Effectiveness of End-User Debugging Software Features: Are There Gender Issues?

Laura Beckwith[1], Margaret Burnett[1], Susan Wiedenbeck[2], Curtis Cook[1], Shraddha Sorte[1], Michelle Hastings[1]

[1]Oregon State University
Corvallis, Oregon
{beckwith, burnett, cook, sortes, hastingm}@cs.orst.edu

[2]Drexel University
Philadelphia, Pennsylvania
Susan.Wiedenbeck@cis.drexel.edu

ABSTRACT
Although gender differences in a technological world are receiving significant research attention, much of the research and practice has aimed at how society and education can impact the successes and retention of female computer science professionals—but the possibility of gender issues *within software* has received almost no attention. If gender issues exist with some types of software features, it is possible that accommodating them by changing these features can increase effectiveness, but only if we know what these issues are. In this paper, we empirically investigate gender differences for end users in the context of debugging spreadsheets. Our results uncover significant gender differences in self-efficacy and feature acceptance, with females exhibiting lower self-efficacy and lower feature acceptance. The results also show that these differences can significantly reduce females' effectiveness.

Author Keywords
Gender, debugging, end-user programming, end-user software engineering, Surprise-Explain-Reward.

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

For some time, we have been working on a concept we call "end-user software engineering" [8]. The essence of the end-user software engineering concept is to tightly intertwine into end-user programming environments features that aid end users in guarding against errors in the "programs" they create (e.g., spreadsheets). As part of this work, we developed a motivational strategy called Surprise-Explain-Reward [33]. Surprise-Explain-Reward is based on the concept of curiosity, and aims to encourage end users to take advantage of useful debugging features available. Our empirical results with this strategy have been encouraging [24, 27, 33].

Still, in our empirical work, we began to observe differences in how males and females regarded the features available. While individual differences, such as experience, cognitive style, and spatial ability, are likely to vary more than differences between gender groups, research from several domains has shown gender differences that are relevant to computer usage [4, 22]. Further, Czerwinski et al.'s work [14, 28] has already shown that important gender differences can exist that can be accommodated within computer systems—*but only if we know about them*.

Investigating gender-related issues within software aiming to support end-user programmers is the focus of this paper. Although there have been gender studies meant to understand and ameliorate the low representation of females in the computing field [11, 21], there has been little emphasis on software's *design* attributes and how these design attributes affect males' and females' performance in computing tasks.

In this paper, we empirically consider gender differences of end-user programmers, focusing on gender differences in software confidence and how these differences impact end users' usage of debugging software features. The overall research questions are:

> *RQ1: Are there gender differences in self-efficacy that impact effective end-user debugging?*
>
> *RQ2: Are there gender differences in users' likelihood of acceptance of unfamiliar features in end-user programming environments?*

CHI 2005, April 2–7, 2005, Portland, Oregon, USA.

We have used two empirical vehicles for our investigation of these questions: exploratory empirical work to derive appropriate subquestions for the context of end-user debugging, followed by a summative laboratory experiment. First we present the theoretical basis and exploratory empirical work from which our experimental subquestions were derived. We then present our controlled laboratory experiment investigating these subquestions in the context of an end-user programming environment.

THEORETICAL AND EMPIRICAL BASIS

Confidence and Self-Efficacy

Gender differences regarding computer confidence have been widely studied, revealing that females (both computer science majors and end users) have lower self-confidence than males in their computer-related abilities [5, 9, 17, 19, 21, 30].

Bandura's [2, 3] self-efficacy construct helps to explain lack of self-confidence and its effects. Self-efficacy is a person's judgment about his or her ability to carry out a course of action to achieve a certain type of performance. Bandura argues that achieving a desired type of performance depends on two factors, the skills needed to carry out the task and the perception of efficacy that will allow the individuals to use their skills effectively. High self-efficacy is critical in problem solving because self-efficacy influences the use of cognitive strategies, the amount of effort put forth, the level of persistence, the coping strategies adopted in the face of obstacles, and the final performance outcome.

In end users' computer usage, Busch found that at the end of a year-long computer applications course, female students had lower self-efficacy than male students [9]. Females lacked confidence particularly in their ability to succeed at complex tasks in word processing and spreadsheets. Similarly, Torkzadeh and Koufteros [29] found that females in a business computer applications course had lower self-efficacy than males on computer file and software management activities. Other research has shown that low self-efficacy affects females' perceptions of a software application before actual use [18], raising the possibility that females with low self-efficacy may avoid using it altogether.

This research led to the following subquestions that our experiment sought to answer: *Are there gender differences in self-efficacy in the domain of end-user debugging?* And *is self-efficacy tied to effectiveness in end-user debugging?*

Potential Impacts on Feature Acceptance

Our next two research subquestions are regarding feature acceptance and were derived as follows. First, through self-efficacy literature and a short survey of our own we consider how confidence and perceived risk might be tied to feature acceptance. We then add preliminary empirical explorations in the domain of end-user debugging to derive these final two research subquestions.

