Gender Differences in End-User Debugging Strategies

Laura Beckwith*, Valentina Grigoreanu*, Neeraja Subrahmaniyan*, Susan Wiedenbeck†, Margaret Burnett*, Curtis Cook*, Karin Bucht*, Russell Drummond*

*Oregon State University
Corvallis, Oregon
{beckwith,grigorev,subrahmn,burnett,cook,bucht,drummond}@eecs.orst.edu

†Drexel University
Philadelphia, Pennsylvania
Susan.Wiedenbeck@cis.drexel.edu

ABSTRACT

Little is known about the strategies end-user programmers use in debugging their programs, and even less is known about gender differences that may exist in these strategies. Without this type of information, end-user programming systems cannot know the "target" at which to aim, if they are to support male and female end-user programmers' debugging. In this paper, we present a study investigating this issue. We asked a group of end-user programmers to debug spreadsheets and to describe the strategies they used to carry out this task. Using quantitative and qualitative methods, we analyzed the strategies reported, considered whether the strategies could be confirmed by observations of participants' behaviors, and looked for relationships among participants' strategy choices, gender, and debugging success. Our results indicate that males and females debug in quite different ways, that there are considerable opportunities for improving support for end-user debugging strategies for both genders, and that the types of features commonly found to aid debugging may be especially deficient in supporting strategies the females prefer to use in debugging.

Author Keywords

Gender, debugging, end-user programming, end-user software engineering, strategy.

ACM Classification Keywords

D.2.5 [Software Engineering]: Testing and Debugging; H.1.2 [Information Systems]: User/Machine Systems—Human factors; H.4.1 [Information Systems Applications]: Office Automation—Spreadsheets; H.5.2 [Information Systems]: Information Interfaces and Presentation—User Interfaces (D.2.2, H.1.2, I.3.6)

INTRODUCTION

Are there gender differences in the ways male and female end-user programmers go about their problem-solving?

Since we first posed this question three years ago [3], we have been working to understand gender differences that arise in a particularly problem-solving intensive form of work, namely end-user programming. We have discovered gender differences in willingness to approach and to eventually adopt new features [4], differences in attitudes toward software features [6], and differences in playful tinkering with features [5]. Other results of gender differences in software-based tasks are also beginning to emerge in the literature [7, 9, 14, 23, 28].

However, the gender differences reported so far do not consider the question of strategy. Strategy refers to a reasoned plan or method for achieving a specific goal. If there are differences in the strategies male and female end-user programmers would like to follow in their software-based problem solving, and if their preferred problem-solving strategies are not well supported by the end-user programming environments available to them, this could lead to loss of productivity by the gender affected. Without taking relevant gender differences into account in the design of environments used by end-user programmers, obstacles can be introduced into the software that could impact the success of half the population the software is intended to support.

This paper reports the results of an experiment we conducted to investigate the strategies used by male and female end-user programmers in the course of debugging spreadsheets. We designed the experiment to capture data suitable for quantitative analysis, and to some extent, qualitative analysis as well. We collected participants' own reports of their strategies, and analyzed our data to obtain corroborating evidence through observed behavior patterns that might substantiate the participants' self-reported patterns. Most important, we compared the strategies used by males and females and evaluated the relationship of these strategy choices to males' and females' debugging success. Thus, our overall research question was:

Are there gender differences in the strategies used by male and female end-user programmers for debugging their spreadsheets?

BACKGROUND AND RELATED WORK

Although there are no previous reports of research into gender differences in debugging strategies, there have been many reports of strategy differences in other domains, such as psychology and education (e.g., [12]). However, these differences have not been investigated in the realm of programming environments, which provide specialized tools that guide, support, and potentially influence strategy.

There has, however, been significant research into novice and expert programmers in various programming-related tasks. Novice programmers are related to end-user programmers, but not quite the same. We use the term "novice programmers" to describe people who aspire to become professional programmers, but are just beginning their training. We contrast this with end-user programmers, who do not usually aspire to become professional programmers.

Research shows that novice and expert problem solvers approach problems using different strategies. Novices typically approach a problem using a depth-first approach, pursuing the solution of a single sub-problem to its end, then sequentially move on to solve the next sub-problem [11]. Expert problem solvers use a breadth-first approach, iteratively developing solutions to sub-problems while keeping the solutions of the sub-problems at the same level of detail [11]. The advantage of the breadth-first approach is that the problem solver gains of a wider view of the evolving solution. Novice problem solvers reduce their cognitive load by isolating sub-problems, although at the disadvantage of possibly discovering crucial interactions late in the solution. It is reasonable to expect that end-user programmers might adopt strategies similar to novice programmers'.

The strategy differences discussed above are also reflected in the area of program debugging. Success in debugging is strongly related to program comprehension. Unlike novices, expert programmers attempt to gain a high-level of understanding of the program before they begin to debug [15, 19, 25]. They also use a different strategy to read a program than do novices [15, 19]. In the imperative programming paradigm, experts read a program in the order in which it is executed, which provides a hierarchical understanding of the program at many levels. Novices, by contrast, read the program sequentially like reading a book, leading to a less coherent understanding of parts of a program and their possible relevancy to the bugs.

