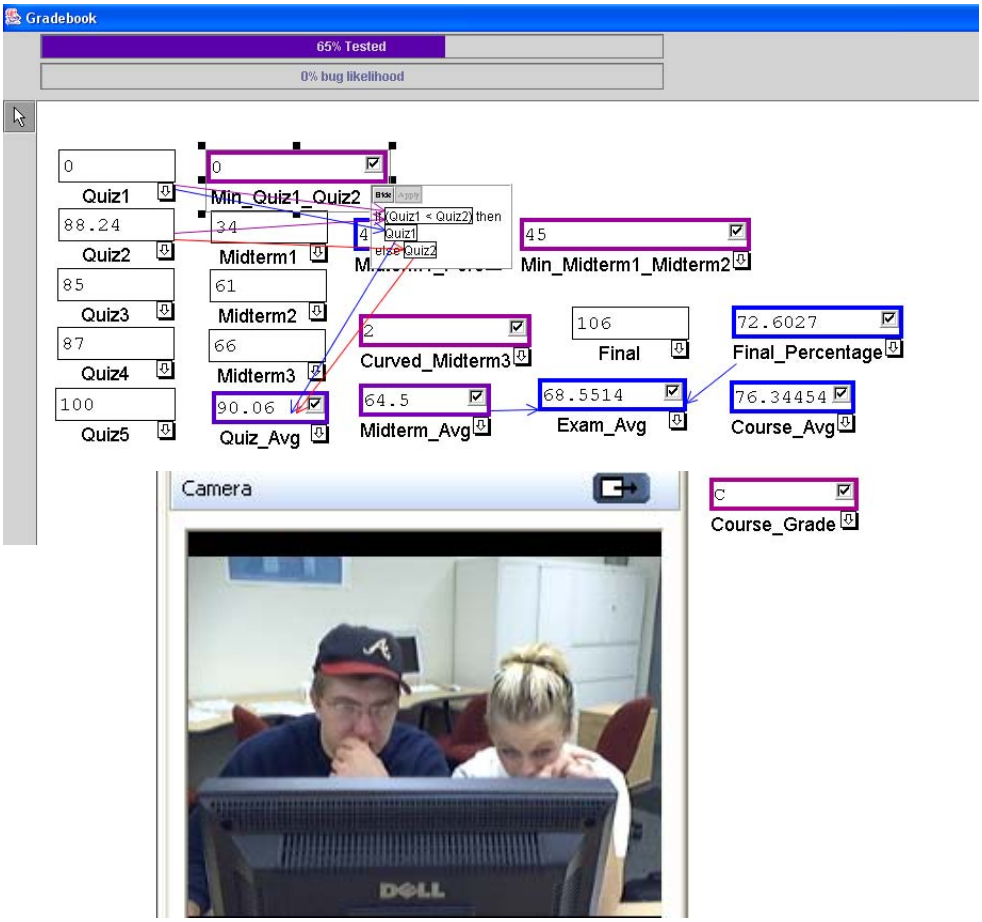


Figure 1: Experiment data capture example.

cells and sub expressions. In Figure 1, the participant has turned on the arrows for the Min_Quiz1_Quiz2 and the Exam_Avg cells.

2.3 Tutorial[1]

The goals of the tutorial were to familiarize the participants with the think-aloud procedure, explain pair-programming guidelines, and to familiarize participants with the environment enough to proceed with their debugging task.

The tutorial began with a think-aloud practice where the pair verbalized a task that they had recently worked on together, namely finding their way to the experiment. The researcher also provided basic pair-programming guidelines. Specifically, the participants were told they would switch between two roles: the *driver*, controlling the mouse and keyboard, and the *reviewer*, who contributes actively to problem solving. In the experiment, they switched roles every ten minutes.

The brief tutorial on the environment was hands-on, with the pair working on a sample spreadsheet problem together at the same machine. Participants learned mechanics of changing input values and editing formulas, as well as mechanics of the unique actions available in the environment: namely, placing checkmarks, placing X-marks, and turning arrows on and off. For example, participants were instructed to "middle-click" on a cell to bring up the cell's arrows. However, the tutorial did not explain how to interpret the visual feedback they received as a result.

2.4 Tasks

We asked participants to test two spreadsheets, Gradebook (shown in Figure 1) and Payroll (Figure 2). For each task, the participants were provided with an untested spreadsheet and a description of the spreadsheet's functionality. They were also provided with an example of the expected output values for given inputs

These spreadsheets were each seeded with five faults created by real end users. We made use of the spreadsheets and faults of earlier experiments [6, 8, 36] that were

[1] Complete tutorial can be found in Appendix C.

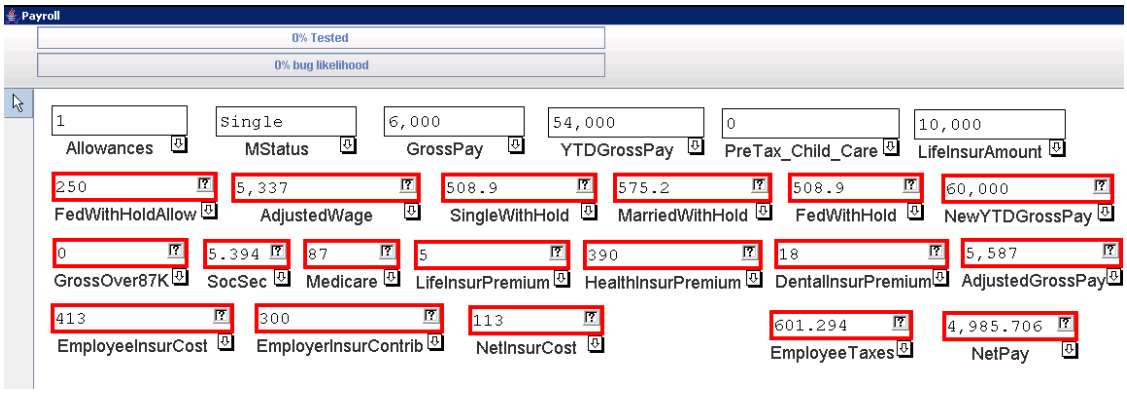


Figure 2: Screenshot of Payroll spreadsheet task.

created as follows: Three end users were provided with the following: (1) a "template" spreadsheet for each task with cells and cell names, but no cell formulas; and (2) a description of how each spreadsheet should work, which included sample values and correct results for some cells. Each person was given as much time as he or she needed to design the spreadsheet using the template and the description.

From the collection of faults left in these end users' final spreadsheets, five were chosen that provided coverage of the categories in Panko's classification system [29] (based upon Allwood's classification system [2]). Under Panko's system, mechanical faults include simple typographical errors or wrong cell references. Logical faults are mistakes in reasoning and are more difficult to detect and correct than mechanical faults. An omission fault is information that has never been entered into a cell formula, and is the most difficult to detect [29]. The Gradebook spreadsheet was seeded with three of the users' mechanical faults, one logical fault, and one omission fault, and Payroll with two mechanical faults, two logical faults, and one omission fault. Additional information pertaining to the spreadsheets and their faults is included in Appendix E.

Participants had time limits of 20 and 40 minutes for Gradebook and Payroll respectively. These simulated the time constraints that often govern real-world computing tasks, and also prevented potential confounds, such as participants spending too much time on the first task or not enough time on the second task. The

participants were given more time on the Payroll task because it was the more difficult of the two due to its larger size, greater length of dataflow chains, intertwined dataflow relationships, and more difficult faults. All participants performed the (easier) Gradebook task first to allow a gradual introduction to the environment before the more challenging Payroll task. The participants were instructed, “Test the … spreadsheet to see if it works correctly and correct any errors you find.”

3 Methodology

The methodology we adopted consisted of four main activities: the segmentation of the data into topic-related units of utterances, the development of a coding scheme through a bottom-up organization of the units, the application of the codes to the data, and the calculation of agreement measures to evaluate the stability and robustness of the resulting coding scheme.

3.1 Segmentation of the Data

The primary data were audio recordings of participants' utterances, synchronized with video recordings of their physical behavior and screen states. To create an integrated data record, the audio recordings were transcribed and supplemented with context obtained from the video and screen data (e.g., gestures and actions). Because a single utterance alone does not allow an analysis to be sensitive to common context and thread of discussion, the transcripts were segmented into *stanzas* [15]. A stanza, typically about 8-15 lines, is a unit of utterances that occurs between shifts of topic (see Figure 3 for an example). Appendix A includes a larger sample of transcribed and coded data as well as a link to the complete dataset.

F. Let's change everything. [tries changing formula]
E. Yeah, but we got the right answer.
F. Did that change the answer at all?
F. Oh wait, did I change the symbol? [changes the formula]
F. Oh, now we're down to 30 percent tested.
F. I wonder if I go like that- [changes the formula back]
F. Oh no, crazy.
F. Oh, I guess now it's just this again. [checks off cells that changed]
F. I don't get how you get to 100%, it's like a test you can't pass. Every time I do this it gets lower.
E. Yeah, I don't know.
F. Well, that's confusing.
Switch places. (.)

Figure 3: A stanza in which participants E and F discuss the debugging strategy of "changing everything". Notation: [actions] denote actions taken, (.) a pause in speaking, and *italics* the researcher's instant message to the participants. See Appendix B for complete Transcription Conventions.

3.2 Deriving the Codes

The goal of the codes was to support analysis of participants' information gaps. We considered an information gap to have occurred when a participant asked a question, stated a tentative hypothesis, expressed surprise, made a judgment about whether an information gap was present, or provided an explanation to his or her partner (implying that the partner had an information gap). We refer to such utterances as *information gap instances*.

The research literature does not report coding schemes that are directly applicable to the information gaps experienced by end-user debuggers. Most studies of information gaps focus on users in learning or tutorial situations (e.g., [3, 30]). In contrast, we are interested in the just-in-time learning users undergo to enhance their *productivity*, i.e., to make progress in solving a problem; we will return to the relationships between our coding scheme and others' in Chapter 5. In trying to increase productivity, the user must balance the costs of learning a new technique or feature—which may or may not be relevant to a task—against its potential benefits for task performance. In these circumstances, learning may be just one of a set of competing goals.

To develop a coding scheme that matched our aims, we (the authors of [21]) first grouped stanzas from two of the transcripts into an affinity diagram[2], adjusting the concepts and relations as we progressed. This grouping process allowed us to focus on *types* of information gaps, as we compared and organized information gap instances according to their semantic content such as a question about what might be a suitable strategy. As types of information gaps were identified, descriptions and example utterances for each candidate coding category were collated. The coding scheme was applied to one transcript repeatedly with different coders each time. The codes were refined to be less ambiguous after each application until acceptable

[2] A group decision-making technique designed to sort a large number of items into "related" groups, from the perspective of those doing the sorting.

agreement was achieved (above 80%, see next section for calculation of agreement) across coders. The 10 codes we identified in this fashion are described in Table 1.

3.3 Application and Agreement

Two of the authors independently coded all of the transcripts. Multiple codes were allowed per stanza, as there may have been multiple information gap instances contained in a group of related utterances. The coders discussed their initial coding

Table 1: The coding scheme.

Code	Description	Examples
Feature/ Feedback	Question or statement expressing general lack of understanding of the meaning of a specific visual feedback or action item, but with no goal stated.	"So with the border, does purple mean its straight-up right and blue means it's not right?"
Explanation	Explanation to help partner overcome an information gap. The explanation may be right or wrong.	"<border color> just has to do with how much you've been messing around with it."
Whoa	Exclamation of surprise or of being overwhelmed by the system's behavior.	"Whoa."
Help	Question or statement explicitly about the need for additional help.	"Help."
Self-Judgment	Question or statement containing the words "I" or "we," explicitly judging the participant or the pair's mastery of the environment or task.	"I'm not sure if we're qualified to do this problem."
Oracle/ Specification	Question or statement reasoning about a value and/or a formula.	"Divided by 10? I don't know...I guess it should be times 10."
Concept	Question about an abstract concept, as opposed to a question about a concrete feature/feedback item on the screen.	"What does 'tested' mean?"
Strategy Question	Explicitly asks about what would be a suitable process or what to do next.	"What should we do now"
How Goal	Asks how to accomplish an explicitly stated goal or desired action. (An instance of Norman's Gulf of Execution [27].)	"How do you get 100%?"
Strategy Hypothesis	Suggests a hypothesized suitable strategy or next step to their partner.	"Let's type it in, see what happens."

decisions and made changes if they agreed that a code had been inadvertently overlooked or misapplied.

A widely used rule of thumb is that 80% agreement or higher between coders indicates a reasonably robust coding scheme. Because more than one code could be placed on a stanza, the calculation of agreement for a particular stanza required comparing two sets of codes (one from each coder). The percentage of agreement for a stanza was calculated by dividing the size of the intersection by the size of the union. For example, if one rater coded a stanza {Help, Self-Judgment} and the second coded it {Strategy Hypothesis, Self-Judgment}, then the agreement for that stanza would be |{Strategy Hypothesis, Self-Judgment} $\cap$ {Help, Self-Judgment}| / |{Strategy Hypothesis, Self-Judgment} $\cup$ {Help, Self-Judgment}| = 1/3 = 33%. The average of all 425 coded stanzas resulted in 90% agreement.

4 Results

Table 2 lists the frequencies of each type of information gap found in the 425 stanzas, and Figure 4 shows the distribution over time. (One pair was excluded from the time graphs, since their overall time was considerably less than that the others'.)

Table 2: Code frequencies.