The studies by Busch and by Torkzadeh and Koufteros were done a number of years ago, and software has changed significantly since then. Thus, in part to confirm this phenomenon in 2004-era software, and in part to consider potential ties with feature acceptance, we ran a small survey. Our survey looked for links between respondents' software confidence and their self-reported willingness to explore new features in their real-world computer usage, with questions such as "I avoid working with new software since it requires more time to learn," "If something goes wrong with the software (like the program crashes), I believe I can fix it," and "I enjoy exploring new features provided with the software." Questions were answered on either a five-point Likert scale or a ranking of choices. The respondents were 21 psychology and business majors: 14 females and 7 males. Our survey results were extremely consistent with the above findings: in *all ten* of our questions about software confidence and respondents' acceptance of new or advanced software features, females' mean scores were lower than the males'. (In fact, even with this small sample size, many of these differences were statistically significant.)

There are at least two levels of acceptance of unfamiliar features in software: the initial willingness to try a feature, which we refer to as "willingness to approach," and repeated genuine usage of a feature, which we refer to as "willingness to adopt." Research on technology acceptance in organizational settings has shown that users' early perceptions of software have a long-term effect on both their intention to use a technology and their actual usage of it [13, 15]. Two key perceptions that influence both initial and longer-term acceptance are the software's perceived usefulness and its perceived ease of use. Research on gender differences in technology acceptance [31] suggests that males are most strongly influenced by their perception of usefulness, both initially and over a period of use. Females are initially more influenced by their perceptions of ease of use and social norms (defined in [31] as the opinions of others), but over time the role of social norms declines markedly and ease of use and usefulness become their main criteria for acceptance and actual use.

Gender differences in perceptions of risk are also likely to have close ties with an individual's probability of acceptance. Blackwell's Attention Investment Model [6] explains the role of perception of risk in problem solving. According to the model, a user weighs four factors before taking an action: perceived benefits, expected pay-off, perceived cost, and perceived risks. Given a discretionary situation in which the user has choices of alternative courses of action, high perceived risk and cost of an action that are not offset by the perceived benefits and pay-offs are likely to result in a decision not to pursue the action.

Research has shown that females perceive higher risk than males in both exceptional situations and everyday activities (e.g., explosion in a nuclear power plant vs. driving a car) [16]. Other research on risk and gender has documented

that females are more risk-averse in their financial decisions [20]. A meta-analysis of 150 studies on gender and risk taking also found that females engaged in less risk taking than males [10]. The meta-analysis did not address risks of computer use directly. However, it did find that intellectual risk taking, defined as activities involving mathematical or spatial reasoning skills, was greater in males than in females [10]. This is of interest to our work because the possible risks our participants faced in completing the task were intellectual rather than physical or financial.

In our own recent empirical work on end-user debugging [24, 26, 27], in which we were investigating questions unrelated to gender, we collected the gender of every participant for later exploration. We were then able to "mine" these data for gender differences. In addition, we conducted a think-aloud study to help derive research subquestions [4]. The purpose of these explorations was to gain insights into whether investigation of gender differences in end-user debugging would be warranted.

These preliminary explorations did point to the need for further investigation. In fact, we found numerous gender differences relating to measures of software confidence and feature acceptance. For example, in our think-aloud study the females' confidence was not only lower, it tended to drop over the course of the study much more than the males' confidence levels did. Further, the mining of our recent experimental studies showed pronounced differences between males and females in amount and type of activities in which they engaged. (Figure 1 shows the profiles for one male and one female whose activity patterns were fairly representative of their genders.) Also, males were interested readers of explanations describing new features early in the experiment, but not so later, whereas females delayed most of their reading until much later (Figure 2).

Drawing from the previous research cited in this subsection and the implications of our own preliminary results, we chose to design the experiment so that it could answer the following two additional research subquestions: *Are there gender differences in end users' willingness to approach new debugging features?* And *are there gender differences in their willingness to then adopt these new features?*

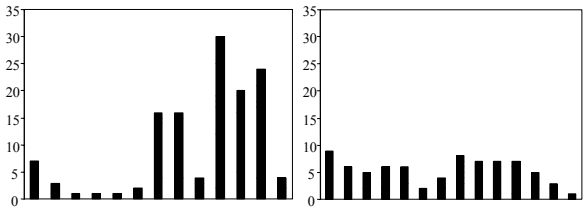

Figure 1. Activity profiles of one male (left) and one female (right). The male did more actions and used more features than the female. The horizontal positions represent points in time during the experimental task. Height represents frequency of debugging feature usage.

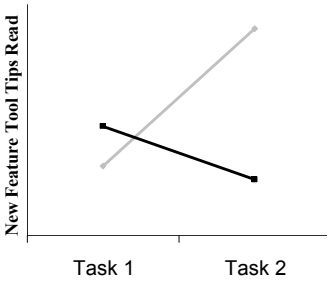

Figure 2. Males' (dark line) mean interest in tool-tip accessible explanations describing new features started higher than females' (light line) and then declined, whereas females' interest increased.

EXPERIMENT

The experimental design is presented here. Note that the experiment's procedures, materials, tutorial, questionnaires, transcripting software, and tasks have been used in or derived from previous studies ([24, 26, 27, 33], except where other sources are stated below). In addition, they have been "tested" using analytical (cognitive walkthroughs) and empirical (pilot participants) methods.