Some of our own previous work provides preliminary information on strategies adopted by end users in debugging of spreadsheets. In a study of fault localization [22], users adopted two kinds of strategies when they noticed an incorrect value in the spreadsheet. The ad hoc strategy consisted of examining cell formulas randomly. The dataflow strategy consisted of following the dependencies back from the cell with the incorrect value through cell references until they found the fault. The results showed that the dataflow strategy was more successful than the ad hoc strategy overall, and this was particularly true in a group that received visual fault localization feedback.

None of the above studies has investigated differences in strategy between males and females. However, in other preliminary work we used data mining on transcripts of male and female participants in a previous study of end-user spreadsheet debugging to identify patterns of feature usage by males and females [13]. Interesting insights emerged from the data mining. First, we discovered that patterns of feature usage by males successful in debugging were similar to female who were *not* successful in debugging, suggesting that male and females' optimum strategies may be different. Second, unsuccessful males used a much higher number of patterns involving the dataflow arrows feature than successful males, suggesting that unsuccessful males may have relied entirely on arrows in their testing decisions, while ignoring the current state of program execution.

In summary, the above literature points out ways that strategies lead to success in programming problems. This paper extends this work by investigating some of the strategies discussed above, as well as other end-user debugging strategies, with the addition of gender as a factor.

EXPERIMENT

Participants and Procedures

There were 61 participants: 37 females and 24 males. The participants were undergraduates from a variety of majors. They had prior experience using spreadsheets, but very limited programming experience. A pre-experiment questionnaire, based on Compeau and Higgins' validated scale [10] contained 10 self-efficacy questions specific to end-user debugging tasks.

A 25-minute hands-on tutorial was presented to familiarize participants with the spreadsheet features. Subsequently, participants carried out two experimental tasks. Participants' actions and the system's feedback were captured in electronic transcripts, along with their final spreadsheets.

A final post-session questionnaire included questions assessing participants' comprehension of and attitudes toward the features they had used. The post-session questionnaire also contained an open-ended question that asked the participants what they perceived their own strategies to be for finding and fixing errors.

Environment and Software Features

The environment used in the study is a research spreadsheet environment that includes explicit support for test-

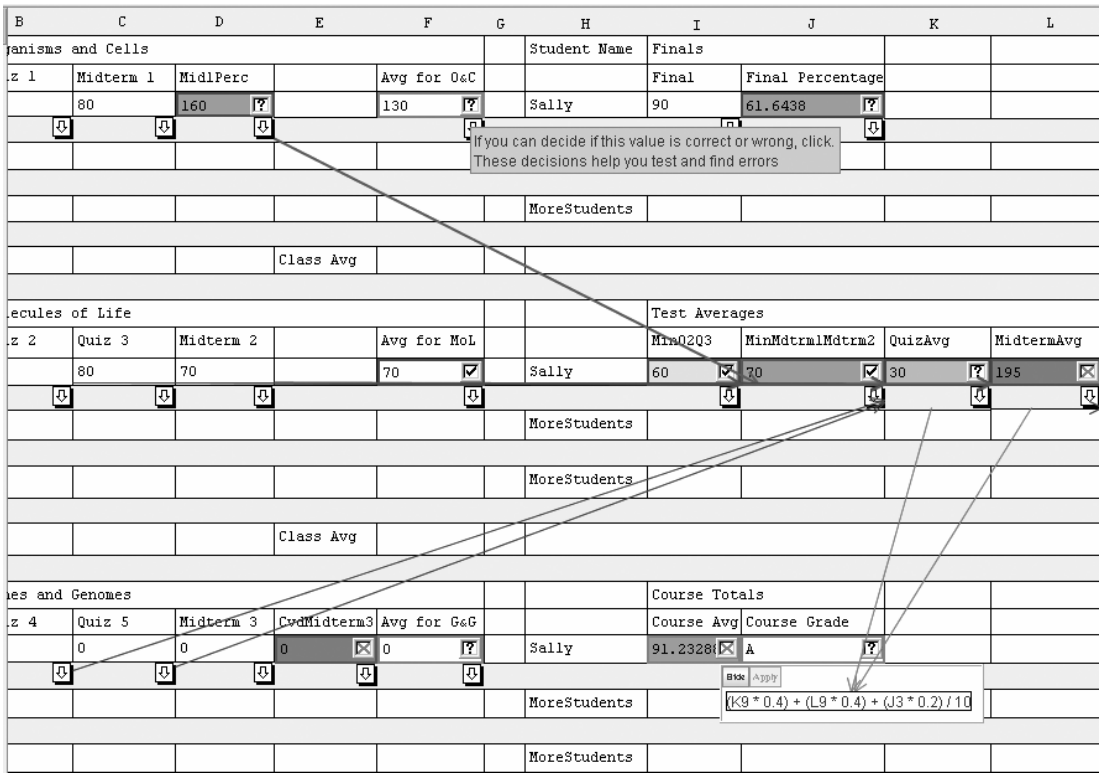


Figure 1. The Gradebook spreadsheet used in our experiment. The user notices an incorrect value in `Course_Avg`—the value is obviously too high—and places an X-mark in the cell. As a result of this X and the checkmark in `MinMdtrm1Mdtrm2`, eight cells are highlighted as being possible sources of the incorrect value, with some deemed more likely than others.

ing and debugging by end-user programmers in the form of WYSIWYT ("What You See Is What You Test"). WYSIWYT is a collection of testing and debugging features for end-user programmers [8]. We chose to use our research spreadsheet system because its extensive set of debugging features provides participants more choice of testing and debugging features than Excel. Our environment also included a logging capability, which provided the ability to collect the extensive activity data necessary for analysis of activity patterns that relate to different strategies. The spreadsheets had an Excel-like appearance, since participants were familiar with Excel.