Code	Count (Percent of Total)
Features/Feedback: Feature/Feedback (questions)	77 (10%)
Explanation	48 (6%)
Big Information Gap: Whoa	14 (2%)
Help	23 (3%)
Self-Judgment	67 (9%)
Oracle/Specification	316 (40%)
Strategy: Concept	8 (1%)
Strategy Question	39 (5%)
How Goal	20 (2%)
Strategy Hypothesis	169 (22%)

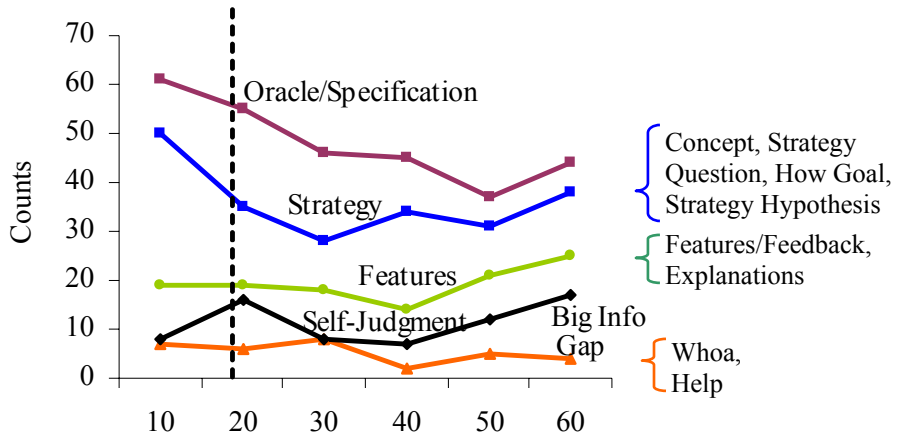


Figure 4: Code frequency within each 10-minute interval. Task 2 (Payroll) began after 20 minutes. Similar codes were grouped to aid discussion of results.

4.1 Questions and Explanations About Features and Feedback

A widely used approach to introducing users to a new interface is to provide information about the meaning of features: both user actions available, such as options user can select, and feedback items they might receive, such as red underlines under misspelled words. This information is often contained in tool tips and/or online help systems organized by feature.

In our study, the information gaps that are satisfied by this kind of explanation were observed as questions participants asked about what specific features mean, such as "What does the purple border mean?" (type Feature/Feedback), and as explanations of a specific feature's meaning by one participant to the other, such as "I think the purple means it's wrong" (type Explanation). (Explanations suggest an information gap because they imply that one participant thinks the other is lacking this information.)

As Figure 4 indicates, feature-oriented gaps were highest at the end of the experiment. Still, as can be seen in Table 2, the combined percentage of feature-oriented information gaps was a surprisingly low 16%. Recall that, as business students with multiple years of spreadsheet experience, the participants had fixed bugs in spreadsheets before, seemingly leaving only orientation to the unfamiliar interface as a barrier. Yet, few of their information gaps were about the interface, despite our removal of feature-oriented support.

Practical implications: End-user debugging explanation approaches that center mainly on the meaning of the system's features and feedback—a common strategy in online explanation systems—would address only a fraction of what our participants wanted to know.

4.2 Big Information Gaps: Whoa! Help!

"Whoa!" Approximately 2% of the responses expressed surprise and confusion at feedback that had just occurred on the screen. Information gaps of type Whoa were

often in response to several visible changes occurring at once, such as turning on dataflow arrows (Figure 1). Another 3% of the information gaps explicitly expressed a general need for help (coded Help), implying that there was a need for more information to even be able to verbalize a more specific question. Both of these types of questions expressed a lack of clues about the current situation or what to do about it. These results are good reminders that sometimes when a user needs explanations, a more specific question does not readily occur to them. In our study, this happened 5% of the time.

Practical implications: A look at the timing of the Whoa and Help instances provides some guidance as to how a debugging explanation system might address this type of information gap. First, note in Figure 5 that the general requests for information (type Help) were greatest at the beginning of the first task when little was known about the environment and task, and at the 50-minute point (30 minutes into the more difficult spreadsheet). This timing suggests that end-user debuggers may need more broad-based support at the beginning of the task and in moments of particular difficulty, such as suggesting ideas to help the users (re-)connect to features or strategies that may help them.

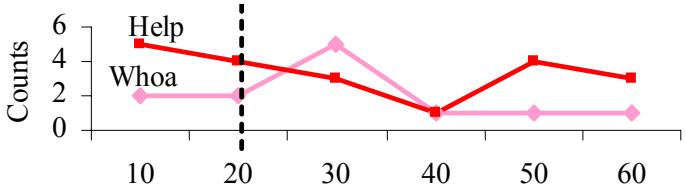


Figure 5: Frequency of Help and Whoa codes within each 10-minute interval.

Second, as Figure 5 shows, type Whoa occurred mostly in the middle of the experiment. At this time participants had enough experience to form an early mental model of how the environment worked. However, the application of this model during the more difficult task may point out a serious misconception. According to research into the psychology of curiosity [24], moments of surprise such as these are opportune times for explanations, as people curious about such surprises seek to

satisfy the information gap that led to the surprise. An explanation system that kept track of the amount the user has used the features and the amount of recent feedback may be able to determine whether a generic "help!" button push is more likely to be the result of a type Help versus a type Whoa information gap. In our study, for the Whoa type of information gap, a look at the system state sometimes revealed the likely cause of confusion. In these cases, a context-sensitive explanation system might successfully respond to Whoa requests by providing assistance on the most recent feedback.

4.3 Self-Judgments: Am I smart enough to succeed at this task?

Of the participants' information gap instances, 9% were self-judgments of their own mastery of the system or of the debugging task, suggesting that self-judgment was a significant factor in their cognitive processing as they worked on the bugs. These self-judgments are instances of metacognition, in which a learner monitors the success of his or her own learning processes [14]. Metacognitive activity is well-established as an important influence on learning and understanding [38].

These judgments also provide a view of the participants' self-efficacy. Self-efficacy is a person's belief that they will succeed at accomplishing a specific task, even in the face of obstacles [5]. According to self-efficacy theory, the amount of effort put forth is impacted by a person's self-efficacy. In our own work with self-efficacy, we have seen it have a significant effect on end users' willingness to use advanced debugging features [8]. In that work, some users' self-efficacy was much lower than warranted, particularly among females. In previous studies as well as in the current one, we have also observed examples of participants overrating their own performance, saying things like "We did it right" when in fact they had not. Both overrating and underrating performance may point to failures of the system to provide accurate feedback regarding the users' debugging progress or users not correctly interpreting the feedback given.

Practical implications: An effective explanation system that succeeds at fulfilling end-user debuggers' information gaps may also improve the accuracy of users' self-judgments. Due to the effects of self-efficacy and metacognition, this in turn may help increase debugging success simply by helping users persist in their efforts.

4.4 Oracle and Specification Questions: Is this the right value/formula?

In debugging a spreadsheet, it may not always be clear to users whether or not a value is correct. In software engineering, difficulty determining whether a value is right or wrong is called the "oracle problem." The oracle problem is important because its presence weakens many of the user's problem-solving devices such as the power of immediate visual feedback, user tinkering, and testing behaviors. After all, these behaviors are not helpful when the user cannot tell whether the result is right or wrong.

"How do we know if that's right or not?" This information gap instance not only shows one example of the oracle problem occurring, it also expresses a request for information about *how* to decide whether a value is right. A closely related problem that arises in debugging is whether the *formula* ("source code") correctly implements the specifications or, if the user has already determined that it does not, how to make it do so: "So the average, why is it divided by 3?"

In our study, 40% of the information gap instances fell into the Oracle / Specification category. Note that this large fraction of the total set of questions is about the task (debugging), not about the features or the system. This is consistent with Carroll and Rosson's description of the "active user" [11], who focuses much more on the task at hand than on the availability of potentially interesting user interface features.

Practical implications: Some information gap instances of this type centered on a particular cell, such as "We need some more money for this…we're missing $300." Such instances may be well served by an explanation that suggests changes to a spreadsheet to produce a desired output, such as the direction of Abraham and Erwig's goal-based debugging suggestions [1], or by an approach that explicitly supports investigation into the reason for a specific value or event, as with Ko and Myers's Whyline work [22]. Other information gap instances encompassed a larger subset of the spreadsheet, such as "Where is it getting the wrong math here?" One possible solution to this type of question might be to remind the user of narrowing-down techniques such as WYSIWYT with fault localization [10].

4.5 Strategy: What should we do?

Fully 30% of the information gap instances pertained to strategy issues. There were four codes relating to strategy: Concept, Strategy Question, How Goal, and Strategy Hypothesis. The primary type in this group at every time period was Strategy Hypothesis (Figure 6), in which participants actively hypothesized strategies, which they usually proceeded to try out. This again calls to mind the active user—one who seeks mainly information directly pertinent to their goal. Type Strategy Hypothesis alone accounted for 22% of the information gap instances.

Practical implications: Most of the strategy information gap instances were global in nature, rather than being about a particular feature (e.g. "What should we do next?"). Due to the lack of a contextual tie, a feature-anchored explanation such as a

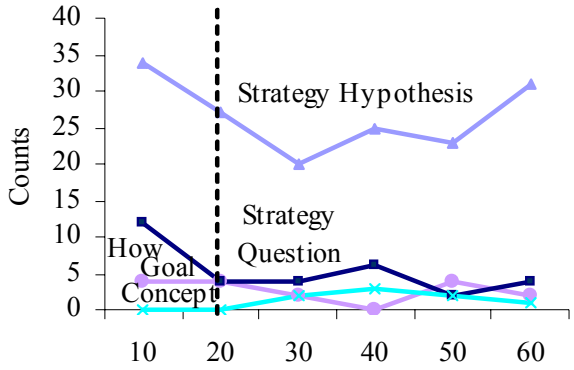


Figure 6: Frequency of strategy codes within each 10-minute interval.

tool tip seems a poor fit for this sort of information gap. Even so, some of the remarks, while global, still had ties to particular features. For example, "What's testing?" (Concept) could, in our setting, be answered in explanations of the testing features, and "How do we get to 100%?" (How Goal) might be answered in explanations of the testing progress indicator (top of the spreadsheet environment in Figure 1).

This group of information gaps presents a good opportunity for improvement in end-user debugging explanations. In some help systems, strategy is addressed in separate tutorials about a system's usage, but this seems an inappropriate solution given the active users our participants appear to be. The key may lie in linking feature-located and feature-centric explanations with broader explanations that tie the use of features into strategic goals. In [7] the use of broader help information in expandable tool tips, including some coverage of strategy, was proposed. This is an example of the "layered" approach to explanations recommended by [13] for use in minimalist instruction aimed at active users; given our observations of participants' active debugging style, this may be a step in the right direction.

4.6 Implications of Co-occurrences

Two code types co-occurred in the same stanza with certain other types an inordinate number of times: Self-Judgments, and Strategy Hypotheses.

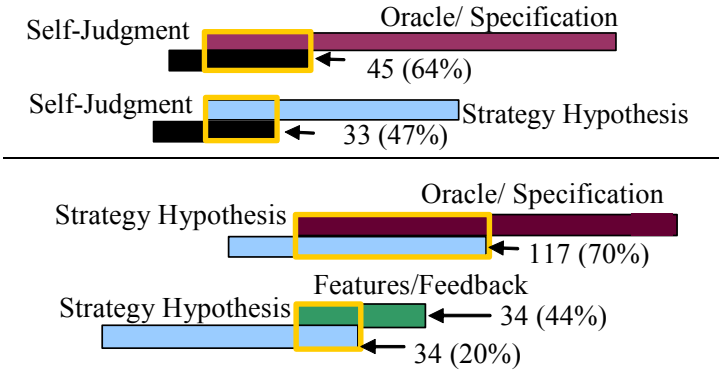


Figure 7: The top two code co-occurrences for (top) Self-Judgments and (bottom) Strategy Hypotheses.

A majority (64%) of the Self-Judgments occurred in stanzas also showing Oracle/Specification information gaps, as Figure 7 illustrates. Also, 47% co-occurred with Strategy Hypotheses. These were far ahead of the third most common co-occurrence, at only 19%, with Features/Feedback, not shown in the figure. (The percentages exceed 100% because more than two codes sometimes occurred in a single stanza.) This suggests that the most appropriate places for debugging explanations to attempt to improve users' ability to self-judge will be in the context of problem-oriented communications and with strategy-oriented communications. In particular, it appears that a system's feature explanations are not likely to be the right context for assisting users in making more accurate self-judgments of their performance.

Furthermore, 70% of the Strategy Hypothesis instances co-occurred with Oracle/Specification instances, implying that participants' main interest in strategy was in applying it to the problem domain, as opposed to building it up with the features as a starting point. The second-ranked co-occurrence was with Features/Feedback at only 20% of the Strategy Hypotheses. However, the flip side of this runner-up was that 44% of the Feature/Feedback information gap instances included Strategy Hypotheses, implying that feature-centric "hooks" to strategy hints would be welcomed by users—but would not alone be enough, since they would still leave 80% of the Strategy Hypothesis gaps unfilled.

5 Comparison to Other Work

Researchers have developed coding schemes for users' questions and comments in settings other than end-user debugging. To help understand what might be end users' unique needs in the debugging context, we first considered and now compare our coding scheme to several others.

We searched the literature for work that coded some form of information gap. The most relevant works (Table 3) centered on questions people asked and barriers they experienced. Anthony et al. [3] analyzed the questions students directed to a simulated algebra tutor. Person and Graesser [30] summarized a number of studies examining human-human tutoring dialogs. Gordon and Gill [17] analyzed questions designed for knowledge elicitation from domain experts [17]. Ko and Myers [23] studied a usage context somewhat similar to our own—problems experienced by novices learning a programming language (Visual Basic.Net).

Prevalence To compare our coding scheme with these, we studied the category description and illustrative examples for each question or comment category, to determine similarity to one or more of our codes. We used a relatively liberal decision rule in identifying overlap—if we could find multiple data instances from one of our own categories that would have been captured by a code in another scheme, we labeled it as a "match."

Despite the variation in information and task context, we identified some overlap for all of our codes except two—Help and Explanation. The lack of overlap for these codes may be partially due to our experimental set-up that reflected collaborative end-user debugging: for example, working with a partner probably encourages users to offer explanations to one another.

Table 3: Overlap of our codes with others' coding schemes.

Our Codes	Anthony et al. [3]	Person/Graesser [30]	Gordon/Gill [17]	Ko/Myers [23]
Goal: Find out what end-user debuggers want to know	Goal: Design intelligent tutor based on student questions	Goal: Design intelligent tutor based on human-tutor dialogue	Goal: Knowledge acquisition from experts for information systems	Goal: Describe barriers in learning a programming language
Feature/Feedback	Interface	N/A	Event, State	Understanding
Explanation	N/A	N/A	N/A	N/A
Whoa	N/A	N/A	N/A	Understanding
Help	N/A	N/A	N/A	N/A
Self-Judgment	N/A	Meta-comment	N/A	N/A
Oracle/Specification	Answer-oriented	Problem-related	N/A	N/A
Concept	Principle-oriented, Definition	N/A	Concepts	N/A
Strategy Question	N/A	N/A	N/A	Design
How Goal	Process-oriented, Interface	N/A	Goal, Goal/Action	Use, Selection, Coordination
Strategy Hypothesis	N/A	Reminding example	N/A	N/A

Instead of asking about specific procedures, our users often seemed to operate at the more abstract level of goal-setting, such as asking about a suitable process to follow (Strategy Question) or making a goal-setting proposal to the partner (Strategy Hypothesis). Together these two codes accounted for 27% of our data, but we found little overlap between these codes and the other schemes. The clearest case of overlap is with Ko and Myers [23], who created a "Design" code to classify novice programmer problems that are inherent to programming and distinct from language mechanisms. These researchers' setting was similar to ours because it contained aspects likely to be unfamiliar to users while also being challenging enough that they sometimes needed help just to identify reasonable goals.