Participants and Procedures

The 27 male and 24 female participants (mostly business students) started by filling out a pre-session questionnaire which collected participant background data and included the self-efficacy questions based on a slightly modified version of Compeau and Higgins' validated scale [12]; the modifications made the questionnaire task-specific to end-user debugging. Participants were asked to answer on a five-point Likert scale their level of agreement with the statements. For example, "I could find and fix errors… if there was no one around to tell me what to do as I go," "…if I had seen someone else using it before trying it myself," and "…if I had a lot of time to complete the task."

The following background data were collected: gender, major, year or degree completed, GPA, programming experience (to bar participants with more programming experience than is usual for business students), spreadsheet experience, previous use of the study's prototype environment, and whether English was their primary language. We did not collect data on other factors that might seem relevant, such as mathematical ability, because the population of interest was spreadsheet users (other than trained programmers) at a roughly equivalent point in their academic careers, regardless of any other talents they may have. Statistical analysis of the background data showed that the females were academically a little younger than the males[1] (ANOVA $F(1,48)=4.528$, $p<.039$). There were no

[1] Post hoc analysis using the non-parametric Kruskel-Wallis test showed that the difference in academic age was not predictive of any of the outcome measures (performance measures or behavior patterns) in this study.

significant differences between the genders in any other background data collected.

All participants received the same treatment; the only independent variable was gender. The study took place over eight sessions conducted in February and July of 2004. The two times do not threaten validity because the February and July males were analyzed as a single group, which was compared against the February and July females as a single group. Each participant attended one session. The participants were seated one per computer in a small lab. After participants completed the questionnaire we administered a 35-minute "hands-on" tutorial to familiarize participants with the environment. The participants were then given two spreadsheet debugging tasks. We captured their actions in electronic transcripts, as well as their final spreadsheets. At the conclusion of each task, we administered post-task questionnaires in which participants self-rated their performance on the task. The second task's post-session questionnaire also included questions assessing participants' comprehension of features in the environment.

Environment

The debugging features that were present in this experiment were part 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 [8]. 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 underlying assumption behind WYSIWYT is that, as a user incrementally develops a spreadsheet, he or she can also be testing incrementally. Figure 3 shows an example

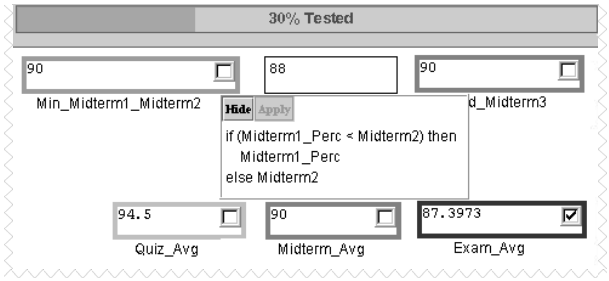

Figure 3. An example of WYSIWYT in Forms/3.

of WYSIWYT in Forms/3 [7], the research spreadsheet environment used in this experiment. In WYSIWYT, untested cells have red borders (light gray in this paper). 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: this communicates a successful test. Behind the scenes, checkmarks increase the "testedness" of a cell according to a test adequacy criterion based on formula expression coverage (described in [25]), and this is depicted by the cell's border becoming more blue (more black in this paper). Also visible in the figure, the progress bar (top) reflects the testedness of the entire spreadsheet.

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 put an X-mark in the cell's decision box. X-marks trigger fault likelihood calculations, which cause cells suspected of containing faults to be colored in shades along a yellow-orange continuum (shades of gray in this paper), with darker orange shades given to cells with increased fault likelihood. Figure 4 shows an example of this behavior in one of the

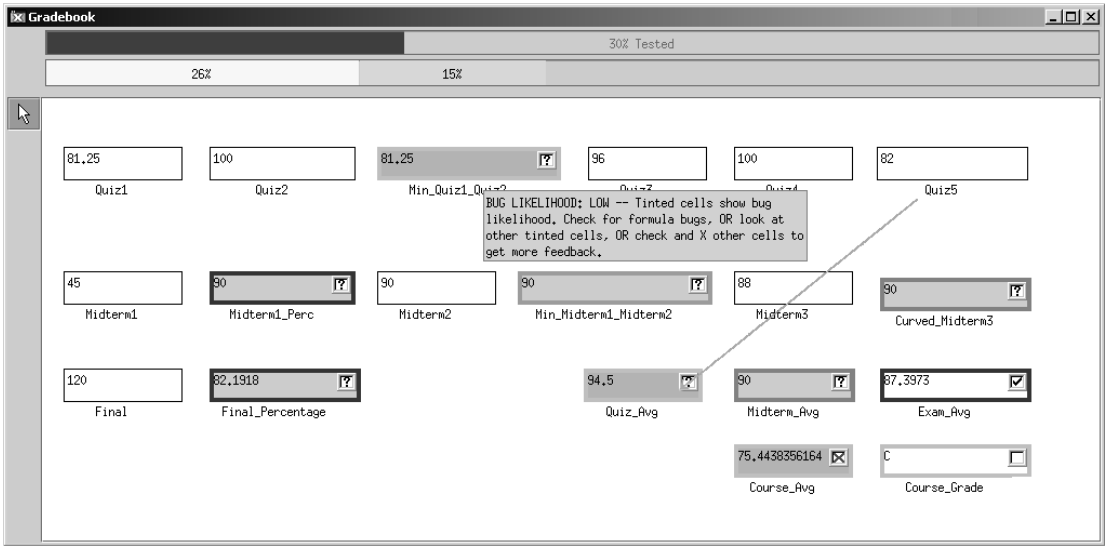

Figure 4. The user notices an incorrect value in `Course_Avg`—the value is obviously too low—and places an X-mark in the cell. As a result of this X and the checkmark in `Exam_Avg`, eight cells are identified as being possible sources of the incorrect value, with some deemed more likely than others.

spreadsheets the participants debugged. The intent is to lead the user to the faulty cell (colored darkest orange).