With WYSIWYT, at any time, the user might notice that some cell's value is correct, at which point he or she can check it off. Borders of untested cells are red (light gray in this paper), partially tested cells are a shade of purple (intermediate shades of gray), and fully tested cells are blue (black). The optional dataflow arrows (seen in Figure 1) are also colored to reflect testedness of specific relationships between cells and sub-expressions. For example, if a user checks off the MinQ2Q3 cell in Figure 1, the system updates all affected cell border colors that fed into the answer of MinQ2Q3, the color of any visible dataflow arrows, and a "tested %" progress bar at the top of the spreadsheet (not shown), all reflecting the new testedness.

Instead of noticing that a value is correct, the user might notice that it is wrong, and can "X it out" (cell Course_Avg in Figure 1) instead of checking it off. Behind the scenes, X-marks trigger fault likelihood calculations, which cause cells suspected of containing faults to be highlighted in shades along a yellow-orange continuum (shades of gray in this paper), with darker orange shades given to cells with increased fault likelihood [8].

In addition, the arrow tabs in the bottom of each cell are used to open the cell's formula. Once a formula is made visible (as in the lower right of Figure 1), it stays visible until the user comes back to close it. This device allowed participants to have multiple formulas open simultaneously, increasing the viability of debugging strategies based on code inspection, if a participant was so inclined. These and all features in the environment were supported with tooltips (top center of Figure 1).

Tutorial

To avoid suggesting strategies to our participants, the tutorial was taught simply as a "tour of features." The tutorial covered features available in the spreadsheet environment, without any problem-solving scenario that might suggest how to build a strategy using the feature.

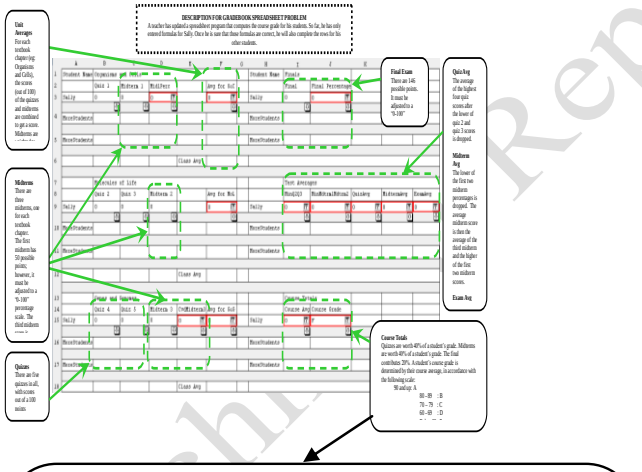
The tutorial covered six features: Tool tips, Checkmarks, X-Marks, Arrows, Formula Edits, and Help Me Test. Par-

ticipants also received a one-page quick-reference style handout with all the features, to help them stay oriented in the tutorial and to refer to later in the experiment. In the tutorial, the participants got explanations of the features and hands-on practice. At the end of the tutorial, they were given time to further explore the features by working on a practice spreadsheet debugging task.

Tasks

Participants tested two spreadsheets, Gradebook (Figure 1) and Payroll. Other than the layout, the spreadsheets were and the seeded faults were the same as those of [4], with a few adjustments to accommodate the changes in layout. The spreadsheets were derived from real end-user spreadsheets written in Excel, and were seeded with faults we harvested from end users. There were a total of 11 faults representative of the fault categories in Panko's classification system [21], six in Gradebook and five in Payroll. The layouts were done in an Excel-like manner. We designed the layout so as to avoid potential confounds among different sequences participants might follow. For example, Western reading order was distinguishable from description order, various dataflow orders were distinguishable, and so on.

The participants were given the spreadsheet, a hardcopy description of the spreadsheet (Figure 2), and two hardcopy examples of the spreadsheet with correct values. The time limits for the debugging tasks were 22 minutes for Gradebook and 35 minutes for Payroll. The time constraints were meant to simulate time constraints frequently encountered in real world computing tasks and to prevent experimental confounds, such as participants spending too much time on the first task or not enough time on the second task, participants leaving early, etc. The tasks were counterbalanced. Also, the order in which the hardcopy handouts were collated for the participants was random across tasks, in order to avoid any systematic influences on participants' strategy choices. The participants were told that a spreadsheet had been updated and that, "Your task is to test the updated spreadsheet and if you find any errors, fix them."



Course Totals
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
80 - 89 : B
70 – 79 : C
60 - 69 : D
Below 60 : F

Figure 2: The description handout added call-outs explaining groups of cells on the spreadsheet. (Miniaturized for space reasons, with one of the callouts blown up for readability.)

RESULTS

We begin with the end: who had the most success debugging? We used an edit to a faulty formula as a surrogate for bugs found (even if the edit was not correct), and we used changing the formula correctly as our measure of bugs fixed. As Table 1 shows, the males outperformed the females. Males tended to have more success than females in finding bugs ($F[1,59]=3.39$, $p<.071$) and were significantly more successful in carrying forward those "finds" to actually fixing the bugs ($F[1,59]=12.20$, $p<.001$).