Only one of the other schemes overlapped with the Self-Judgment code (a judgment about the mastery level of "I" or "we"). The importance of reflection about one's own knowledge state (metacognition) in learning and problem solving is well-established [14]. It is not yet clear what sorts of cognitive or social settings are most likely to evoke reflection about one's capacities during problem-solving episodes. Perhaps collaborative work situations, as in our experiment, encourage self- or pair-evaluation as a sort of knowledge calibration mechanism; alternatively it may simply be that other researchers have been less attuned to metacognition and thus have made no analogous distinctions in their coding.

6 Follow Up: Recorded Demonstrations

The results presented above reveal the diversity of information end-user debuggers want to know about. One issue that arose from this data, however, was that our existing presentation style (textual explanations inside of pop up tool tips) would not be able to support this variety of information without heavily burdening the user with reading. We thus began to consider recorded demonstrations as an explanation approach. The strengths of this approach—reported in [4, 18, 28, 32, 34, 39]—seemed likely to provide at least four advantages over static approaches.

First, our users were observed to be very interested in strategy (30% of all information gaps). Strategies, in general, can be described as a series of steps, i.e. a procedure. Palmiter and Elkerton pointed out that recorded demonstrations can be very effective at teaching a procedure [28]: their demonstration groups were faster and more accurate than groups receiving textual instructing when learning a procedural task. They warned, however, that retention was found to be slightly worse in the subjects who received the recorded demonstrations. We do not view this as a major obstacle since performance—not education in the long term—is our primary concern.

Secondly, we found there to be a significant number of information gaps where users judged their own behavior (9% of all information gaps). Research into self-efficacy (see Section 4.3 for definition) has found that it can have a significant impact on users' success [5, 8]. Additionally, Bandura's work with self-efficacy identified one source of a person's self-efficacy as vicarious experience, meaning that seeing others successfully perform a task will help to persuade the user that they too will be successful [5]. Such an experience would be nearly impossible to produce with text alone. A recorded demonstration, though, could display an actual person succeeding at the task the user was struggling with.

The third reason relates to the active disposition of our users. Past research has taken such "active users" into account when constructing textual user manuals [13]. These works primarily follow a Minimalist Learning approach [11], one which we have employed with our textual explanations. However, evidence from the field of video base

instruction suggests that instructions can be given more efficiently with video than with text [39]. This is desirable because if we can provide more efficient instructions we can return our active users to their task more quickly.

Fourth, using our interface while recording the demonstrations will improve the closeness of mapping of our explanations to the activities we would like the users to perform. Closeness of mapping—one of the Cognitive Dimensions of Notations [16]—describes how closely a representation is related to its domain. The Cognitive Dimensions provide a framework for designers whose goal is to ease the mental load placed on the users attempting to make use of the notation.

As a next step, we looked to the work of Plaisant and Shneiderman to advise the design of our demonstrations. Their work with recorded demonstrations provided the guidelines shown in the first column of Table 4 for producing successful demonstrations [32].

Table 4: Guidelines for producing recorded demonstrations.

Constraint or Guideline	Type	Source / Rational
Provide procedural instruction rather than conceptual information	Semantic Content	Plaisant and Shneiderman [32]
Keep segments short (15 to 60 seconds)	Structure	
Ensure that tasks are clear and simple	Semantic Content	
Coordinate demonstrations with textual documentation	Syntax	
Use spoken narration	Form	
Be faithful to the actual user interface	Form	
Use highlighting to guide attention	Form	
Ensure user control	Structure	
Keep file sizes small	Structure	
Strive for universal usability	Form	

Unfortunately, these guidelines are predominately geared towards the syntax, form, and structure of the recorded demonstration, as the second column of Table 4 points out. In order to guide the development of the demonstrations' semantic content we identified additional constraints grounded by our formative study, which are given in Table 5.

Table 5: Design constraints used to guide the content of our demonstrations.

Constraint or Guideline	Type	Source / Rational
Present information in a concrete to general sense	Semantic Content	Concrete information gives a "how to" example, while general information gives advice that they can apply in other cases.
Include head shot of person performing actions	Form	Having an actual person on the screen will give the user someone to relate to, presenting the opportunity for a vicarious experience aimed at improving the user's self-efficacy.
Interpret feedback	Semantic Content	Accurate interpretation of feedback avoids problems in the accuracy of the users' self-judgments.
Mention oracle problem and what to do if values are hard to decide about	Semantic Content	Substantial empirical evidence has shown end-user programmers struggle with this issue (e.g. [21, 31]).
Make benefits clear	Semantic Content	According to the Surprise-Explain-Reward strategy [40] and Attention Investment [9], giving the user a way to judge benefits will have a significant impact on their future actions.
Keep active user in mind	Semantic Content	The users we intend to support are those whose primary goal is working on their task.
Use informal terminology	Syntax	Avoid intimidating vocabulary that could prevent the user from relating to, or identifying with the demonstrator.
Verbalize the reasoning of the speaker	Semantic Content	To keep user from getting lost, the demonstrator will need to think out loud.
Give strategy information	Semantic Content	Based on our think-aloud study, users' information gaps frequently involve strategy.

Based on the goals of an individual demonstration, tradeoffs were made as to which guidelines or constraints were the highest priorities. With these guidelines and constraints in mind, we produced a series of recorded demonstrations.

6.1 Detailed Design and Creation of the Demonstrations

The creation of each recorded demonstrations involved three steps: First the content of the demonstration was written out in text (Figure 8). This provided a rough script for the demonstrator to follow to ensure that we were adhering to our content constraints.

Next, we began to move more away from content and closer towards presentation

Title	Script	Constraints met
Finding Errors	*Start with a small gradebook spreadsheet*	
	Look, in this Gradebook spreadsheet this student has scores above ninety on all quizes and exams, but their Course Grade (*Highlight course grade cell*) is a B so something is wrong here.	hear me reasoning, concrete to general
	In general, if you can tell that some of the values on the spreadsheet are either wrong or right, Xing out wrong values and checking off right values will help you to find errors in the formulas. If you can't tell if a value is right or wrong try changing the values in the uncolored cells.	informal terminology, address oracle problem, gives strategy hint
	Need to try to make this obvious that the cells I check are right and the ones I X are wrong	(maybe make this a constraint for all recorded demos)
	Show a screenshot with an x-mark on a sink cell	
	The coloring of cells' interiors show the system's guesses at which cells are likely to have an error in them. Probably not all of the cells with interior colors have an error, but at least one of them does.	Interpret Feedback, informal terminology
	The more checks and Xs you place, the better the guesses get.	make benefits clear,
	For example I can tell that the value in the ** cell is also wrong, but that the values in the ** and ** cells are correct.	
	So now the system has narrowed our search for the error from ** cells down to **	

Figure 8: Example script for demonstration with matching constraints.

by producing storyboards that illustrated visual aspects of the demonstrations that we wanted to match up with the text in the scripts. The scripts produced in the first step generally contained visual cues at certain points in the dialogue that aided the creation of the storyboards. For example, the circled line of the script in Figure 8 can be seen visually in the third window of Storyboard 1 (Figure 9).

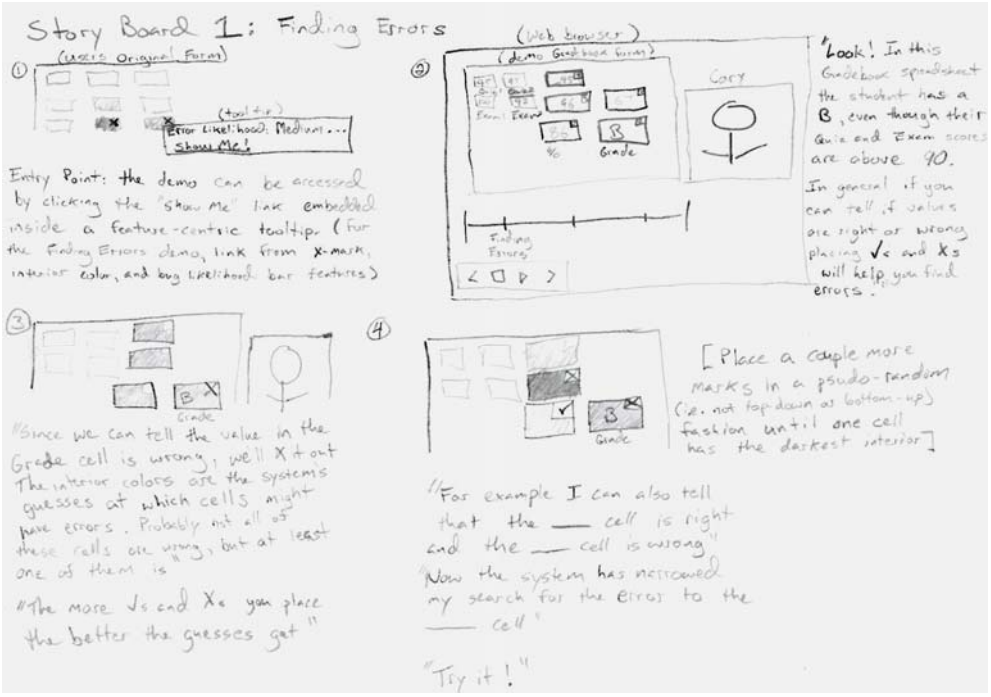


Figure 9: Example storyboard for recorded demonstration.

In the third step—where Plaisant and Shneiderman's guidelines were most useful—we created the final product. Screen plus user audio-visual capture technology (the same that was used to obtain the information gap data during the think-aloud study) was used to create the demonstrations. This allowed us to interleave the recording of the desired actions (performed on the user interface) with the video and audio of the person performing the actions. We experimented with using one demonstrator (male) as compared to two (a male and a female, where one of the demonstrators performed the role of a confused user), ultimately choosing the two demonstrator situation because it more closely fit the rationale behind the "Include head shot…" constraint. The complete collection of storyboards can be found in Appendix F and the demonstrations can be viewed at: http://web.engr.oregonstate.edu/~burnett/Forms3/RecordedDemos/RecordedDemos.htm

Once completed, the demonstrations needed to be integrated into our research prototype and made available to the user. As suggested by the results presented in Section 4.6, we chose to link the demonstrations to feature-centric explanations. Since there was not a one-to-one relationship between features and demonstrations (more features than demonstrations) we linked demonstrations to the explanation(s) that most closely related to the content of the demonstration. For example, the Make Testing Progress

demonstration was linked to the explanation for arrows, the checkmark and to the testing progress bar as they are all primarily used for making testing progress.

The "link" was accomplished by appending a string holding the file name of the related demonstration onto the text of each tool tip as shown in Figure 10.

```
Model
Let tooltipText = what we want the user to see
Let demoFileName = the video file's name
Let internalTooltip = tooltipText + demoFileName

Model sends internal tool tip to front end.
Front end parses internal tool tip back into tooltipText and demoFileName

Front End
Let button = new Button ("Show Me")
Display tooltipText and button

If users pushes button then
        Open browser with ConstantAddress  + "?demo=" + demoFileName
        Where ConstantAddress is on Dr. Burnett's web space
```

Figure 10: Pseudo-code for the integration of recorded demonstrations into our research prototype.

Thus, the web page was displayed as a result of the "Show Me" button being pressed and was passed the file name of the demonstration using query string parameters. A media player was embedded into the web page and programmed to start playing the file found at the location given in the query string. The complete details of this implementation are displayed at:
https://secure.engr.oregonstate.edu/wiki/forms3/index.php/Recorded_Demonstration_Documentation

6.2 Future Work

An empirical evaluation of the recorded demonstrations is needed to ascertain whether they are indeed filling the intended information gaps. This will require at least an analysis of the effect the demonstrations have on users' strategy (similar to that done in [33]), self-efficacy (similar to [7, 8]), and how disruptive they are to the users' primary goal of completing their task (similar to [35]).

We chose a link from a feature-centric explanation as an entry point to the demonstration. This decision was supported by the results of our investigation of co-

occurrences in Section 4.6. However, the evaluation of the recorded demonstrations themselves may also point to a more appropriate entry point.

Lastly, it is not likely that these demonstrations will completely fill the information needs of our users. In fact, any information gaps that we are successful in filling will provide a scaffolding allowing users to advance to the next step where they are likely to encounter new types of information gaps. We hope that supporting this next generation of information gaps will arise as the next challenge to face, as it will indicate that this work has helped the progression of users' problem-solving ability.

7 Conclusion

The pair think-aloud study presented in this thesis was aimed at capturing the information gaps arising for end users in the course of debugging spreadsheets. While further investigation is needed to determine the generality of the results to other settings, there were several implications that seem applicable to a variety of end-user debugging systems. To summarize:

- Unlike what is done in many software systems, debugging explanations for end-user programmers should not be primarily focused on how the debugging features work. In our study, feature-oriented explanations would address only a fraction of what our participants wanted to know.
- The greatest need for explanations fell in the Oracle/Specifications category: figuring out whether a value was right or wrong, whether a particular snippet of code (formula) was right or wrong, and how to fix values and formulas that were wrong. The prevalence of this category points to a need for more research on how to support it.
- The second most common category was Strategy. Strategy information gaps outnumbered feature-oriented information gaps by a 2:1 ratio. To date, there has been almost no research on supporting information gaps of this type.
- Debugging explanations should focus not only on local information gaps, (e.g., pertaining to one cell), but also on global information gaps (e.g., pertaining to an entire spreadsheet).
- When a generic "help" request is made, an explanation system might be able to figure out, from the system state and from the timing of the request, if it is a (re-)connect question versus a feedback-oriented surprise.
- Debugging explanations should strive to fulfill users' needs to self-judge their progress. This category contributed a surprising 9% of the information gaps. Accurate self-judgment matters to debugging effectiveness for both its self-efficacy and its metacognitive implications.