The optional dataflow arrows are colored to reflect testedness of specific relationships between cells and subexpressions. (The user can turn these arrows on/off at will.) In Figure 4, the user has popped up Quiz5's arrow, which shows both that Quiz5 is referenced in Quiz_Avg's formula and that this relationship is not yet tested.

The way these features are supported is via the Surprise-Explain-Reward strategy [24, 27, 33]. If a user is surprised by or becomes curious about any of the feedback of the debugging features, such as cell border color or interior cell coloring, he or she can seek an explanation, available via tool tips (Figure 4). The aim of the strategy is that, if the user follows up as advised in the explanation, rewards will ensue [27]. Some of the potential rewards are functional—such as being led directly to a bug—and some are affective—such as increased progress in the progress bar. One aspect of interest in this experiment was whether, if gender differences in confidence were present, they might impact Surprise-Explain-Reward's success in encouraging users to approach and adopt new features.

Tutorial

In the tutorial, participants performed actions on their own machines with guidance at each step. The tutorial did some teaching of the checkmark feature (including its associated testedness-colored arrows feature), but did not include any debugging or testing strategy instruction. The tutorial did no teaching of the X-mark feature. Instead, participants were simply shown that it was possible to place X-marks and given time to figure out any aspects of the feedback that they found interesting. This design allowed us to gather information on three types of "newness" of software features: one type corresponding to the traditional way of thinking about formula errors (namely, formula editing), another type not previously encountered but explicitly taught (checkmarks and arrows), and a third type completely untaught (X-marks).

Half of the tutorial sessions were presented by a male graduate student and half were presented by a female graduate student. This design ensured that approximately 50% of males were instructed by a same-gender instructor and 50% by an opposite-gender instructor (and likewise for the females) [32], serving to distribute any gender effect of the tutorial presenter equally over the two genders.

Tasks

The experiment consisted of two spreadsheets, `Gradebook` and `Payroll` (Figure 4 and Figure 5). To make the spreadsheets representative of real end-user spreadsheets, `Gradebook` was derived from an Excel spreadsheet of an (end-user) instructor, which we ported into an equivalent Forms/3 spreadsheet. `Payroll` was a spreadsheet designed by two Forms/3 researchers using a payroll description from a real company.

These spreadsheets were each seeded with five faults created by real end users. To obtain these faults, we provided three end users 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, we chose five that provided coverage of the categories in Panko's classification system [23] (based upon Allwood's classification system [1]). 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 [23]. We seeded `Gradebook` 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. `Payroll` was intended to be the more difficult task due to its larger size, greater length of dataflow chains, intertwined dataflow relationships, and more difficult faults.

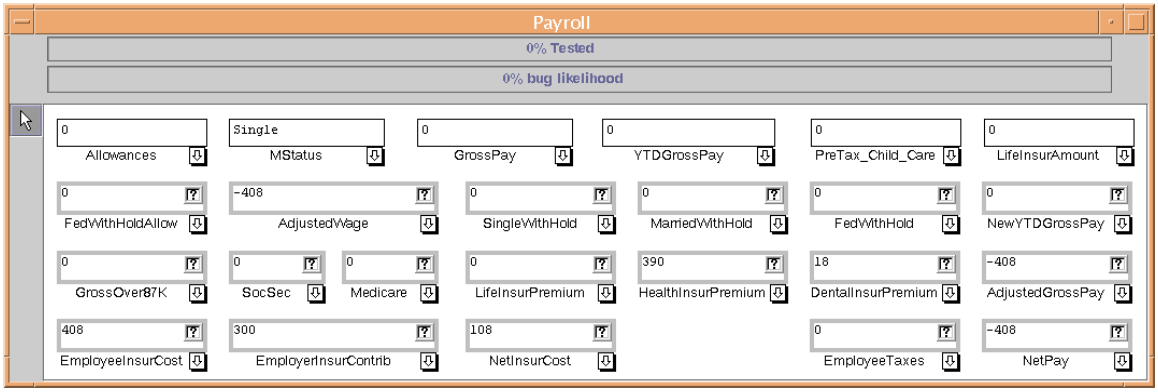

Figure 5. The Payroll spreadsheet.

The participants were provided these `Gradebook` and `Payroll` spreadsheets and descriptions, with time limits of 22 and 35 minutes, respectively. There are two reasons for the time limits. One is related to external validity: computing tasks in the real world are governed by time constraints, and time limits provide some simulation of this fact. The other is related to internal validity: removing the time limits would have introduced possibilities that participants would spend so much time on the first task they would be unwilling to spend time on the second, would leave as soon as the participation fee was collectible, and so on, which would introduce confounds into the data. The use of two spreadsheets reduced the chances of the results being due to any one spreadsheet's particular characteristics. The experiment was counterbalanced with respect to task order so as to distribute learning effects evenly. The participants were instructed, "Test the … spreadsheet to see if it works correctly and correct any errors you find."