This result corroborates those of another study in which male end-user programmers significantly outperformed the females in some aspect of debugging [4]. To investigate how this result may have related to strategy choices, we first turn to the participants themselves to understand the debugging strategies they were pursuing.

For the purposes of investigating strategies and how well they appeared to be working for our participants, we categorized the participants as "successful" or "unsuccessful", depending on whether they fixed the median number of bugs (5.5). In this manner, we can compare strategies used by successful females with those of the successful males. Of the 37 female participants, 14 were categorized as successful and 23 as unsuccessful, and of the males, 16 were successful and 8 were not.

What strategy did participants *say* they were using?

The concept of strategy includes the notion of mental planning with intent. Since it is not possible to observe mental planning and intent directly, we first collected what participants *said* their strategies were (on the post-session questionnaires), and second for indications in their observable actions that might provide evidence bearing

Category	Bugs Found	Bugs Fixed
Males (n=24)	8.45 (2.19)	6.71 (2.46)
Females (n=37)	7.14 (2.92)	4.25 (2.83)

Table 1. Mean (SD) number of bugs found/fixed.

out their reported strategies.

We derived an initial set of codes from the participants' responses. The codes focused on mentions of strategy per se, and on mentions of particular artifacts as being important to their strategy. Two coders independently coded subsets of the participants' responses and compared them, developing the codes further and iterating until an acceptable level of agreement was achieved. At the point the agreement rate achieved 84%, demonstrating a reasonably robust set of codes, a single coder coded all remaining responses.

The strategies fell into five major categories (Table 2), with an three additional fine-grained strategic concepts.

Testing. Many participants described their strategy in terms of testing, or said that a testing-oriented artifact (feature or example values handout) was important to their strategy.

Code Inspection. Testing and Code Inspection are complementary techniques, and our end-user participants used both of them. Code inspection was defined for our purposes as looking at formulas to judge them.

Specifications. Related to code inspection is the idea of specifications. Some of our participants reported guiding their efforts by the specifications, (provided by the description handouts in our study).

Dataflow group. Describing any sequence related to dataflow was classified as dataflow, as well as those who said they used arrows to guide them, since arrows are dataflow-based. This was a popular category among the successful males

Spatial. A third sequence was spatial, following the layout of the spreadsheet, such as in Western reading order. Interestingly the successful males and unsuccessful females were the ones reporting this strategy.

In addition, three lower-level descriptions of strategies were identified: Those who described a "things to do" oriented strategy to guide their efforts (5 participants) and those who described working in a very incremental fashion (3 participants). Since the number of responses was small, we will not discuss these lower-level strategies further.

	Testing (9)	Code Inspection (3)	Specification Follower (2)	Data–flow (4)	Spatial (1)
SM (n=16)	2.06 (23%)	1 (33%)	1 (50%)	2 (50%)	0.06 (6%)
SF (n=14)	2.36 (26%)	1.29 (43%)	1.57 (79%)	0 (0%)	0.07 (7%)
UM (n=8)	2.5 (28%)	1.13 (38%)	0.75 (38%)	0 (0%)	0 (0%)
UF (n=23)	1.91 (21%)	0.83 (28%)	1 (50%)	1.5 (38%)	0.09 (9%)

Table 2. The headings show code categories and their maximum score. Scores count the number of concepts or artifacts mentioned related to that category. The cells show the mean score and percentage of the maximum score for successful males/females and unsuccessful males/females.

The raw numbers and percentages of the maximum scores give slightly different views of the data, but both emphasize successful males' reliance on dataflow. This is in direct contrast to the successful females, *none* of whom mentioned any dataflow strategy or artifact at all in a positive manner. As we will discuss later, the successful males' strategies differed significantly from the successful females' strategies in other ways as well.

Strategies and observable behavior patterns: Examples

If participants used the strategies they indicated, there should be some indication in their data through their behavior patterns. For example, a participant who said their strategy was testing might have achieved high "testedness" scores in the environment, might have used checkmarks and/or X-marks reasonably often, or, even if he or she did not rely much on the environment's testing features, might have edited cell values reasonably often to try out different test cases.

In this section, we report qualitatively observations of several interesting participant behavior patterns, to demonstrate concretely how they can contribute credibility to the participants' stated strategies.

We made the observations by re-playing their log files through the system, which allowed us to relive their work multiple times, observing every action they made and seeing the same results and feedback as they saw. As our way of observing their progression through cells, we defined a "visit" to a cell as any physical touch by the user to a cell, such as checking off its value, opening its formula, and so on.

First, consider the strategy of following the description. A female participant described her strategy as "going for the information given on the handouts and making sure each formula fit that description, looking at the resulting numbers and making sure they all made sense, if not I'd go back to the formulas…"

She behaved in a way that clearly demonstrated her adherence to this strategy. For example, in the Gradebook problem, she began by visiting the cells described by the first call-out in the printed description (recall Figure 2), as per Western reading order of the description, left to right and top to bottom. After several minutes of visiting cells in this area, she then moved to the two cells that calculate the "Final Exam" score, which are explained together on the description's second call-out. About 8 minutes into the task, she moved to the cells described by the third call-out of the description, namely the midterm exams

call-out. This call-out describes cells that come *after* the next ones in Western reading order in the spreadsheet itself, which shows adherence to the description's order rather than to the order of the spatial layout of the spreadsheet. Five minutes later she moved on to the section of the spreadsheet described by the next call-out in the description, the "Test Averages" section. Her first visit was to the middle cell in that group, "Quiz Average"—which is the first cell mentioned in the description's call-out.