The above results have specific implications for designers of debugging support for end-user programmers, and also identify some open research questions in this area.

Additionally, we have presented the design and implementation of a complementary explanation approach grounded in the results of a think-aloud study. The design of this approach was aimed at exploiting the benefits of recorded demonstrations to address active users' strategy and self-judgment information gaps as well as to improve our explanations' closeness of mapping. The creation of design constraints from the literature and from our study advised the content of the demonstrations.

We hope that following up on these results will help to fill end-user programmers' critical information gaps that currently serve as barriers to the genuine effectiveness of end-user programming.

Bibliography

[1] Abraham, R., Erwig, M., Goal-directed debugging of spreadsheets, *IEEE Symp. Visual Langs. Human-Centric Comp.,* 2005, 37-44.

[2] Allwood, C., Error detection processes in statistical problem solving. *Cognitive Science*, 1984, 413-437.

[3] Anthony, L., Corbett, A. Wagner, A., Stevens, S., Koedinger, K., Student question-asking patterns in an intelligent algebra tutor, *Conf. Intelligent Tutoring Sys.,* 2004, 455-467.

[4] Baecker, R., Showing instead of telling, *In Proc. of SIGDOC,* 2002, 10-16.

[5] Bandura, A., Self efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 1977, 191-215.

[6] Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., Cook, C., Tinkering and gender in end-user programmers' debugging, *ACM Conf. Human Factors Comp. Sys.,* 2006, 231-240.

[7] Beckwith, L., Sorte, S., Burnett, M., Wiedenbeck, S., Chintakovid, T., Cook, C., Designing features for both genders in end-user software engineering environments, *IEEE Symp. Visual Langs. Human-Centric Comp.,* 2005, 153-160.

[8] Beckwith, L. Burnett, M., Wiedenbeck, S., Cook, C., Sorte, S., Hastings, M., Effectiveness of end-user debugging software features: Are there gender issues? *ACM Conf. Human Factors Comp. Sys.,* 2005, 869-878.

[9] Blackwell, A., First steps in programming: A rationale for attention investment models, *Proc. IEEE Symp. Human-Centric Comp. Langs. Envs.,* 2002, 2-10.

[10] Burnett, M., Cook, C., Rothermel, G., End-user software engineering, *Communications of the ACM*, 2004, 53-58.

[11] Carroll, J., Rosson, M., Paradox of the active user*, In Interfacing Thought: Cognitive Aspects of Human-Computer Interaction,* J. Carroll (Ed.), MIT Press, 1987, 80-111.

[12] Conway, M., et al., Alice: Lessons learned from building a 3D system for novices, *ACM Conf. Human Factors Comp. Sys.,* 2000, 486-493.

[13] Farkas, D., Layering as a safety net for minimalist documentation, In Carroll, J. M., (Eds.), *Minimalism Beyond the Nurnberg Funnel,* MIT Press, 1998, 247-274.

[14] Forrest-Pressly, D., MacKinnon, G., Waller, T., *Metacognition, Cognition, and Human Performance,* Academic Press, 1985.

[15] Gee, J., *An Introduction to Discourse Analysis,* Routledge, London, 1999.

[16] Green, T., Cognitive dimensions of notations, In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V*. Cambridge University Press, 1989. pp 443-460.

[17] Gordon, S., Gill, R., Knowledge acquisition with question probes and conceptual graph structures, In T. Lauer, E. Peacock, A. Graesser (Eds.), *Questions and Information Sys.,* Lawrence Erlbaum Assc., 1992, 29-46.

[18] Harrison, S., A Comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface, *ACM Conf. Human Factors Comp. Sys.,* 1995, 82-89.

[19] Kahler, H., Kensing, F., Muller, M., Methods & tools: constructive interaction and collaborative work: introducing a method for testing collaborative systems, *Interactions*, 7, 3, June 2000, 27-34.

[20] Katalin, E., "Please keep talking": The 'think-aloud' method in second language reading research, *Novelty*, 7, 3, 2000.

[21] Kissinger, C., Burnett, M., Stumpf, S., Subrahmaniyan, N., Beckwith, L., Yang, S., Rosson, M., Supporting end-user debugging: What do users want to know? *ACM Conf. Advanced Visual Interfaces,* 2006, (to appear).

[22] Ko, A., Myers, B., Designing the Whyline: A debugging interface for asking questions about program failures, *ACM Conf. Human Factors Comp. Sys.,* 2004, 151-158.

[23] Ko, A., Myers, B., Aung, H., Six learning barriers in end-user programming systems, *IEEE Symp. Vis. Lang. Human-Centric Comp.,* 2004, 199-206.

[24] Lowenstein, G., The psychology of curiosity, *Psychological Bulletin,* 116, 1, 1994, 75-98.

[25] Miller, R., Myers B., Outlier finding: Focusing user attention on possible errors, *ACM User Interface Soft. Tech.,* 2001, 81-90.

[26] Nardi, B., *A Small Matter of Programming: Perspectives on End User Computing,* MIT Press, 1993.

[27] Norman, D., *The Design of Everyday Things,* New York, NY: Doubleday, 1988.

[28] Palmiter, S., Elkerton, J., An evaluation of animated demonstrations for learning computer-based tasks, *ACM Conf. Human Factors Comp. Sys.,* 1991, 257-263.

[29] Panko, R., What we know about spreadsheet errors, *Journal of End User Computing,* 1998, 15-21.

[30] Person, N., Graesser, A., Fourteen facts about human tutoring: Food for thought for ITS developers, *AIED Workshop on Tutorial Dialogue,* 2003, 335-344.

[31] Phalgune, A., Kissinger, C., Burnett, M., Cook, C., Beckwith, L., Ruthruff, J. R., Garbage in, garbage out? An empirical look at oracle mistakes by end-user programmers, *IEEE Symp. Visual Langs. Human-Centric Comp.*, 2005, 45-52.

[32] Plaisant C., Shneiderman, B., Show me! Guidelines for producing recorded demonstrations, *IEEE Symp. Visual Langs. Human-Centric Comp.*, 2005, 171-178.

[33] Prabhakararao, S., Cook, C., Ruthruff, J. R., Creswick, E., Main, M., Durham, M., Burnett, M., Strategies and behaviors of end-user programmers with interactive fault localization, *IEEE Symp. Visual Langs. Human-Centric Comp.*, 2003 15-22.

[34] Rieber, L. P., Animation in computer-based instruction, *Educational Technology Research and Development*, 38 (1), 1990, 77-86.

[35] Robertson, T., Lawrance, J., Burnett, M., Impact of high-intensity negotiated-style interruptions on end-user debugging, *J. Visual Langs. Comp.,* to appear 2006.

[36] Ruthruff, J. R., Phalgune, A., Beckwith, L., Burnett, M., Cook, C., Rewarding 'good' behavior: End-user debugging and rewards, *IEEE Symp. Visual Langs. Human-Centric Comp.,* 2004, 107-114.

[37] Wagner, E., Lieberman, H., Supporting user hypotheses in problem diagnosis on the web and elsewhere, *ACM Int. Conf. Intelligent User Interfaces,* 2004, 30-37.

[38] Weinert, F., Kluwe, R., (Eds.) *Metacognition, Motivation, and Understanding,* Lawrence Erlbaum Associates. 1987.

[39] Wetzel, C., Radtke, P., Stern, H., *Instructional Effectiveness of Video Media*, Lawrence Erlbaum Associates. 1994.

[40] Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., Durham, M., Rothermel, G., Harnessing curiosity to increase correctness in end-user programming, *ACM Conf. Human Factors in Comp. Sys.,* 2003, 305–312.

Appendices

Appendix A: Sample of Transcribed and Coded Dialogue

Stanza	Text	Cory Code	Neeraja Code	Aggre-ment
1	e. whats the difference between like the two checkmarks?	bmean	bmean	1
2	f. or the two xs? one of them, the darker one means its right, and then the lighter one means seems right maybe e. oh, ok () [rest of tutorial] f. you know how we can click on the red boxes and show the arrows where they go, if we click on one of these boxes [input cells] will it show, is there any way to get an arrow to go to that one [refering to an arrow from quiz1 to min_q1_q2] no f. oh, ok () [rest of description]	hq	bmean,hq	0.5
3	f. ok, so lets start by quiz1, just we got this, enter 81.25 e. 81.25, apply f. ok, quiz average e. wheres the quiz average? oh right there f. it's right down there e. is 16.2 f. it doesn't- e. we have to put in quiz2 too though, we have to put in all the quizes f. oh, ok, ya go ahead e. so whats quiz2? f. 100 e. ah, smart e. whats quiz3? f. 100 f. quiz4 would be 96, and quiz5 would be 100	sh, vf	vf	0.5
4	e. ok, so thats the average now, that doesn't make any sense f. ya, thats ninety- e. ok, so ya that has to be wrong f. scroll that down and see e. quiz1 minus minimum f. min quiz1 quiz2? e. ok, to find the average you have to divide it, so f. wait, wait, hold on don't change anything yet, e. oh, thats divided by f. do not change anything yet. e. I'm not f. miny quiz1, minus quiz1 and quiz2. I don't get what their asking for? (.)	vf,sqh	vf,sqh	1
5	e. its an if statement, so if quiz-	vf	vf	1

	f. one is less than [both] quiz2 then quiz1 e. oh that must mean they get to drop the lowest score between those two f. one of the two, ya			
6	f. I guess we should read the front of this [reading] f. ok, [reading aloud] the midterm is the average of the third midterm and the higher of the first two f. the first midterm is out of 50 points the second midterms are worth 100 points, students receiving a non-zero grade on the third midterm recieve a two point bonus e. so two extra credit points f. the final exam is out of 146, blah blah blah f. there are 5 quizes, the average is calculated on only four of these scores, dropping the lower of the first two quizes e. ok f. ok, so	sh,vf	vf	0.5
7	f. minus quiz1 or ok minus that [refering to min_q1_q2] f. would we divide by 4? e. ya e. well, is this right though [refering to min_q1_q2], this part up here f. if quiz1, I think so, thats- e. if quiz1 is < quiz 2, then quiz2 that wouldn't be quiz1 f. if quiz1- e. oh wait f. is less than quiz2, then quiz1, huh? e. ya i think it needs to be switched f. ok, well lets write down whatever switches we make e. ok f. so- oh no just keep that, just switch the >= to the <= e. ok f. or the less than e. if quiz1 f. you didn't change anything f. ok e. if quiz1 is greater than qui2 then quiz1 else quiz2, ya see f. ok, that makes sense e. see there we go, see	vf, sh	sh,vf	1
8	f. ok, then I'm still saying change this to 4 [refering back to quiz_avg] e. ok f. cause that just makes way more sense e. ya f. and that would be- e. ya, that looks way better f. that would be the right number	vf	vf	1

	e. well, no why- f. oh, wait why would it be e. ya, it wouldn't be 94 cause it- f. cause we have 100, 100, 96, 100, um go back up here [min_q1_q2]			
9	f. see its minusing 100 i think, ya, ya it was right the first time, roll that back the other way [> back to <] e. are you sure? f. yep watch e. ok f. ok, 81.25, this minus this [quiz_avg minus min_q1_q2 ?] divided by four, look 99 e. ok, ya f. see, I roll it	vf,sh,l	l,sh,vf	1
	f. ok so we got it, so we can check this one right e. maybe as we go along we should cross off the words- f. well we got these little checkmarks, ok check that one as definitely being right [min_q1_q2] e. ok check			
10	e. and then this ones right? [refering to quiz avg] f. that one, should be right, ya e. ok	vf	vf	1
11	f. ok 21% tested, ok, now we can enter these ones in Midterm1 2 and 3, um roll with 100 or wait 45 e. 45?? f. um hm, e. ok, midterm2? f. 96, then midterm3 is 80 e. midterm3 is 80? f. ok hide that [refering to midterm3]		sh,vf	0
12	f. now midterm3, ok click on this for a sec, oh no, curved midterm, oh ok ya, they get the extra 2 bonus points, click that down so we can look at it real quick [refering to curved_m3] [reading formula] f. ok ya thats fine, hide e. ok	vf	vf	1
13	e. so then the final f. check that one cause its right [refering back to curved_m3] e. ok f. and then, the final he got 129, final 129 f. ok now wait, lets check this box, is that ok? [refering to final_per] e. yep f. ok, cause lets see, the final is out of 146, click that down, [reading formula] e. so thats correct? f. ya, we can click e. click it	vf, sh	sh,vf	1
14	f. oo, we're 26 percent tested			

	[laughter]			
15	e. midterm1 percentage, 2 * midterm1 f. is it out of 50? that would make sense, if its out of 50 e. so this is correct? f. 2 * whatever, ya that should be right f. that would make sense wouldn't it? if he got 45, he got a 90 overall, ok ya, thats very clever	vf	vf	1
16	[e opens fmla for min_m1_m2] f. ok, wait I didn't want to look at that one yet f. why do we have all these arrows up here? e. cause I lit them up, well it tells you where to go f. i know, but i'm getting lost with the two- f. ok, I guess lets look at midterm average first [reading fmla for midterm_avg] f. wow, thats perfect except we need to figure out first if thats 90, we need to make sure that 90 % is the lowest, I guess is what I'm tring to say	whoa,sh, eright,vf, bmean	bmean,er ight,vf,sh, whoa	1
17	[reading fmla for min_m1_m2] f. ok, wheres midterm2? e. right there f. ok, so thats 96 out of 100, so the 90s whats it, out of 100? f. ok, ya so thats right, its dropping that one f. so we actually got 96 e. so you can drop either midterm? is that what its saying? f. the higher of the first two midterms, so you can drop either one of these number 1 and number 2[refering to m1 m2], number 3 you get 2 bonus points for taking it, but thats it e. ok, so we wanna drop the 90 f. and thats whats in that box [refering to min_m1_m2], so thats- hit hide e. so is this correct? f. actually lets just see if we got the right grades now	vf	sqh,vf	0.5
18	[turns over ss desc.], nope f. course avg should be a 92, so we've got issues e. midterm avg, doesn't say (.) f. well our midterm avg isn't going to be 143, is it? e. no f. so lets figure that out [reading fmla for midterm_avg]	sh, vf	vf,sh	1
19	f. oh, we should look at Cory conversation, maybe we should open that just in case e. he didn't say anything f. i was gonna say, I have no idea how that works e. no it will flash			
20	[reading fmla for midterm_avg] e. so don't you think we should change this [mouse moves over Curved_m3]	vf, sh	sh,vf	1

f. well what I'm saying is curved midterm3 doesn't make any sense
f. it would have to be plus midterm3
e. hold on, lets look at this guy [opens fmla for curved_m3]
f. ya we got to add plus midterm3
f. so just add-
[change midterm_avg fmla to have midterm3 in it]
f. ok, ya hit that, try that one
f. still high, ok midterm1 percentage
e. ok well why would you include that [midterm1 perc] if your not going to include that because we only include the higher the higher of the two
f. because its subtracting 90 right there, see what its saying, we gotta account for everything, cause if we enter it in whats going to happen there
f. midterm1 percentage plus midterm 2 plus curved midterm3 plus midterm 3
e. which is right there
f. ya minus 90 divided by 2
f. so the question is why are we getting-
e. should it be divided by 3?
f. no, cause we only have 2
e. oh ya
f. midterm 1 + midterm2 + midterm3 minus 90, thats a problem [only fmla open is midterm avg]
f. oh no midterm1, so 90 + 96 + 86, so why do we get such a high number?
f. cause the average between those two isn't even close to that
e. man if we had a calculator, I wonder if we could go on here and look at the calculator, do you think that that average is right?
f. I don't know, Cory can you look at the calculator? (.)
[typing into im]
f. he said 5 seconds, I think he can hear (.)