Measures

The independent variable in this study was gender. The dependent measures were self-efficacy as measured by a Likert-scale pre-session questionnaire, overall percent testedness, seeded bugs fixed, new bugs introduced, time to first use of features, feature usage, score on a post-session questionnaire's comprehension questions, and opinions given on a Likert-scale post-session questionnaire. Details of these measures are provided with the results to which they apply.

RQ1 RESULTS: GENDER DIFFERENCES IN SELF-EFFICACY AND EFFECTIVE DEBUGGING

Gender Differences in Self-Efficacy

As discussed earlier, gender differences in computer self-efficacy have been found in several computing situations. Our analysis of the pre-session self-efficacy questionnaire revealed that these differences were also present for debugging: females had significantly lower self-efficacy than the males (Mann Whitney: U=181, tied $p<0.018$). See Figure 6 and Table 1. Cronbach's alpha for the ten-item questionnaire was .879 on 49 cases, indicating high reliability. Self-efficacy literature suggests that high self-efficacy is critical for problem solving [2, 3], which predicts that for our results, high self-efficacy will be tied with high debugging effectiveness, a point we will return to shortly.

The self-efficacy literature further suggests that previous experience is one of the factors determining self-efficacy. To consider whether this held in our domain, we examined the participants' previous spreadsheet experience as a predictor of self-efficacy. The relationships for the whole group—and for females—were significant (linear regression: all: $F(1,45)=8.721$, $\beta=.403$, $R^2=.162$, $p<.005$; males: $F(1,23)=4.002$, $\beta=.385$, $R^2=.148$, $p<.057$; females: $F(1,20)=5.751$, $\beta=.473$, $R^2=.223$, $p<.026$). This relationship raises the possibility that, at least for females, low self-efficacy may be addressable by finding ways to increase their experience level.

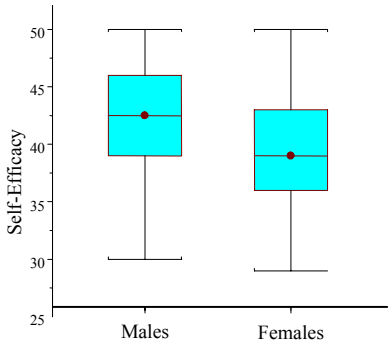

Figure 6. Males' and females' pre-session self-efficacy. (Maximum possible self-efficacy was 50.) The center line of each box represents the median self-efficacy score. The boxes show the ranges encompassed by 50% of the scores of each gender. The whiskers extending above and below the boxes show the remaining upper and lower 25% of the scores.

Gender	Self-Efficacy	Final Percent Testedness
Males	42.27 (4.69) n=26	62.85 (21.36) n=27
Females	38.96 (5.11) n=23	54.79 (25.04) n=24

Table 1. Mean (standard deviation) and number of participants[1] for males' and females' self-efficacy and final percent testedness.

Ties to Effectiveness

We first considered the relationship between self-efficacy and effective usage of WYSIWYT debugging features. We chose final percent testedness (refer back to the progress bar in Figure 3) as our measure of effective usage for two reasons. First, percent testedness can only increase through strategically checking off input/output value combinations. (Recall, input values must be chosen that actually add testing coverage of formula expressions.) Second, final percent testedness in previous experiments has been significantly tied with success in debugging [8].

As Figure 7 shows, females' self-efficacy was indeed a significant predictor of their final percent testedness. For the males, however, self-efficacy was not a predictor of their effective usage of the debugging features (linear regression: males: $F(1,25)=.365$, $\beta=-.569$, $R^2=.015$, $p<.551$; females: $F(1,22)=4.52$, $\beta=2.09$, $R^2=.177$, $p<.046$). From

[1] Note that the number of participants does not sum to 51. Some participants did not complete the questionnaire. Incomplete questionnaires were also the reason for other sample sizes in this paper that do not sum to 51.

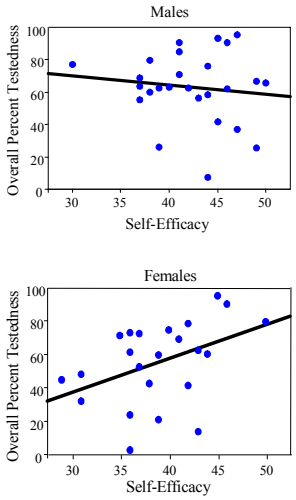

Figure 7. Self-efficacy as a predictor of final spreadsheet testedness. The regression lines show the females' positive relationship of self-efficacy to spreadsheet testedness and also suggest greater variability among females. The means are given in Table 1.

this we can conclude that self-efficacy had important implications for females' problem-solving choices.

These choices of how much to use WYSIWYT testing features mattered: as in our previous studies, effective usage of the testing features (as measured through percent testedness) was predictive of the number of bugs fixed. The results of linear regression analysis of percent testedness on the number of bugs fixed were significant over all participants and also were significant for each gender (linear regression: all: $F(1,49)=21.701$, $\beta=.554$, $R^2=.307$, $p<.0001$; males: $F(1,25)=16.60$, $\beta=.632$, $R^2=.399$, $p<.0004$; females: $F(1,22)=6.818$, $\beta=.486$, $R^2=.237$, $p<.016$).