There are six call-outs in the description, and she ultimately visited cells in five out of those six in the order listed on the description. Thus, her reported strategy was clearly demonstrated by her behavior patterns for the spreadsheet.

Many of the males described strategies relating to dataflow. For example, successful male wrote about being guided by the flow of data/information: "[I started] with the handouts to determine what the spreadsheet is used for and how it's supposed to work. Starting in the upper-left region of the spreadsheet try to follow the flow of information to eventually determine if the spreadsheet is functioning correctly." In a way highly consistent with this self-described strategy, his actions showed initial exploration, followed by focuses on cells as grouped by dataflow chains.

For example, in Payroll, he began with a combination of ad hoc code inspection and testing, looking at a few cell's formulas and putting in several of the example handout's values. About 5 minutes into the task he turned on dataflow arrows, and immediately followed one of the arrows back (dataflow upstream). He frequently used arrows, and each time he followed them along the dataflow path, generally finding out what cells affected the cell he was interested in, moving "upstream" to those cells. He used arrows a total of 8 times in Payroll. He tended to stay within the flow of one dataflow chain at a time. When he left a dataflow chain, he reliably would then switch his focus to the new dataflow chain, one chain at a time.

As his self-described strategy mentioned, he "follow[ed] the flow of information," which was clearly evident in his use of arrows to guide him on this strategy. In contrast to the female, after some early exploration, this male let the dataflow guide his movement around the spreadsheet, rather than the description.

The evidence from these three particular participants' logs of their behavior matches closely with their self-reported strategies. From this jumping off point as insight into the strategies they reported following, we follow up on specific strategies with statistical evidence supporting these.

Statistical signs of strategy differences

The results thus far suggest gender differences in strategies that participants talk about and, based on qualitative investigation of logs, their descriptions of strategies are observable in their behavior. The next step in understanding the gender differences in strategies and behavior is to look for these differences quantitatively in the logs.

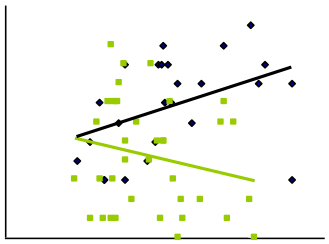


Figure 3: Total bugs fixed predicted by frequency of dataflow, between genders. Darker color-male, lighter-female

Specifically, we looked for evidence of dataflow behaviors by the males compared to the females, and looked for code inspection by the females compared to the males.

Dataflow: males' vs. females' behaviors

In the dataflow strategy individuals debug by considering cells that are related through cell references. These related cells form a dataflow chain, where the beginning of the chain is an input cell (e.g., a value such as Sally's Quiz 2 in Figure 1) and the end of the chain is a desired output (e.g., Sally's course grade in Figure 1). The cells between the input and output are related because "upstream" cells are referenced in the formulas of cells farther down the chain.

There are two kinds of dataflow strategies, depth-first and breadth-first. In the depth-first approach the individual debugging a spreadsheet stays in the same chain, i.e., visiting cells that are all related to each other through cell references. The individual may visit a cell in any order and as many times as desired as long as they stay in the same chain. By contrast, in the breadth-first approach the individual stays at the same level of dataflow, visiting cells from different chains that are at that same level as often as they want, before moving on to the next level.

Several of the features available in the environment support depth-first dataflow traversal through the spreadsheet. Arrows, for example, are one of these features; they provide feedback about the relationships of cells further along the dataflow chain. When the user turns on the arrows for a given cell he or she immediately sees which cells are related through cell references. In a similar manner, X-marks provide feedback on cells earlier in the dataflow chain. When the user places an X-mark, the cell interiors of upstream cells in the dataflow chain are colored to show the likelihood of bugs in these related cells. Both features may guide users into depth first dataflow within their testing and debugging process.

To investigate whether participants followed a depth first dataflow we assigned each participant a depth first score, representing how often they moved between cells in a

dataflow chain. When a participant moved from one cell to another in the dataflow chain they received a point. These points accumulated until they exited that dataflow chain. We also considered a count of the instances a participant entered into a depth first chain.

During the first task the males followed depth first dataflow significantly more than the females, as measured both through the average chain length of the depth first dataflow (males: 3.27(0.73) females: 2.89(0.67); ANOVA: $F(1,59)=4.56$, $p<0.037$), and more often entered into a depth-first search strategy (males: 84.29(71.82), females: 64.27(40.92); ANOVA $F(1,59)=4.39$, $p<0.04$).

For the males, their use of the depth first search strategy (measured through depth first dataflow score, which combined their average chain length with the maximum chain length) is statistically tied to their arrow usage over both experimental tasks (linear regression: $F(1,22)=20.68$, $p<0.0002$, $\beta=0.077$, $R^2=0.48$). For the females, however, their arrow usages is not related to their usage of the depth first search strategy (linear regression: $F(1,35)=3.74$, $p<0.59$, $\beta= -0.024$, $R^2=0.008$). In other words, arrows helped to guide the males into a depth first search strategy, but when the females used arrows there was no relationship to their overall depth first search strategy.