The entire dataset can be found at:

http://web.engr.oregonstate.edu/~burnett/Forms3/RecordedDemos/all-coded-dialogue.xls

Appendix B: Transcription Conventions

<u>emphasis</u>	spoken emphasis on word
(.)	audible pause in speaking
(ia)	inaudible speech
()	irrelevant speech (off topic, reciting of spreadsheet description, reading off values)
[comments]	comments and actions
Line of text	group of words said as if they "go together"
Blank line	separates "stanzas" (clumps of statements on the same topic)
X.	letter of the alphabet representing the subject who was speaking
Italic	anything spoken or typed by the researcher
-	sentence or thought ends abruptly

Appendix C: Tutorial Materials

Collaboration Tutorial

Hi, My name is Cory， I will be leading you through today's study.

The other people involved in this study are Dr. Margaret Burnett, Dr. Curtis Cook, and the assistants helping me out today.

Just so you know, I'll be reading through this script so that I am consistent in the information I provide you and the other people taking part in this study, for scientific purposes.

The aim of our research is to help people create correct spreadsheets Past studies indicate that spreadsheets contain several errors like incorrectly entered input values and formulas. Our research is aimed at helping users find and correct these errors.

For today's experiment, I'll lead you through a brief tutorial of Forms/3, and then you will have an experimental tasks to work on.

But first, I am required by Oregon State University to read aloud the text of the "Informed Consent Form" that you currently have in front of you:

- (Read form).

Please do NOT discuss this study with anyone. We are doing later sessions and would prefer the students coming in not to have any advance knowledge.

Questions?

Contact:

- Dr. Margaret Burnett burnett@cs.orst.edu
- Dr. Curtis Cook cook@cs.orst.edu

Any other questions may be directed to IRB Coordinator, Sponsored Programs Office, OSU Research Office, (541) 737-8008

Think aloud practice:

In this experiment we are interested in what you say as you perform some tasks that we give you. In order to do this we will ask you to TALK TO EACHOTHER CONTINUALLY as you work on the problems. If any of you is silent for any length of time, the assistant will remind you to keep talking to each other. It is most important that both of you keep talking. Do you understand what I want you to do?

Good. For practice, I want you to talk to each other while you answer the following question:

How did you find your way to this room while coming here with your partner?

How to Work in Pairs

You will work in pairs as you do the spreadsheet tasks.

The two of you will work on the same computer with one mouse and one keyboard. One of you will be the "driver" who controls the mouse and keyboard, doing all the input.

The other will be the “reviewer.” The reviewer is *not* a passive observer. The reviewer analyzes the current situation, discusses the situation with the driver, makes suggestions and comments, looks for errors, and thinks about the next steps: There should be lots of discussion and interaction between the driver and reviewer.

The driver and reviewer change every 10 minutes. You will be notified when it is time to switch by Instant Messenger, which I will explain further shortly.

Experiment

In this experiment you will be working with the spreadsheet language Forms/3. To help you become familiar with the features of Forms/3, we're going to start with a short tutorial in which we'll work through a sample spreadsheet problem. After the tutorial you will be given a spreadsheet and asked to test it, correcting any errors you find.

As we go through this tutorial, I want you to actually PERFORM the steps I'm describing. When I say, "click", I'll always mean click the left mouse button once unless I specify otherwise. Pay attention to your computer screen while you do the steps.

If you have any questions, please don't hesitate to ask me to explain.

For each spreadsheet that we will be working, you will have a sheet of paper describing what the spreadsheet is supposed to do
Hand out purchase budget description, wait for them to read it

Now open the Purchase Budget spreadsheet by clicking on the bar labeled Purchase Budget at the bottom of the screen

This is a Forms/3 spreadsheet. There are a few ways that Forms/3 spreadsheets look different than the spreadsheets you may be familiar with:

- Forms/3 spreadsheets don't have cells in a grid layout. We can put cells anywhere. However, just like with any other spreadsheet you can see a value associated with each cell.
- We can give the cells useful names like PenTotalCost. (Point to the cell on the Spreadsheet).
- Some of the cells have colored borders.

Let’s start by looking at a few formulas. To open the formula for the Pens cell, click on the arrow on the lower right hand side of the cell. This cell is just a value. Try changing the value to 25 and click the apply button.

Now open the formula for the PenQCheck cell. You may notice that this cell is not just a value. Its formula says if the sum of the Pens and PensOnHand is greater than 68, then the cell should contain “not enough pens”, and otherwise it should contain “pen quantity ok”. Fortunately, for the purposes of this tutorial this formula is incorrect. Lets try changing this formula so that it correctly prints “pen quantity ok” if Pens + PensOnHand is greater than 68 and “not enough pens” otherwise. There are two ways to make this change, either switch the text that comes after the else part with the text that comes after the then part OR change the > symbol to a <= symbol. Then click the apply when your done.

Forms/3 has several features that respond visually to your actions to help you test and find errors in your spreadsheet.

You might be wondering what does testing have to do with spreadsheets? Well, as you have just seen it is possible for errors to exist in spreadsheets, but what usually happens is that they tend to go unnoticed. It is in our best interest to find and weed out the errors in our spreadsheets so that we can be confident that they work correctly in all situations.

One of the features that you may have noticed is the small box with a question mark in it in the upper right hand corner of the cell. This decision box is how we communicate to the system which values we think are correct and which we think are wrong.

Go ahead and click the decision box in the upper right corner of the PenTotalCost cell. Four choices appear – 2 X marks and 2 check marks. Past research has shown that people who use these features are more effective at testing and finding errors in their spreadsheets. Placing an X on this cell means the value is wrong, placing a check means the value is right.

Use the sample correct values in spreadsheet description to help you decide which mark to place. Now try placing one of these marks by clicking on it. One of the visual responses to this action occurs in the progress bars at the top of the page. To undo this decision click on the mark again. Notice that everything went back to how it was. I'll give you a minute to try each of these marks out. Wait one minute

Another feature is arrows. Position your mouse to the middle of the PaperQCheck cell and click the middle mouse button (the scroll wheel). Notice that arrows appear. Click the middle mouse button again on any one of these arrows—it disappears. (PAUSE) Now, click the middle mouse button again on PaperQCheck cell—all the other arrows disappear.

Make this whole paragraph first person
Any question should be directed to me. I will be out of the room, but there are two ways to ask me questions: you can either say "Cory" and then your question into the microphone or you can type your question into the instant messenger. Let's try asking me a question using the microphone, will the driver say into the microphone "Cory, what does the X mark mean". I will respond as soon as I get back to my computer. All of my responses will show up in the instant messenger window that is labeled Cory at the bottom of your screen. However, there is a short delay any responses will take 10 to 15 seconds. I will also instant message you when it is time to switch places.

You also get three free hints to be used to help you test and find errors. (hand them tokens) Simply wave one of these in front of the screen and I'll come back in the room to give you a hint.

have them minimize the Purchase Budget spreadsheet

Spreadsheet task
Gradebook.frm hand out description

Here is a gradebook spreadsheet problem. Let's read the second paragraph at the top of the description:

"Your task is to test the updated spreadsheet to see if it works correctly and to correct any errors you find."

The front side of this description describes how the spreadsheet should work.
Also, if you turn to the backside of this sheet (*turn over your description*), you'll see that two correct sample report cards are provided to you. You can use these to help you in your task.

While you are working on the spreadsheet, remember to keep talking to each other. Please talk together about any questions you have about Forms/3, about the task or errors, any thoughts you have about how to do the task, or any reasons why you are taking particular actions to complete the task. If you are silent for a while, the assistant will remind you to keep talking to each other.

Remember, your task is to test the spreadsheet, and correct any errors you find. If you have any questions, talk about it with your partner or ask Cory. You may find learning those checkmarks and X-marks useful.

Start your task now, and I'll tell you when time is up.

Have them switch places

Payroll.frm hand out description

Here is a Payroll spreadsheet problem. Let's read the second paragraph at the top of the description:

"Your task is to test the updated spreadsheet to see if it works correctly and to correct any errors you find."

The front side of this description describes how the spreadsheet should work.
Also, if you turn to the backside of this sheet (*turn over your description*), you'll see that two correct sample payroll stubs are provided to you. You can use these to help you in your task.
While you are working on the spreadsheet, remember to keep talking to each other. Please talk together about any questions you have about Forms/3, about the task or errors, any thoughts you have about how to do the task, or any reasons why you are taking particular actions to complete the task. If you are silent for a while, the assistant will remind you to keep talking to each other.

Remember, your task is to test the spreadsheet, and correct any errors you find. If you have any questions, talk about it with your partner or ask Cory. You may find learning those checkmarks and X-marks useful.

Start your task now, and I'll tell you when time is up.

Appendix D: Questionnaires

Number: _____

Background Questionnaire

2. Gender (circle your selection): Male / Female

3. Age < 20 20 – 29 30 – 39 40 – 49 50 – 59 60+

4. Major or Educational Background: _____

5. Year or Degree Completed: Fresh. Soph. Jun. Sen. Post Bac. Grad.

6. Cumulative GPA: _____

7. Do you have previous programming experience?

   a. High school:

      - How many courses? _____
      - What programming languages? _____

   b. College:

      - How many courses? _____
      - What programming languages? _____

   c. Professional and/or recreational

      - How many years? _____
      - What programming languages? _____

8. Have you ever created a spreadsheet for (please check all that apply):

   - ❑ A high school course How many? _____
   - ❑ A college course How many? _____
   - ❑ Professional use How many years? _____
   - ❑ Personal use How many years? _____

9. Have you participated in any previous Forms/3 experiments? Yes / No

10. Is English your primary language? Yes / No

    If not, how long have you been speaking English? _____ years.

Pre-session Questionnaire

The following questions ask you to indicate whether you could use a new spreadsheet system under a variety of conditions. For each of the conditions please indicate whether you think you would be able to complete the job using the system.

Given a spreadsheet which performs common tasks (such as calculating course grades or payroll) I could find and fix errors:

... if there was no one around to tell me what to do as I go.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had never used a spreadsheet like it before.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had only the software manuals for references.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had seen someone else using it before trying it myself.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I could call someone for help if I got stuck.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if someone else had helped me get started.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had a lot of time to complete the task.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had just the built-in help facility for assistance.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if someone showed me how to do it first.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had used similar spreadsheets before this one to do this same task.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

What is your relationship to the person doing the study with you today (i.e. close friend, friend, classmate, significant other, etc)?

Number: _____

Post-session Questionnaire (Gradebook)

I. Circle the answer corresponding to how much you agree or disagree with the following statements.

1. I am confident that I found all the bugs in the Gradebook spreadsheet? (circle one)

Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

2. I am confident that I fixed all the bugs in the Gradebook spreadsheet? (circle one)

Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

II. How much additional time would you need to complete this task?

_____ None. It only took me _____ minutes.

_____ None. I took about the entire time.

_____ I would need about _____ more minutes.

_____ I am not sure.

Number: _____

Post-session Questionnaire (Payroll)

Circle the answer corresponding to how much you agree or disagree with the following statements.

1. I am confident that I found all the bugs in the Payroll spreadsheet? (circle one)

Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

2. I am confident that I fixed all the bugs in the Payroll spreadsheet? (circle one)

Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

3. How much additional time would you need to complete this task?

_____ None. It only took me _____ minutes.

_____ None. I took about the entire time.

_____ I would need about _____ more minutes.

_____ I am not sure.

4. If there are still errors in the spreadsheet this is because… (Circle **1** reason you agree with most)

a. The computer should have helped me spot the errors

b. I should have spent more time trying to find the errors

c. There was not enough time

d. None of the above

Post-session Questionnaire

The following questions ask you to indicate whether you could use a new spreadsheet system under a variety of conditions. For each of the conditions please indicate whether you think you would be able to complete the job using the system.