Finally, we considered the "bottom line" via two measures of debugging effectiveness: bugs fixed, and new bugs introduced. Recall that we had seeded each spreadsheet with five bugs. For purposes of this paper, we count as "bugs fixed" those seeded bugs that were no longer present by the end of the task. "Bugs introduced" are bugs that were not seeded, but were present at the end of the task.

Although there was no significant difference between the females' and males' performance in fixing the seeded bugs (Mann Whitney: U=300.5, p<0.651), the females introduced significantly more bugs than the males did (Mann Whitney: U=227.5, p<0.011). See Table 2. The gender difference in bugs introduced is confirmed by a gender difference in participants introducing the bugs: 9 of the 24 females introduced bugs, which is significantly greater than the 2 males (out of 27 total) who introduced bugs (Fishers Exact Test: p<.015). Note that these new bugs were never fixed.

Gender	Seeded Bugs Fixed (10 possible)	New Bugs Introduced
Males (n=27)	5.815 (2.167)	.111 (.424)
Females (n=24)	5.667 (2.014)	.583 (.974)

Table 2. Mean (standard deviation) performance of males and females on bugs fixed and new bugs introduced that still remained at the end of the task.

RQ2 RESULTS: GENDER DIFFERENCES IN ACCEPTANCE OF UNFAMILIAR FEATURES

Was females' lower self-efficacy tied to lower acceptance of the debugging features that might have helped their effectiveness? As mentioned earlier, participants had access to three types of features: (1) the ability to edit formulas, which is a feature common to all spreadsheet environments, (2) the WYSIWYT features we taught in the tutorial (checkmarks and arrows), and (3) the fault localization (X-mark) feature, which was not taught at all. We will label these three types of features as Type Familiar, Type Taught, and Type Untaught, respectively. We use these types to consider two forms of feature acceptance: willingness to initially approach a feature, and then willingness to adopt it (i.e., commit to repeated genuine usage during debugging).

Willingness to Approach New Features

Females were inclined to approach the Type Familiar feature earliest, using it significantly earlier than the males did (ANOVA: $F(1,49)=5.33$, $p<.025$). In contrast to this, males were much earlier to approach the new features (Type Taught and Type Untaught): the gender difference was significant for Type Taught features (ANOVA: Taught: $F(1,49)=8.694$, $p<.005$; Untaught: $F(1,40)=3.40$, $p<.073$). Figure 8 shows the mean time of first usage for each of these feature types.

Willingness to Adopt New Features

Our criterion of adoption was repeated *genuine* usage. Measuring genuineness required somewhat different

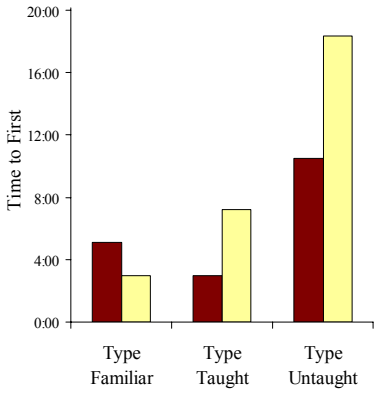

Figure 8. Males (dark bars) first used new (taught and untaught) features much earlier than females (light bars).

measures for each feature type. For the Type Familiar feature (formula edits), we simply used frequency of edits. This was a reasonable measure of genuine usage because editing a formula requires intellectual investment and pertains directly to debugging. However, for Type Taught features (checkmarks and arrows), the intellectual cost of usage was low, a single click. Furthermore, the effects on debugging are only indirect, because after a checkmark a formula edit was not necessarily expected (since placing a checkmark indicated belief that a cell's value was correct). For these features, it was not possible to determine presence of intellectual involvement, but there were patterns for which its absence could be inferred. We thus omitted Type Taught actions toggled again and again on the same cell by the participants *after* they had stopped editing formulas. After filtering these out, we then used frequency of the Type Taught actions as our measure.

For the Type Untaught feature (X-marks), intellectual cost was low, but there was a detectable route from genuine usage of the feature to debugging: following the advice of an X-mark's feedback leads eventually to formula edits on a colored cell. Thus, for Type Untaught features, a participant was counted as adopting X-marks if he or she placed more than one X-mark in at least one task, and then eventually followed up by editing a colored cell's formula. Since only about 60% of the participants exhibited this behavior and their frequency of usage according to this definition was necessarily low (1 or 2 was typical), counting participants rather than frequency was the right measure for Type Untaught feature adoption.

By these measures, the only type of feature for which females had a higher adoption rate was the Type Familiar feature of formula editing (ANOVA: $F(1,49)=4.979$, $p<.03$). See Table 3. Males, however, were more willing to adopt the new features: they performed significantly more Type Taught actions than females, as Table 4 shows (ANOVA: $F(1,49)=4.971$, $p<.03$). Furthermore, significantly more males used Type Untaught features than females did, as Table 5 shows (Fisher's Exact Test: $p<.01$).

Gender	Type Familiar Features
Males (n=27)	23.8 (9.58)
Females (n=24)	29.8 (9.66)

Table 3. Mean (standard deviation) number of Type Familiar features.

Gender	Type Taught Features
Males (n=27)	123.41 (68.27)
Females (n=24)	87.54 (47.67)

Table 4. Mean (standard deviation) number of actions associated with Type Taught features.