Furthermore, males' use of depth first dataflow was significantly predictive of task success. The number of instances depth first dataflow predicted how many bugs they eventually fixed (linear regression: $F(1,22)=2.28$, $p<0.04$, $\beta=0.0085$, $R^2=0.18$). For females the same relationship does not hold, in fact the relationship suggests a slightly negative effect on bugs fixed when using a depth first search dataflow strategy (linear regression: $F(1,35)=2.82$, $p<0.27$, $\beta= -0.006$, $R^2=0.035$). Both of these regression relationships are pictured in Figure 3.

In fact, this corroborates the successful males' discussion of strategies, they had significantly higher discussion of strategies related to dataflow than everyone else (successful males 0.25 (.82), successful females 0 (0), unsuccessful males 0 (0), and unsuccessful females 0.043 (.84); ANOVA $F[3,57]=3.04$ $p<.036$).

Code inspection: males' vs. females' behaviors

Do females do more code inspection? We have mentioned in our code analysis that females consider their strategies to include code inspection more often than the males. However, directly measuring code inspection is difficult; in the process of looking for errors one must look at formulas (inspect the code). We opted to look for indications of "non-code inspection," investigating how often input cells were edited. Editing input cells offers an indication of checking specific different test cases for formulas. We suspected that participants engaged more with code inspection would do less testing with different input values, and those deciding to test their formulas with multiple values (recall the males did significantly more overall testing of their spreadsheets) were not relying solely on code inspection. Males edited input cells significantly more often than the females ($F[1,59]=7.61$, $p<0.0077$; Males: 48.7 (27.93); Females: 31.6(13.62))

What worked for males, what worked for females?

In this section we present a mixture of qualitative and quantitative evidence about the strategies that led to success and failure by males and females.

The selectivity hypothesis [18] proposes that the genders differ in their strategies for information processing. It predicts that males' information processing is characterized by striving for efficiency. Therefore, males have a propensity to use simple heuristics in information processing (e.g., single cues that are readily available) in order to reduce cognitive load. They deviate from the heuristic strategy only if they are forced to do more elaborative processing because of the needs of a complex task. This kind of simplifying processing strategy has also been called an "as-needed" strategy [17] because the user avoids broad comprehension by only processing information as needed for the situation at hand. By contrast to males, females tend to maximize the comprehensiveness of their information processing, looking for multiple, subtle cues, paying attention to detail, and making elaborative inferences. The hypothesis predicts that females are likely to employ detailed, elaborative information processing strategies in both simple and complex decision tasks.

Empirical research by O'Donnell and Johnson [20] supports the selectivity hypothesis. Their study, based on simple and complex auditing tasks, found that there were interactions between the task complexity and gender. In particular, for the simple task females took longer and processed more cues than the males did (since the males used a heuristic approach). However, on the more complex, analytical task the males adopted comprehensive processing similar to that of the females on the simple task.

To understand the success of males' strategies we compared the unsuccessful males to the successful males. Our question was whether the unsuccessful males used heuristic, as-needed strategies while the successful males used comprehensive strategies. From different sources of information it appears that the answer is yes.

First, in the Gradebook spreadsheet, successful males did more visits to cells (i.e., using checkmarks, X-marks, arrows, formula edits, and posting formulas) than unsuccessful males, although the differences were not often significant. An example of low visits by unsuccessful males is cell L9, which was visited significantly less often by unsuccessful males than the successful males (this cell

is interesting b/c it falls right after a cell "similar" to it in what it need to calculate - but has a different kind of bug).

Second, in the Gradebook spreadsheet unsuccessful males spend their time editing non-buggy formulas significantly more than did successful males (successful males' mean 3.38 (2.4); unsuccessful males' mean 5.88 (2.1), ANOVA $F[1,22]=6.40$ $p<.02$). Successful males, on the other hand, spent their time editing buggy formulas significantly more than did unsuccessful males (successful males' mean 8.5 (4.0); unsuccessful males' mean 4.0 (2.2), ANOVA $F[1,22]=8.55$ $p<.008$). Although all the males appeared to be trying to use a dataflow strategy, the layout of the Gradebook was such that users would not get to the buggy cells before encountering many other cells. It appears that the unsuccessful males began to edit cells prematurely without having a grasp of the whole spreadsheet. This is consistent with the heuristic, as-needed strategy. The successful males' strategy was comprehensive processing.

In the payroll spreadsheet it was observed that the successful males visited two cells significantly more often than the unsuccessful males, cells G13 and H13 (Cell G13: successful males' mean 8.56 (5.32), unsuccessful males' mean 3.25 (2.05), ANOVA $F[1,22]=7.30$, $p<.013$; Cell H13: successful males' mean 13.56 (10.98), unsuccessful males' mean 4.13 (2.4), ANOVA $F[1,22]=5.65$ $p<.027$). Cells G13 and H13 should impact a cell down the dataflow chain, but that cell's formula had a bug, which confused G13 with another cell with a similar name. It is not obvious to spot the bug unless one understands what the two cells H13 and G13 are supposed to be doing in the spreadsheet. Once again, the unsuccessful males' failure to process the cells and relationships comprehensively inhibited their ability to recognize a bug.