Given a spreadsheet which performs common tasks (such as calculating course grades or payroll) I could find and fix errors:

... if there was no one around to tell me what to do as I go.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had never used a spreadsheet like it before.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had only the software manuals for references.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had seen someone else using it before trying it myself.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I could call someone for help if I got stuck.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if someone else had helped me get started.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had a lot of time to complete the task.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had just the built-in help facility for assistance.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if someone showed me how to do it first.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
... if I had used similar spreadsheets before this one to do this same task.	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

5. Mark how you found the following features for **finding and fixing errors**:

Cell border colors helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Interior Cell Coloring (yellow and red) helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
X-marks helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Checkmarks (√) helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Arrows helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Instant messenger responses helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Three free hints helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
My partner helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Percent tested indicator helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree
Bug likelihood bar helped me make progress	Strongly Disagree	Disagree	Neither Agree Nor Disagree	Agree	Strongly Agree

5a. Rank your preference for the following features from **1 – most preferred feature to 9 – least preferred feature**:

_____ Cell border colors

_____ Interior cell colorings

_____ X-marks

_____ Checkmarks

_____ Arrows

_____ Instant messenger responses

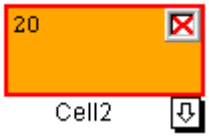
_____ Three free hints

_____ Percent testedness indicator

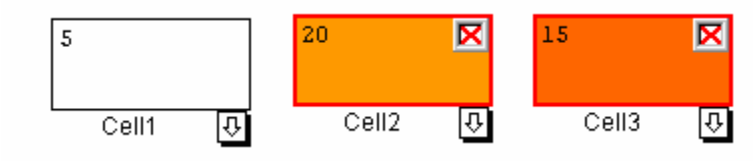
_____ Bug likelihood bar

6. What does the X- mark in the decision box mean?

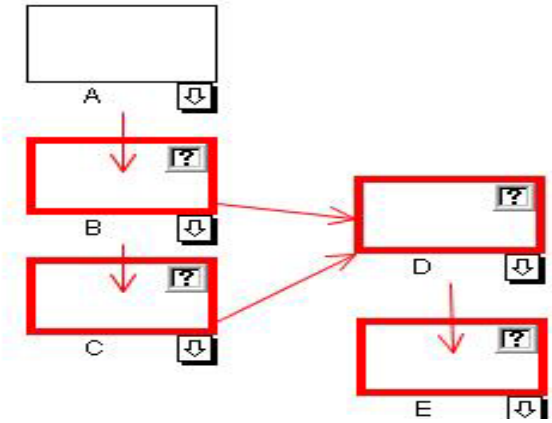
7. In the figure below what does the orange color in the interior of the cell mean?



8. In the figure below what does it mean when the color in the interior of one cell is darker than others?



Please provide any other general comments you may have regarding the cell interior colorings:



Q9 to Q14: Refer to the Figure Above and choose your answers from the choices below.
One or more Questions can have the same answer.

9. If we place an X- mark in cell D the color of the cell D:
   a. Remains the same
   b. Gets darker
   c. Gets lighter
   d. Don't know

10. If we place an X- mark in cell D the color of the cell C
   a. Remains the same
   b. Gets darker
   c. Gets lighter
   d. Don't know

11. If we place an X- mark in cell D the color of the cell E
    a. Remains the same
    b. Gets darker
    c. Gets lighter
    d. Don't know

Assume for the next three Questions (12-14) that an X- mark has been placed on the cell D.

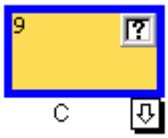
12. If we place an X- mark in cell C the color of the cell C
    a. Remains the same
    b. Gets darker
    c. Gets lighter
    d. Don't know

13. If we place an X- mark in cell C the color of the cell B
    a. Remains the same
    b. Gets darker
    c. Gets lighter
    d. Don't know

14. If we place a Checkmark in cell C the color of the cell D
    a. Remains the same
    b. Gets darker
    c. Gets lighter
    d. Don't know

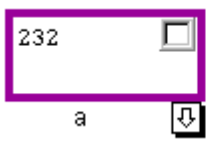
15. What does a blue border of a cell with a yellow-orange interior mean (refer to figure below)?

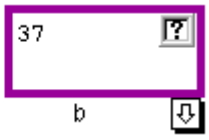
(Circle 1 option for each part)



a) The value is: (circle 1)	CORRECT	WRONG	COULD BE EITHER
b) The cell is: (circle 1)	TESTED	UNTESTED	COULD BE EITHER
c) The cell has: (circle 1)	BUG LIKELIHOOD	NO BUG LIKELIHOOD	COULD BE EITHER
d) My answers to a, b, and c are just guesses.	YES, JUST GUESSES	NO, NOT GUESSES	
e) The combination of blue border and yellow-orange interior colors on this cell: (circle 1)	MAKES SENSE	MAKES NO SENSE	NOT SURE

16. A spreadsheet has two cells that look as follows.





Suppose you have determined that their values are correct. To increase percent testedness, you would check:

_____cell a only. Why?_____
_____cell b only. Why?_____
_____cells a and b. Why?_____
_____neither. Why?_____
_____I'm not sure.

17. There is a purple cell with a **blank** in its decision box.

A) If you place a check mark in that decision box, does the border color change?
_____ yes
_____ no
_____ I'm not sure

B) What is the border color after you've placed the checkmark in the decision box?
_____red
_____same purple
_____"bluer" purple
_____blue
_____I'm not sure

C) The form's Percent Tested indicator will:
_____Increase.
_____Stay the same.
_____Decrease.
_____Not enough information to tell.
_____I'm not sure.

18. There is a purple cell with a **question mark** in its decision box.

A) If you place a check mark in that decision box, does the border color change?
_____ yes
_____ no
_____ I'm not sure

B) What is the border color after you've placed the checkmark in the decision box?
_____red
_____same purple
_____"bluer" purple
_____blue
_____I'm not sure

C) The form's Percent Tested indicator will:
_____Increase.
_____Stay the same.
_____Decrease.
_____Not enough information to tell.
_____I'm not sure.

Directions.
Score yourself. Circle the number that best ranks your behavior for the given question. A rank value of 1 means "Not at all true for me" while a rank of 7 means "very true for me". Be as precise as possible.

Remember: 1 = not true at all ... 7 = always true

1. During class time, I often miss important points because I'm thinking of other things.
1 2 3 4 5 6 7

2. When reading for a course, I make up questions to help focus my reading.
1 2 3 4 5 6 7

3. When I become confused about something I'm reading, I go back and try to figure it out.
1 2 3 4 5 6 7

4. If course materials are difficult to understand, I change the way I read the material.
1 2 3 4 5 6 7

5. Before I study new material thoroughly, I often skim it to see how it is organized.
1 2 3 4 5 6 7

6. I ask myself questions to make sure I understand the material I have been studying.
1 2 3 4 5 6 7

7. I try to change the way I study in order to fit the course requirements and the instructor's teaching style.
1 2 3 4 5 6 7

8. I often find that I have been reading for a class but don't know what it was all about.
1 2 3 4 5 6 7

9. I try to think through a topic and decide what I am supposed to learn from it rather than just reading it over when studying.
1 2 3 4 5 6 7

10. When studying, I try to determine which concepts I do not understand well.

1 2 3 4 5 6 7

11. When I study, I set goals for myself in order to direct my activities in each study period.

1 2 3 4 5 6 7

12. If I get confused taking notes, I make sure I sort it out afterwards.

1 2 3 4 5 6 7

Appendix E: Spreadsheets and Spreadsheet Descriptions

Purchase Budget

You are in charge of ordering office supplies for the office you work at. You must order enough pens and paper to have on hand.

You must keep more than 68 boxes of pens and 400 reams of paper on hand.

Pen and Paper
The quantity of pens and paper that you are ordering and the quantity you have on hand.

Costs of Pen and Paper
The cost of pens is $2 per box, and the cost of paper is twice that, $4.

Pen and Paper Check
These cells are used to check to ensure you are ordering enough pens and paper to restock the shelves.

Example data for correct spreadsheet

Pens	25
Paper	400
PensOnHand	25
PaperOnHand	21
PenTotalCost	50
PaperTotalCost	1600
PenQCheck	not enough pens
PaperQCheck	paper quantity ok

Task: Test the spreadsheet to see if it works correctly and correct any errors you find.

GRADEBOOK SPREADSHEET PROBLEM

Another teacher has updated a spreadsheet program that computes the course grade of a student. Two correct sample report cards and information about the class' grading policy are provided.

Your task is to test the updated spreadsheet to see if it works correctly and to correct any errors you find.

Quizzes and Exams

The exam average is the average of the midterm average and the final exam.

The midterm average is the average of the third midterm and the higher of the first two midterms. The first midterm is out of 50 possible points. The second and third midterms are worth 100 points. Students achieving a non-zero grade on the third midterm receive a two point bonus. The final exam is out of 146 possible points. Exams not based on 100 points have their percents computed for later averaging.

There are five quizzes. The average is calculated on only four of these scores, dropping the lower of the first two quizzes.

Course Grade

Quizzes are worth 40% of a student's grade. Midterms are worth 40% of a student's grade. The final contributes 20%. A student's course grade is determined by their course average, in accordance with the following scale:

90 and up : A	70 – 79 : C
80 - 89 : B	60 - 69 : D
	Below 60 : F

Example Correct Gradebook Report Cards

John Doe	**Report Card**
Quiz1	81.25
Quiz2	100
Quiz3	100
Quiz4	96
Quiz5	100
Midterm1 (Original)	45
Midterm2	96
Midterm3 (Original)	80
Final	129
Course_Avg	92.87
Course_Grade	A

Mary Smith	**Report Card**
Quiz1	0
Quiz2	88.24
Quiz3	85
Quiz4	87
Quiz5	100
Midterm1 (Original)	24
Midterm2	61
Midterm3 (Original)	66
Final	106
Final_Percentage	
Course_Avg	76.34
Course_Grade	C

List of bugs in the Gradebook spreadsheet

The Gradebook spreadsheet was seeded with five faults created by real end users.

Table 6: List of bugs in Gradebook spreadsheet: Output cells with their formulas when the spreadsheet is first loaded. (Note: All the other input cells have a value 0)

Cell Name	Original Formula	Correct Formula
Curved_Midterm3	if Midterm3 > 0 then 2 else 0	if Midterm3 > 0 then Midterm3+2 else 0
Quiz_Avg	((Quiz1 + Quiz2 + Quiz3 + Quiz4 + Quiz5) - Min_Quiz1_Quiz2) / 5	((Quiz1 + Quiz2 + Quiz3 + Quiz4 + Quiz5) - Min_Quiz1_Quiz2) / 4
Midterm_Avg	Midterm1_Perc + Midterm2 + Curved_Midterm3 - Min_Midterm1_Midterm2 / 2	(Midterm1_Perc + Midterm2 + Curved_Midterm3 - Min_Midterm1_Midterm 2) / 2
Exam_Avg	(Midterm_Avg + Final_Percentage) / 3	(Midterm_Avg + Final_Percentage) / 2
Course_Avg	(Quiz_Avg * 0.4) + (Midterm_Avg * 0.4) + (Final_Percentage * 0.2) / 10	(Quiz_Avg * 0.4) + (Midterm_Avg * 0.4) + (Final_Percentage * 0.2)

Table 7: Formula's of output cells in the Gradebook spreadsheet (Note: All the other input cells have a value 0)

Cell Name	Original Formula
Min_Quiz1_Quiz2	if (Quiz1 < Quiz2) then Quiz1 else Quiz2
Midterm1_Perc	2 * Midterm1
Min_Midterm1_Midterm2	if (Midterm1_Perc < Midterm2) then Midterm1_Perc else Midterm2
Curved_Midterm3	if Midterm3 > 0 then 2 else 0
Final_Percentage	Final / 146 * 100
Quiz_Avg	((Quiz1 + Quiz2 + Quiz3 + Quiz4 + Quiz5) - Min_Quiz1_Quiz2) / 5
Midterm_Avg	Midterm1_Perc + Midterm2 + Curved_Midterm3 - Min_Midterm1_Midterm2 / 2
Exam_Avg	(Midterm_Avg + Final_Percentage) / 3
Course_Avg	(Quiz_Avg * 0.4) + (Midterm_Avg * 0.4) + (Final_Percentage * 0.2) / 10
Course_Grade	if Course_Avg >= 90 then "A" else (if Course_Avg >= 80 then "B" else (if Course_Avg >= 70 then "C" else (if Course_Avg >= 60 then "D" else "F")))

PAYROLL SPREADSHEET PROBLEM

- A spreadsheet program that computes the net pay of an employee has been updated by one of your co-workers.
- Below is a description about how to compute the answers.
- On the backside of this sheet are two correct examples, which you can compare with the values on screen.

Your task is to test the updated spreadsheet to see if it works correctly and to correct any errors you find.

FEDERAL INCOME TAX WITHHOLDING

To determine the federal income tax withholding:

1. From the monthly adjusted gross pay subtract the allowance amount (number of allowances claimed multiplied by $250). Call this amount the adjusted wage.
2. Calculate the withholding tax on adjusted wage using the formulas below:
   d. If Single and adjusted wage is not greater than $119, the withholding tax is $0; otherwise the withholding amount is 10% of (adjusted wage – $119).
   e. If Married and adjusted wage is not greater than $248, the withholding tax is $0; otherwise the withholding amount is 10% of (adjusted wage – $248).

SOCIAL SECURITY AND MEDICARE

Social Security and Medicare is withheld at a combined rate of 7.65% of Gross Pay. The Social Security portion (6.20%) will be withheld on the first $87,000 of Gross Pay, but there is no cap on the 1.45% withheld for Medicare.

INSURANCE COSTS

The monthly health insurance premium is $480 for Married and $390 for Single. Monthly dental insurance premium is $39 for Married and $18 for Single. Life insurance premium rate is $5 per $10,000 of insurance. The monthly employer insurance contribution is $520 for Married and $300 for Single.

ADJUSTED GROSS PAY

Pretax deductions (such as child care and employee insurance expense above the employer's insurance contribution) are subtracted from Gross Pay to obtain Adjusted Gross Pay.

Example Correct Payroll Stubs

John Doe		**Month**	**Year-To-Date**
Marital Status – Single			
Allowances	1		
Gross Pay		6,000.00	54,000.00
Pre-Tax Child Care		0.00	
Life Insurance Policy Amount		10,000	
Health Insurance Premium	390.00		
Dental Insurance Premium	18.00		
Life Insurance Premium	5.00		
Employee Insurance Cost	413.00		
Employer Insurance Contribution	300.00		
Net Insurance Cost		113.00	
Adjusted Gross Pay		5,887.00	
Federal Income Tax Withheld	551.80		
Social Security Tax	372.00		
Medicare Tax	87.00		
Total Employee Taxes		1,010.80	
Net Pay		4,876.20	

Mary Smith		**Month**	**Year-To-Date**
Marital Status – Married			
Allowances	5		
Gross Pay		8,000.00	72,000.00
Pre-Tax Child Care		400.00	
Life Insurance Policy Amount		50,000	
Health Insurance Premium	480.00		
Dental Insurance Premium	39.00		
Life Insurance Premium	25.00 (25.0002 ok)		
Employee Insurance Cost	544.00		
Employer Insurance Contribution	520.00		
Net Insurance Cost		24.00	
Adjusted Gross Pay		7,576.00	
Federal Income Tax Withheld	607.80		
Social Security Tax	496.00		
Medicare Tax	116.00		
Total Employee Taxes		1,219.80	
Net Pay		6,356.20	

List of bugs in the Payroll spreadsheet

The Payroll spreadsheet was seeded with five faults created by real end users.