Gender	Adopted	Did Not Adopt
Males (n=27)	22	5
Females (n=24)	11	13

Table 5. Number of participants who adopted Type Untaught features.

The gender difference in adoption of the Type Untaught feature may be partially explained by the answers (on a five-point Likert scale) to a statement included on the post-task questionnaire. The statement said: "... I was afraid I would take too long to learn [X-marks]." Females agreed with this statement significantly more than the males (Mann Whitney: $U=157$, $p<.017$).

However, despite the gender differences in *expectation* of their ability to learn the Type Untaught feature, there were no gender differences in *actual* learning of the feature—even though the males were able to practice it more through their greater adoption of it. In the post-task questionnaire, participants answered nine prediction and interpretation questions related to the Type Untaught feature. Males answered 60% of these questions correctly, and females answered 53% correctly (ANOVA: $F(1,49)=.929$, $p<.34$). This seems to be a case of *inappropriately* low self-efficacy of the females inhibiting their use of this feature.

DISCUSSION

The results of this study establish ties from the well known gender differences in computer-related confidence to end users' debugging behaviors. The females, whose self-efficacy was significantly lower than the males, were less willing to accept the new debugging features in the software environment—which is unfortunate, because these features, which explicitly support testing and debugging, were statistically significant predictors of debugging success.[1]

Females' low self-efficacy may be related to perceptions of risk, exacerbating the problem. Studies have documented females' high perception of risk in intellectual activities involving mathematical or spatial reasoning skills [10]. Applying this to our study, an individual with low beliefs in her ability to succeed at debugging may hesitate to use new debugging features because of the risk they may not pay off in better debugging performance. Further, she may believe that her cost of learning them will be high, due to her low opinion of her own capabilities. As predicted by the Attention Investment Model [6] and borne out by the females' questionnaire responses and actions performed in our study, she may decide to forego the new features and use the debugging feature she already knows, formula editing.

[1] These results may also apply to other low-self-efficacy groups, such as economic, geographic, and ethnic groups with underrepresented access to technology.

Were the females' low self-efficacy predictions a case of realism, or of self-fulfilling prophecy? Females' perceptions of their inability to learn new features were not borne out by their actual learning of these features. This evidence suggests that females' low self-efficacy were a self-fulfilling prophecy: their low expectations about their ability to learn new features prevented them from achieving the benefits the new features might have brought them.

In the present study, females spent the time they "gained" through foregoing the new features by editing more formulas. This resulted in significantly more introduced bugs, perhaps because, without the new features, they had less ammunition to use in tracking down these bugs. As several previous studies have shown, users *do* benefit in effectiveness from the debugging features [8]. However, the data presented in this paper indicate that the degree of benefit is not equal for males and females. This is a troubling result.

Our data also indicate that previous experience with spreadsheets has an important influence on self-efficacy. According to Bandura [2, 3], the most important way of increasing self-efficacy is direct performance experiences. Lower self-efficacy of females for spreadsheet debugging may be remediated by greater experience. Thus, as a female gets more experience, including experience with end-user debugging features, her self-efficacy can be expected to rise, with corresponding increases in effective usage of features that increase performance.

However, there is a circular dependency here—a female may never gain the experience needed to raise her self-efficacy and performance capabilities if she has already concluded that it is too risky or costly due to her perceived capabilities being too low. In this situation time itself is not enough to produce the needed experience to raise self-efficacy. Consequently, looking to other, more aggressive, methods seems warranted.

The relationship between experience and willingness to use new features suggests that a good design strategy may be to focus on how to initially attract females to try the features, thereby increasing their experience level. There are prior research results showing that the Surprise-Explain-Reward strategy effectively draws many users to new features [24, 33], and this strategy provides a possible base for attracting females to relevant new features. However, in the underlying data there are indications of gender differences in some of the interruption-based Surprise-Explain-Reward devices. Our findings in the current study lend support to these indications. Further research into interactions between gender and interruption style in the domain of complex problem-solving tasks such as debugging may provide useful keys to how best to attract females to trying new features.

Females' perception that learning the new features would take them too long also suggests that a partial solution may lie in the *content* of communication that helps users to assess both the worth and risks of using the features. Such communication may need to convince users not only of the features' ease of use, but also of the accuracy risks they are taking by *not* using the features.

CONCLUSION

To date research regarding computer-related gender differences has not considered how the design of *software* interacts with gender differences. Our investigation of this issue was performed in the context of end-user debugging. The main results were:

- Females had lower self-efficacy than males did about their abilities to debug. Further, females' self-efficacy was predictive of their effectiveness at using the debugging features (which was not the case for the males).
- Females were less likely than males were to accept the new debugging features. One reason females stated for this was that they thought the features would take them too long to learn. Yet, there was no real difference in the males' and females' ability to learn the new features.
- Although there was no gender difference in fixing the seeded bugs, females *introduced* more new bugs—which remained unfixed. This is probably explained by low acceptance of the debugging features: high effective usage was a significant predictor of ability to fix bugs.

We believe these findings have implications far beyond debugging. They suggest to designers of software products for end users that, unless appropriate accommodations can be made, there are likely to be important gender differences in the users' willingness to accept new features that can benefit them.