Overall in the Payroll spreadsheet, there were few significant differences in visiting cells by the successful males and the unsuccessful males. This could suggest that for Payroll, a more complex spreadsheet than Gradebook, all males realized that they needed to make the effort of comprehensive processing to actually make progress on the task.

While the males differed in their visits to cells, indicating comprehensive processing, the females, and what contributed to their success was a more subtle.

Unlike the males, statistical evidence for visiting cells, and other types of measures vary little between the successful and unsuccessful females. For this reason we turned back to their self-reported strategies, where we did discover differences. For example, successful females discussed strategies related to code inspection grouping nearly significantly more often than the unsuccessful females (successful females: 1.3, unsuccessful females: .83; ANOVA $F[1,35]=3.98$, $p<.054$). While the unsuccessful females discussed strategies relating to border colors more often (successful females: 0, unsuccessful females: 0.22; ANOVA: $F[1,35]=3.7$, $p<.063$).

How do these self-reported differences in strategies play out in their behaviors? In order to investigate this we have to consider how this focus on testing might play out in actual behaviors. Testing is about trying out different values. However, the successful females did not try out more values according to the raw counts of number of value edited (successful females: 59.4; unsuccessful females 66.3; ANOVA $F[1,35] = 0.37$ $p<.55$).

Put into perspective with the unsuccessful females' reported strategies, this is not surprising. The unsuccessful females are also driven to change input values, although for a different purpose, by the border colors, rather than for the purpose of checking different values. In other words, input values must be changed in order for border color to change; a cell's purple border (indicating a partially tested cell) will not turn blue without changing input value. The unsuccessful females change input cells to get their cells blue, successful females change input formulas to "test" their formulas.

Since their different intentions play out similarly in the logs of their actions, this indicates missing features within our own environment. If these two different strategies the females suggest wanting to employ were supported within the environment (for example, with greater support for code inspection), then we could expect to see differences in the behaviors. However, the females had to make do with the features provided in the environment in attempting to achieve their goals. They adapted their own behaviors to the features available within the environment. Therefore, despite two different overall strategies, the differences in behavior are nearly indistinguishable.

DISCUSSION

A recurring theme in our results was that females' strategies and behaviors were mismatched with the features provided within the environment. Bu contrast, the males' preference for depth-first dataflow strategies were well supported, and tied to their success. The females adopted (or attempted to adopt) several strategies unsupported by the features, at times perhaps leading to unsuccessful debugging.

In past research [4] we found gender differences in self-efficacy[1], and its relation to acceptance of new and unfamiliar features (in particular the X-marks). Females were less willing to try out a feature they had not been explicitly taught, and stated that they believed it would take them too long to learn, but in fact their actual understanding did not differ from the males' [4]. Based on our results from this study we provide a new interpretation of that result: perhaps the females' perception of learning cost was influenced not by the time to learn, but by attempting to

[1] Consistent with [Beckwith et al. 2005], in the this study males had significantly higher incoming self-efficacy.

integrate the X-mark feature effectively into their problem-solving strategy.

Programming tools designed for end-user programmers engaged in debugging do not support the type of problem-solving the females in our study attempted to do, but do provide support for the males' preferred problem-solving styles of depth-first dataflow. For example, Excel provides dataflow arrows, and an "evaluate formula" feature, both providing depth-first dataflow information. This is also true of the features provided in other end-user programming environments, including the "Why Line" [16], UCheck [2], and others [1, 26].

From these insights we propose several suggestions to the designers of end-user programming environments attempting to support end-user debugging. First, since males' strategies were supported by the existing types of dataflow (depth-first) features, these features should continue to be well supported in the future. Second, features which encourage comprehensive processing (an observed behavior adopted by successful males, and a behavior the literature [18] suggests is done regularly by females) will provide benefits to both genders. Third, features designed to support code inspection will provide greater support to females' preferences for problem-solving strategies. Finally, features that support dataflow beyond depth-first, such as breadth-first search, will not relegate females to using feature mismatched to their problem-solving preferences.

CONCLUSION

This paper has reported on strategies used by end-user programmers when debugging. Our results contained a number of surprises, including the following:

Males and females: There were significant gender differences in the strategies males and females were using. Further, the debugging strategies that worked well for the males were not the same ones that worked well for the females.

Dataflow strategies: A long-time emphasis of debugging tools for programming environments, and nearly the *only* type of debugging tool for end-user programming environments, has been dataflow depth-first. However, our results indicate that while that was a useful strategy for the males, females infrequently use this strategy.

Testing and code inspection: Testing was a valued strategy by both genders. However, code inspection (alone or in combination with testing) was a strategy preferred mainly by the successful females.

Other strategies: In addition, participants reported strategies that are almost entirely unsupported in end-user programming environments, including code inspection, ability to incrementally check against specifications, comprehensive overviewing, and support for things-to-consider.

These results show that end-user programming environments have several opportunities for improving their support for end-user programmers' debugging strategies. They also strongly suggest that, until support for a greater variety of debugging strategies is added, female end-user programmers especially will face barriers impeding their debugging efforts.

ACKNOWLEDGMENTS

We are grateful to Carlos Jensen, Mary Beth Rosson, and Martin Erwig for feedback and ideas. We also thank the participants of our study. This work was supported in part by Microsoft Research, by NSF grant CNS-0420533, and by the EUSES Consortium via NSF grants ITR-0325273 and CCR-0324844. We especially thank Saturday Academy's Apprenticeships in Science and Engineering Program for the support of our interns.