Table 8: List of bugs in Payroll spreadsheet: Output cells with their formulas when the spreadsheet is first loaded. (Note: All the other input cells have a value 0)

Cell Name	Original Formula	Correct Formula
SingleWithHold	if AdjustedWage < 119 then 0 else (AdjustedWage -248) *.10	if AdjustedWage < 119 then 0 else (AdjustedWage -119) *.10
MarriedWithHold	if GrossPay < 248 then 0 else (GrossPay – 248)*.10	if AdjustedWage < 248 then 0 else (AdjustedWage - 248)*.10
SocSec	if GrossOver87K = 0 then (GrossPay * 0.062 * 0.0145) else (87000 * GrossPay * 0.062 * 0.0145)	if GrossOver87K = 0 then (GrossPay * 0.062) else (87000 * 0.062)
SocSec	if GrossOver87K = 0 then (GrossPay * 0.062 * 0.0145) else (87000 * GrossPay * 0.062 * 0.0145)	if GrossOver87K = 0 then (GrossPay * 0.062) else (87000 * 0.062)
AdjustedGrossPay	GrossPay - PreTax_Child_Care – EmployeeInsurCost	GrossPay - PreTax_Child_Care - NetInsurCost

Table 9: Formula's of output cells in Payroll spreadsheet when it is first loaded. (Note: All the other input cells have a value 0)

Cell Name	Original Formula
FedWithHoldAllow	Allowances * 250
AdjustedWage	AdjustedGrossPay - FedWithHoldAllow
SingleWithHold	if AdjustedWage < 119 then 0 else (AdjustedWage -248) *.10
MarriedWithHold	if GrossPay < 248 then 0 else (GrossPay - 248)*.10
FedWithHold	if (MStatus = "Single") then SingleWithHold else MarriedWithHold
NewYTDGrossPay	YTDGrossPay + GrossPay
GrossOver87K	if NewYTDGrossPay > 87000 then NewYTDGrossPay - 87000 else 0
SocSec	if GrossOver87K = 0 then (GrossPay * 0.062 * 0.0145) else (87000 * GrossPay * 0.062 * 0.0145)
Medicare	GrossPay *.0145
LifeInsurPremium	LifeInsurAmount *.0005
HealthInsurPremium	if MStatus="Married" then 480 else 390
DentalInsurPremium	if MStatus = "Married" then 39 else 18
AdjustedGrossPay	GrossPay - PreTax_Child_Care - EmployeeInsurCost
EmployeeInsurCost	HealthInsurPremium + LifeInsurPremium + DentalInsurPremium
EmployerInsurContrib	if MStatus = "Married" then 520 else 300
NetInsurCost	if EmployeeInsurCost > EmployerInsurContrib then EmployeeInsurCost - EmployerInsurContrib else 0
EmployeeTaxes	SocSec + Medicare + FedWithHold
NetPay	AdjustedGrossPay - EmployeeTaxes
Mstatus (Input cell)	"Single"

Appendix F: Storyboards

All recorded demonstrations could be accessed via the same entry point (Figure 11). The remainder of this appendix presents a storyboard for each of the individual demonstrations. The screenshots were taken from the final version of the recorded demonstration and the embedded scripts follow what was actually said by each researcher in the demonstration. The topics covered along with which feature(s) they are linked to are as follows:

- Storyboard 1: Overall (default, plays if none other specified)
- Storyboard 2: Making Testing Progress (arrows, checkmark, testing progress bar)
- Storyboard 3: Finding Errors (X-mark, bug likelihood bar)
- Storyboard 4: Fixing Errors (link on web page)
- Storyboard 5: Changing Values (link on web page)
    a. To make testing progress (purple border)
    b. To make values easier to decide about (?)

(Note: each demonstration can also be reached via a link on the web page)

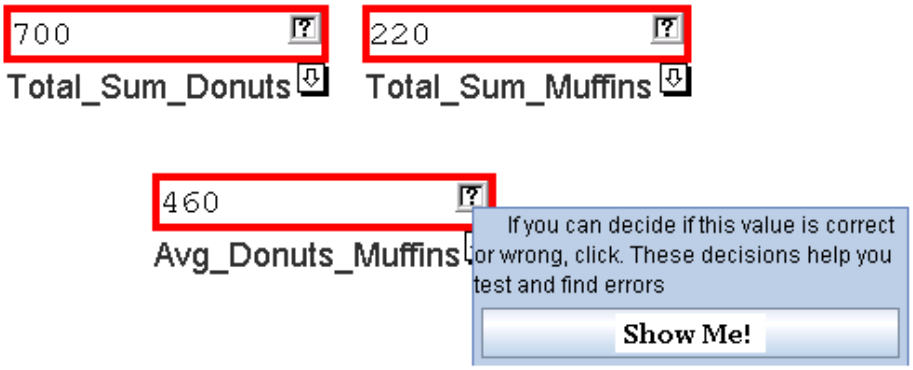
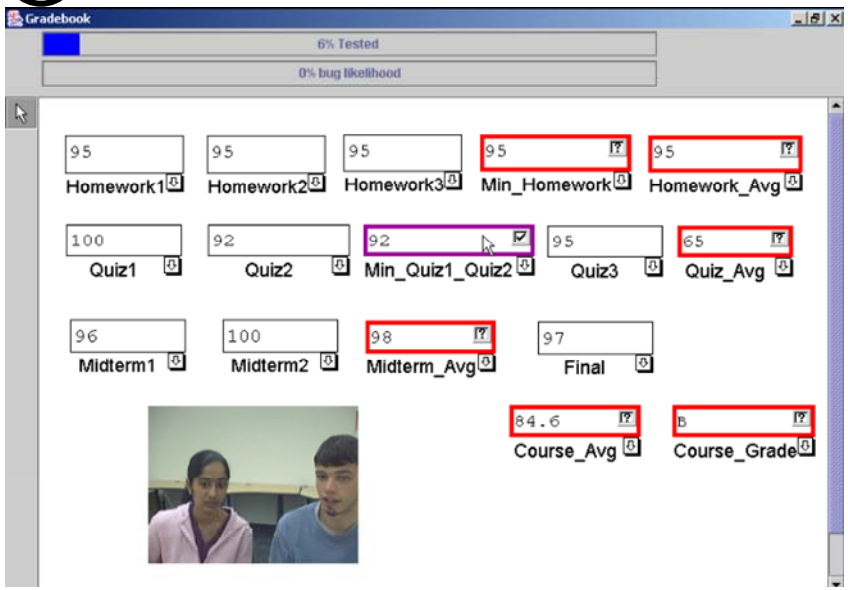


Figure 11: Entry point-the demo can be accessed by clicking the "Show Me!" link embedded inside a feature-centric tool tip.

Storyboard 1: Overall

1

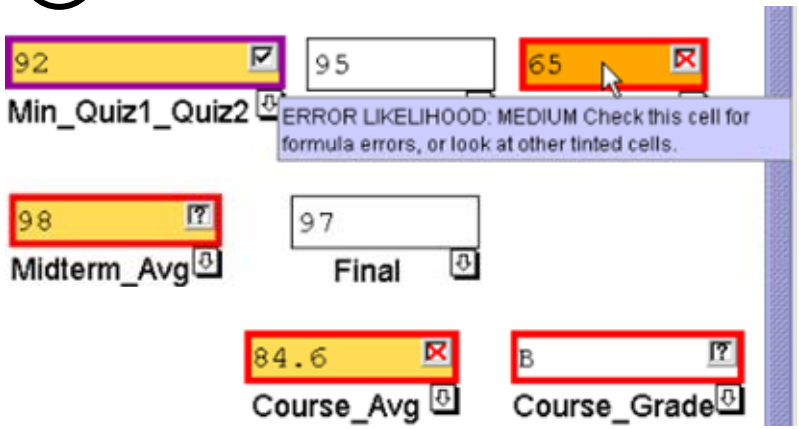


R1: In general, what should I be trying to do?

R2: Well, you should try to make testing progress by checking off cells with correct values in them.

R1: Ok, the value of the Min_Quiz1_Quiz2 cell looks right so I'll check that one off.

2

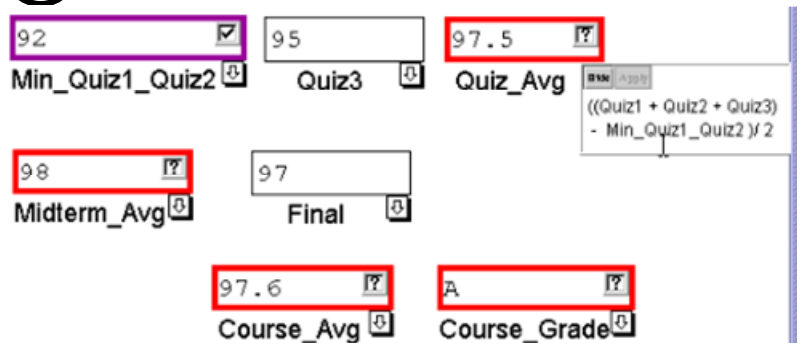


R2: If you find wrong values, use the X-marks and checkmarks together to help find errors in the spreadsheet.

R1: These two look wrong, I'll X them. Looks like the error is in Quiz_Avg.

Storyboard 1 (continued)

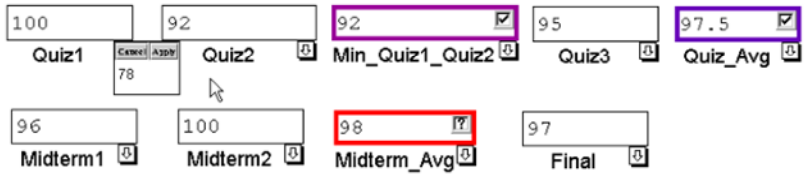
3



R2: Carefully fix any errors you find and then retest your new values.

R1: Ok I'll fix this formula by changing the 3 to a 2, now the value is right, so check!

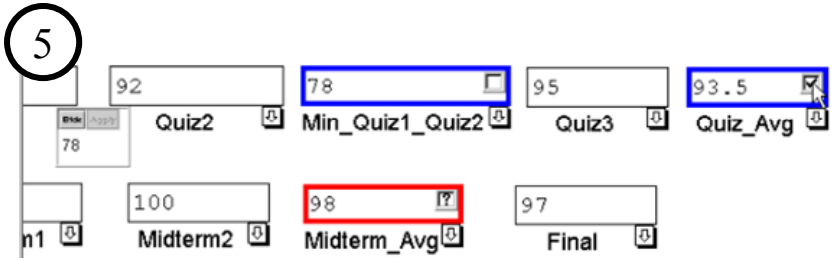
4



R2: Changing input values will aid you in each of these tasks.

R1 Ok I'll change the value of the Quiz1 cell, oh ya it changed and it still looks right so I'll check it again!

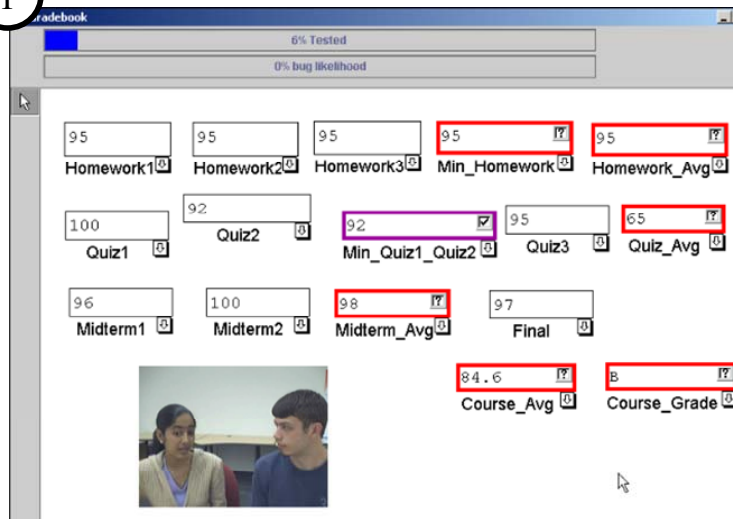
5



R2: If you would like more details there are separate demonstrations over each of these steps. Just follow the links!

Storyboard 2: Making Testing Progress

1

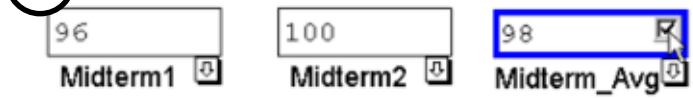


R1: How do I make progress in testing my spreadsheet?

R2: Cells with ?s in their decision box contain values that haven't been tested. To make testing progress, first find one of these values that you can tell is right and put a checkmark on it.

If the border of that cell then turns to blue, that cell has been tested enough, but more testing is always a good idea.

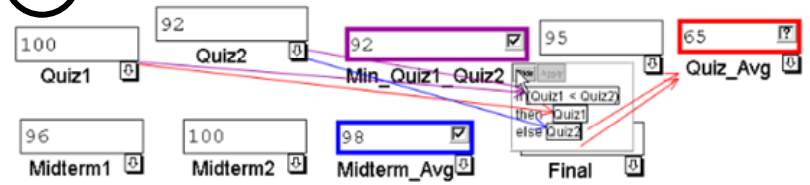
2



R1: What if the border turns to purple?

R2: That means there are situations involving this cell that still haven't been tested. You will need different values to find these situations. The arrows also show testing progress and can help you see the different situations.

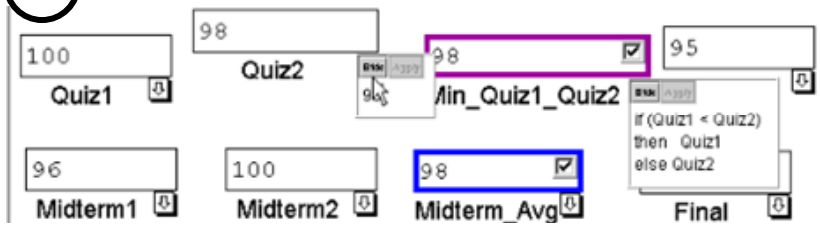
3



R2: If you open the formula tabs for two cells that an arrow connects you will see all of the situations for the two cells and the individual testedness of these situations.

R1: Oh, the "else" situation that has the red arrow pointing to it is the one I haven't tested so I'll change the value and test this one again.