ACKNOWLEDGMENTS

We 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. Michelle Hastings was supported by Saturday Academy's Apprenticeships in Science and Engineering Program.

REFERENCES

1. Allwood, C. Error detection processes in statistical problem solving. *Cognitive Science 8*, 4 (1984), 413-437.
2. Bandura, A. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review 8*, 2 (1977), 191-215.
3. Bandura, A. *Social Foundations of Thought and Action*. Prentice Hall, Englewood Cliffs NJ, 1986.
4. Beckwith, L. and Burnett M. Gender: An important factor in end-user programming environments? In *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing* (2004), 107-114.

5. Beyer, S., DeKeuster, M., Rynes, K., Kostman, A., and DeGregorio, N. Barriers to women's success in Management Information Systems courses. In *Proc. American Psychological Society* (2004).
6. Blackwell, A. First steps in programming: a rationale for attention investment models. In *Proc. IEEE Human-Centric Computing Languages and Environments* (2002), 2-10.
7. Burnett, M., Atwood, J., Djang, R., Gottfried, H., Reichwein, J. and Yang, S. Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming 11*, 2 (2001), 155-206.
8. Burnett, M., Cook, C. and Rothermel G. End-user software engineering. *Communications 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*, 2 (1995), 147-158.
10. Byrnes, J. P., Miller, D. C. and Schafer W. D. Gender differences in risk taking: A meta-analysis. *Psychological Bulletin 125*, (1999), 367-383.
11. Camp, T. The incredible shrinking pipeline. *Communications of the ACM 40*, 10 (1997), 103-110.
12. Compeau, D. and Higgins, C. Computer self-efficacy: development of a measure and initial test. *MIS Quarterly 19*, 2 (1995), 189-211.
13. Compeau, D., Higgins, C. A. and Huff, S. Social cognitive theory and individual reactions to computing technology: a longitudinal study. *MIS Quarterly 23*, 2 (1999), 145-158.
14. Czerwinski, M., Tan, D. S. and Robertson, G. G. Women take a wider view. In *Proc. CHI 2002*, ACM Press (2002), 195-202.
15. Davis, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly 13*, 3 (1989), 319-340.
16. Finucane, M., Slovic, P., Merz., C-K., Flynn, J. and Satterfield, T. Gender, race and perceived risk: the white male effect. *Health, Risk and Society 2*, 2 (2000), 159-172.
17. Fisher, A., Margolis, J. and Miller, F. Undergraduate women in computer science: Experience, motivation, and culture. In *Proc. SIGCSE Technical Symposium on Computer Science Education*, ACM Press (1997), 106-110.
18. Hartzel, K. How self-efficacy and gender issues affect software adoption and use. *Communications of the ACM 46*, 9 (2003), 167-171.
19. Huff, C. Gender, software design, and occupational equity. *ACM SIGCSE Bulletin 34*, 2 (2002), 112-115.
20. Jiankoplos, N. A. and Bernasek, A. Are women more risk averse? *Economic Inquiry 36*, 4 (1998), 620-630.
21. Margolis, J., Fisher, A., Miller, F., Caring about connections: Gender and computing, *IEEE Technology and Society Magazine 18*, 4 (1999), 13-20.
22. Nass, C., Moon, Y., and Green, C. Are machines gender-neutral? Gender-stereotypic responses to computers with voices. *Journal of Applied Social Psychology 27*, (1997), 864-876
23. Panko, R. What we know about spreadsheet errors. *Journal of End User Computing 10*, 2 (1998), 15-21.
24. Robertson, T. J., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, J., Beckwith, L. and Phalgune, A. Impact of interruption style on end-user debugging. In *Proc. CHI 2004*, ACM Press (2004), 287-294.
25. Rothermel G., Burnett M., Li L., Dupuis, C. and Sheretov, A. A methodology for testing spreadsheets, *ACM Transactions on Software Engineering and Methodology 10*, 1 (2001), 110-147.
26. Ruthruff, J., Burnett, M. and Rothermel, G. An empirical study of fault localization for end-user programmers, In *Proc. International Conference on Software Engineering* (2005), to appear.
27. Ruthruff, J., Phalgune, A., Beckwith, L., Burnett, M. and Cook, C. Rewarding 'good' behavior: End-user debugging and rewards. In *Proc. IEEE Visual Languages and Human-Centric Computing* (2004), 115-122.
28. Tan D. S., Czerwinski, M., and Robertson, G. G. Women go with the (optical) flow. In *Proc. CHI 2003*, ACM Press (2003), 209-215.
29. Torkzadeh, G. and Koufteros, X. Factorial validity of a computer self-efficacy scale and the impact of computer training. *Educational and Psychological Measurement 54*, 3 (1994), 813-821.
30. Torkzadeh, G. and Van Dyke, T. Effects of training on internet self-efficacy and computer user attitudes. *Computers in Human Behavior 18*, (2002), 479-494.
31. Venkatesh, V. and Morris M. G. Why don't men ever stop to ask for directions? Gender, social influence, and their role in technology acceptance and usage behavior. *MIS Quarterly 24*, 1 (2000), 115-139.
32. 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.
33. Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., Durham, M. and Rothermel, G. Harnessing curiosity to increase correctness in end-user programming. In *Proc. CHI 2003*, ACM Press (2003), 305–312.