[1] Abraham, R. and Erwig, M. Goal-directed debugging of spreadsheets, *Visual Languages and Human-Centric Computing*, IEEE (2005), 37-44.

[2] Abraham, R. and Erwig, M., UCheck: A spreadsheet unit checker for end users, *J. Visual Languages and Computing*, (2006, to appear).

[3] Beckwith, L. and Burnett M. Gender: An important factor in end-user programming environments? In Proc. Visual Languages and Human-Centric Computing, IEEE (2004), 107-114.

[4] Beckwith, L. Burnett, M., Wiedenbeck, S., Cook, C., Sorte, S., and Hastings, M. Effectiveness of end-user debugging software features: Are there gender issues? In *Proc. CHI 2005*, ACM Press (2005), 869-878.

[5] Beckwith, L. Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., and Cook, C. Tinkering and gender in end-user programmers' debugging, In *Proc. CHI 2006*, ACM Press (2006), 231-240..

[6] Beckwith, L. Burnett, M., Wiedenbeck, S., and Grigoreanu, V. Gender HCI: What about the software? *Computer* (2006), to appear October.

[7] Bruckman, A., Jensen, C., and DeBonte, A. Gender and programming achievement in a CSCL Environment, CSCL 2002, Boulder, CO, January 2002.

[8] Burnett, M., Cook, C. and Rothermel G. End-user software engineering. *Comm. of the ACM 47*, 9 (2004), 53-58.

[9] Busch, T. Gender differences in self-efficacy and attitudes toward computers. *Journal of Educational Computing Research 12*, (1995), 147-158.

[10] Compeau, D. and Higgins, C. Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly 19*, 2 (1995), 189-211.

[11] Cross, N. Expertise in design: An overview. *Design Studies 25,* 5 (2004), 427-441.

[12] Gallagher A., De Lisi R., Holst P., McGillicuddy-De Lisi A., Morely M., Cahalan C. Gender differences in advanced mathematical problem solving, *Journal of Experimental Child Psychology 75*, 3 (2000), 165-190.

[13] Grigoreanu, V., Beckwith, L., Fern, X., Yang, S., Komireddy, C., Narayanan, V., Cook, C., and Burnett, M. Gender differences in end-user debugging, revistited: What the miners found. In *Proc. Visual Languages and Human-Centric Computing,* IEEE (2006), 19-26.

[14] Hartzell, K. How self-efficacy and gender issues affect software adoption and use. *Comm. of the ACM 46*, 9 (2003), 167-171.

[15] Jeffries, R. A comparison of the debugging behavior of expert and novice programmers. Paper presented at the meeting of the American Educational Research Association, 1982.

[16] Ko, A.J. and Myers, B.A. Designing the Whyline: A debugging interface for asking questions about program failures. In *Proc. CHI 2004*, ACM Press (2004), 151–158.

[17] Littman, D. C., Pinto, J., Letovsky, S., and Soloway, E. Mental models and software maintenance. In E. Soloway and S. Iyengar (Eds), Empirical Studies of Programmers. Norwood, NJ, Ablex, 1986, 80-98.

[18] Meyers-Levy, J. Gender differences in information processing: A selectivity interpretation. In P. Cafferata & A. Tybout, (Eds) *Cognitive and Affective Responses to Advertising.* Lexington, Ma, Lexington Books, 1989.

[19] Nanja, N. and Cook, C.R. An analysis of the on-line debugging process. In G. M. Olson, S. Sheppard, and E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop.* Ablex, Norwood, NJ, 1987.

[20] O'Donnell, E. and Johnson, E. N. The Effects of Auditor Gender and Task Complexity on Information Processing Efficiency. *International Journal of Auditing 5* (2001), 91-105.

[21] Panko, R. What we know about spreadsheet errors. *Journal of End User Computing 10*, 2 (1998), 15-21.

[22] Prabhakararao, S., Cook, C., Ruthruff, J., Creswick, E., Main, M., Durham, M., and Burnett, M. Strategies and behaviors of end-user programmers with interactive fault localization. In *Proc. Visual Languages and Human-Centric Computing,* IEEE (2004), 15-22.

[23] Rode, J., Toye, E., and Blackwell, A. The fuzzy felt ethnography – understanding the programming patterns of domestic appliances. In *Proc. of the 2nd International Conference on Appliance Design*, (2004), 10-22.

[24] Ruthruff, J., Phalgune, A., Beckwith, L., Burnett, M. and Cook, C. Rewarding 'good' behavior: End-user debugging and rewards. In *Proc. Visual Languages and Human-Centric Computing,* IEEE (2004), 115-122.

[25] Vessey, I. Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies 23* (1985), 459-494.

[26] Wagner, E.J. and Lieberman, H. Supporting user hypotheses in problem diagnosis on the web and elsewhere. In *Proc. of the International Conference on Intelligent User Interfaces*, ACM Press (2004), 30–37.

[27] Whitworth, J.E., Price, B.A. and Randall, C.H. Factors that affect college of business student opinion of teaching and learning. *Journal of Education for Business 77*, 5 (2002), 282-289.

[28] Zeldin, A. and Pajares. F. Against the odds: Self-efficacy beliefs of women in mathematical, scientific, and technological careers. *American Educational Research Journal 37* (2000), 215-246.