4

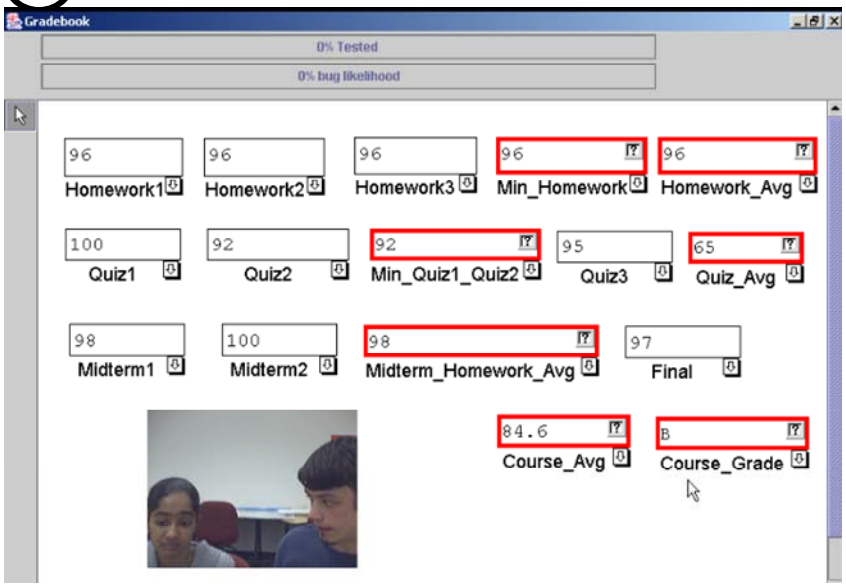


R1: Ok, this value still looks right so I'll check it, but wait, the border color didn't change

R2: Looks like your new value still doesn't get into the "else" situation, a ? in the decision box tells us this is a new testing situation. Your goal for changing values to make testing progress should be to get a ? to appear in the cell.

Storyboard 3: Finding Errors

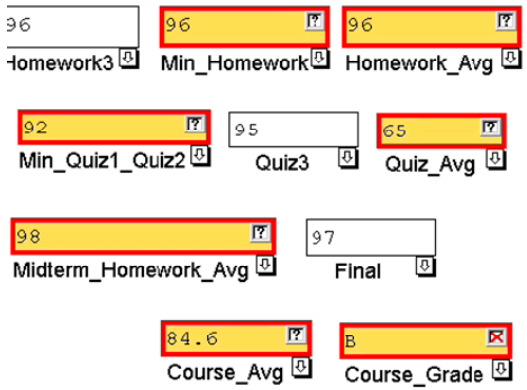
1



R1: Look, in this Gradebook spreadsheet this student has scores above ninety on all quizzes and exams, but their Course Grade is a B so something is wrong here. How do I find the error?

R2: In general, if you can tell that some of the values on the spreadsheet are either wrong or right, Xing out wrong values and checking off right values will help you to find errors in the formulas. If you can't tell if a value is right or wrong try changing the values in the uncolored cells.

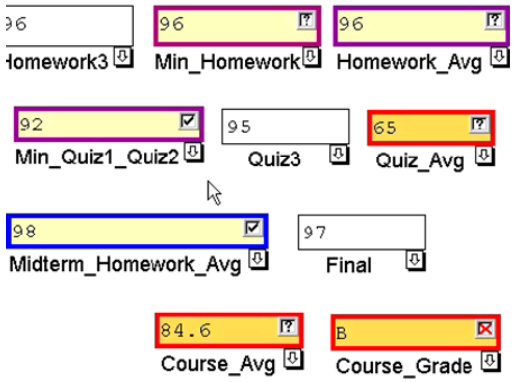
2



R2: The coloring of cells' interiors show the system's guesses at which cells are likely to have an error in them. Probably not all of these cells have an error, but at least one of them does.

The more checks and Xs you place, the better the guesses get. Can you tell if any other values are right or wrong?.

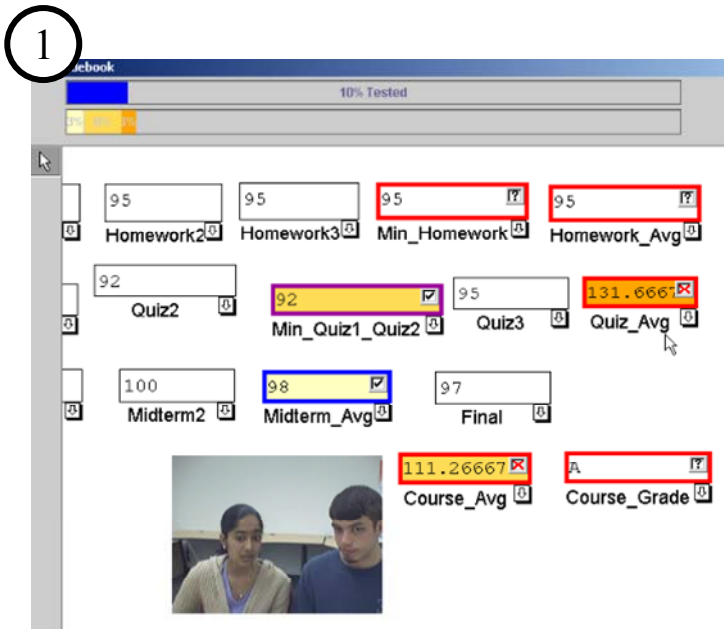
3



R1: Yes, the value of the Midterm_Homework_Avg cell looks right, and so does the value of the Min_Quiz1_Quiz2 cell

R2: Good, so now the system has narrowed our search for the error from 7 cells down to 3. Try that on your own spreadsheet!

Storyboard 4: Fixing Errors

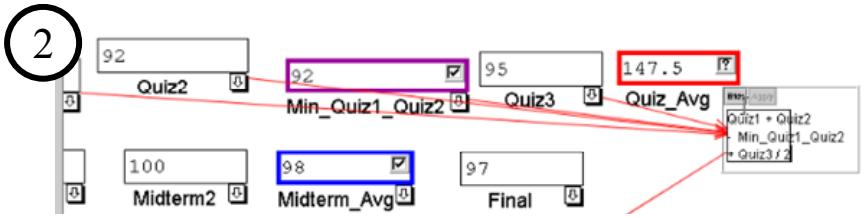


R1: I can tell where the error is but how do I fix it?

R2: After locating the cell likely to have an error in it carefully look at the formula and try to decide if it is doing what it is supposed to.

For example I know that the Quiz Average is supposed to be the average of the third quiz and the higher of the first 2 quizzes.

R1: Looking over the formula it looks like it is averaging 3 quizzes instead of just 2.

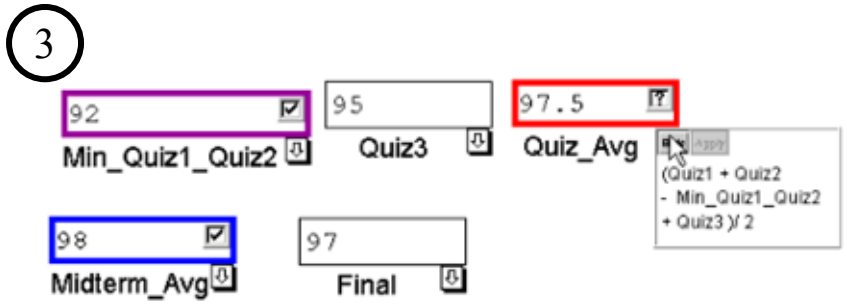


R2: In general there are four things that are likely to be causing the error in the formula:

1) As you have just seen, one of the constants, or numbers in the formula is wrong

2) There is a missing or wrong cell reference in the formula, one quick way to check this is to turn on a cell's arrows to see which cells it is referencing

R1: It looks like it is referencing all the right cells.



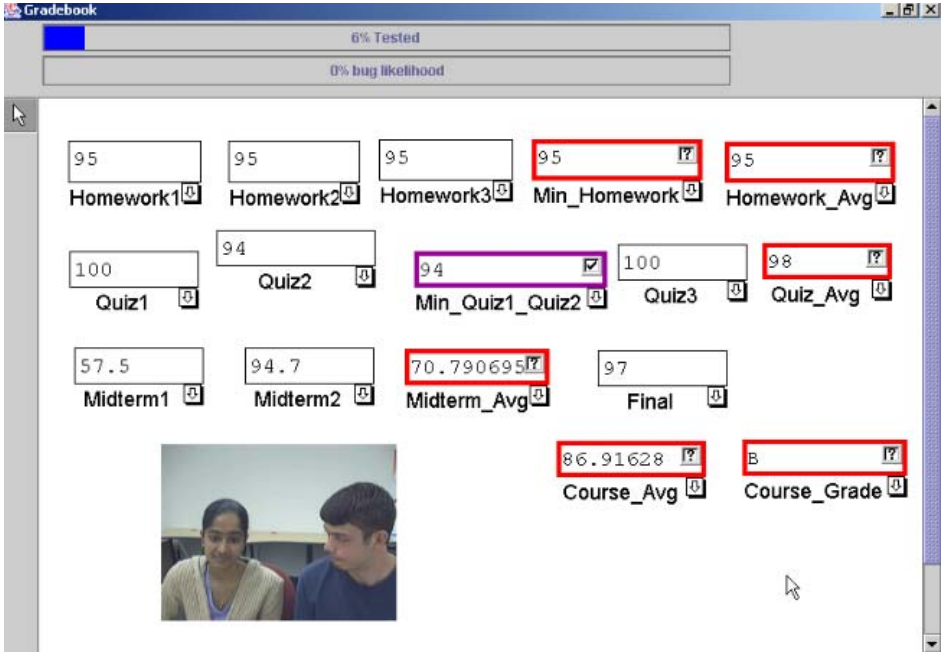
R2: 3) The operators, like +, -, *, /, could be used incorrectly

R1: The operators look right too

R2: Lastly, the parentheses, many formulas use parentheses to group numbers, cell references, and operators, make sure that yours are grouping correctly.

R1: Oh ya, so since there aren't parentheses it is only dividing Quiz 3 by 2 instead of dividing the sum of all the quizzes so I need to add some parentheses around all of this.

Storyboard 5: Changing Values



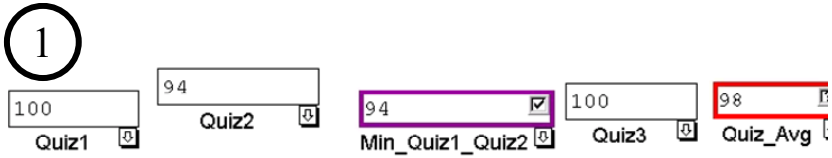
R1: You mentioned that I need to change values, why would you want to change values?

R2: There are two reasons you would want to change values:

1) Find situations you have not yet tested

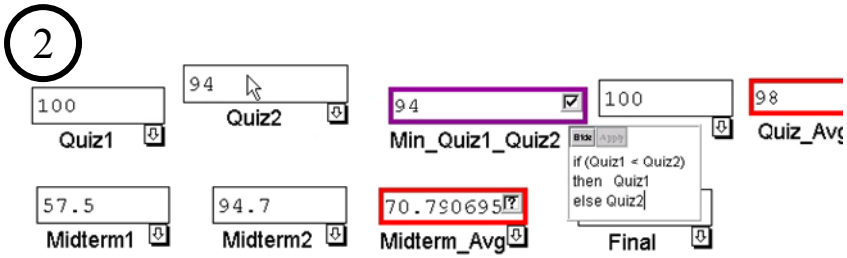
2) Make it easier to tell if values in cells with formulas are right or wrong

Storyboard 5a: To Make Testing Progress

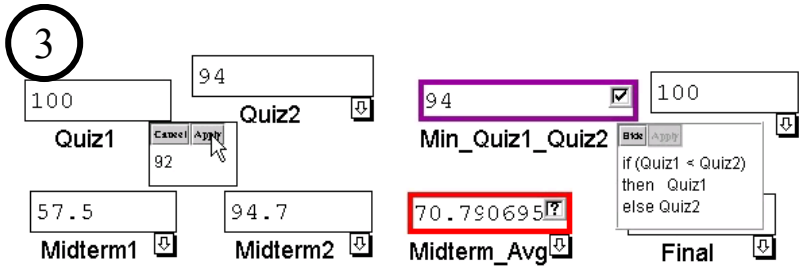


R1: How do I change values to find situations I have not yet tested?

R2: If you are trying to find situations you have not yet tested, it would be a good idea to look at some formulas.

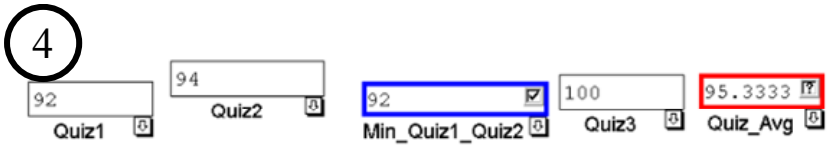


If there is a comparison of two values (i.e. >, <, =) you may want to try a situation where one value is greater than the other, one in which it is less than the other, and one where they are equal.



R1: Oh so since this one says "if Quiz1 < Quiz" and I have already tested Quiz1 bigger than Quiz2 I should try a situation where Quiz2 is bigger?

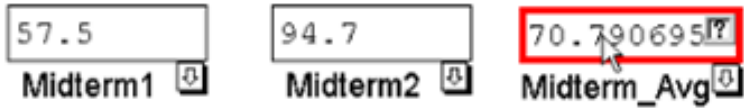
R2: Exactly



R1: That still looks right so I'll check it off.

Storyboard 5b: To Make Values Easier to Decide About

1

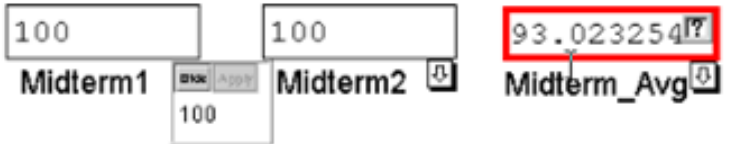


R1: How do I change values to make it easier to tell if Midterm_Avg is right or wrong?

R2: Sometimes it's hard to tell if a value is right or wrong.

Changing values can sometimes make the decision easier.

2



R2: Try putting values like 0, 1, or multiples of 10 into cells since these values can make it easier to tell if a formula has the right arithmetic.

R1: Oh so if I change the values of Quiz1, Quiz2, and Quiz3 cells, it will be easier to tell if the value in the Quiz_Avg cell is correct or not