On to the Real World: Gender and Self-Efficacy in Excel

Laura Beckwith, Derek Inman, Kyle Rector, and Margaret Burnett
Oregon State University
Corvallis, OR USA
{beckwith, inmand, rectorky, burnett}@eecs.orgonstate.edu

Abstract

Although there have been a number of studies of end-user software development tasks, few of them have considered gender issues for real end-user developers in real-world environments for end-user programming. In order to be trusted, the results of such laboratory studies must always be re-evaluated with fewer controls, more closely reflecting real-world conditions. Therefore, the research question in this paper is whether the results of a Gender HCI controlled study generalize -- to real-world end-user developers, in a real-world spreadsheet environment, using a real-world spreadsheet. Our findings are that the concepts revealed by the original laboratory study appear to be quite robust, being demonstrated in multiple ways in this real-world environment.

1. Introduction

In the field of visual and human centric computing languages, there have been many studies using academic prototypes, populations, and tasks. These studies often feature careful controls to limit the number of variables, and thus can achieve clean and clear results. However, too often researchers never take the next step to explore the generalizability of their findings on real-world (widely used and commercially available) products. As a result, their findings cannot be trusted beyond the original, very limited setting.

This paper takes the next step following up on the results of one of our earlier studies involving an academic prototype. The original study examined the effects of *self-efficacy* (a form of self-confidence) and gender on users' problem solving behaviors with the academic prototype spreadsheet environment Forms/3 including its WYSIWYT debugging features [5]. The context was the end-user software engineering task of debugging. The results found significant gender differences in the ways males and females problem solved in the environment.

The purpose of the follow-up study was to explore how the original findings generalize. The term *generalize* is defined as "to give general applicability to" [14]. In order to generalize the results from our initial experiment we made the following changes:

- Different environment: Excel with unlimited access to features.
- Different population: Seattle-area real-world users of Excel.
- Different task: Spreadsheet modification, with emphasis on reliability of changes.

Another goal in the follow-up experiment design is replication, to ascertain whether the same research results will occur if an experiment is replicated. (The term *replicate* is defined as "performance of an experiment or procedure more than once" [14].) There is a delicate balance between generalizing and replicating. If there are too many changes, the new study is no longer replicating the original; if there are too few changes, hardly any generalization can occur. Thus, we replicated the initial experiment procedures to the extent possible given our generalization goals. The factors we replicated were:

- Task domain: End-user software engineering.
- Tutorial: Same style of teaching and introducing features to aid task.
- Design/Procedures: The design was the same, and the procedures were as similar as possible given the new setting.
- Research questions: the same research questions.

Our goal was to discover whether the new experiment in this new setting would again reveal significant gender differences in how males and females problem-solve.

2. Background: Gender-Debugging Study

The study we set out to generalize (referred to as the *gender-debugging study* in this paper) investigated the following two research questions [2]:

RQ1: Are there gender differences in self-efficacy that impact effective end-user programming?

RQ2: Are there gender differences in users' likelihood of acceptance of unfamiliar features in end-user programming environments?

The general format of the study was to give male and female participants (mainly an end-user business student population) two spreadsheets to debug, after being familiarized with the environment. Figure 1 shows several features users had access to for aiding their debugging task. The results were as follows:

- Females had lower self-efficacy (a form of confidence) 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 to accept the new debugging features. A 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 appears to be explained by their low acceptance of the debugging features: high effective usage was a significant predictor of ability to fix bugs.

3. Related Work

Self-efficacy [1] is a person's judgment about his or her ability to carry out a course of action to achieve a certain type of performance. 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.

Busch was one of the first to report gender differences in computer-related self-efficacy which he discovered in teaching a year long course of various computer applications. At the end of the course, the females had significantly lower self-efficacy on completing complex spreadsheet tasks than the males [6].

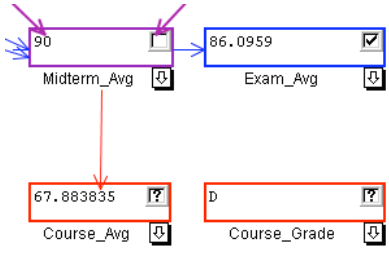

Figure 1. The experimental environment (Forms/3 [5]) used in the gender-debugging study. The features included allowing users to check-off correct values x-out incorrect values (not shown). These actions caused cells' borders and colors to change, reflecting the system's reasoning about "testedness" and likelihood of formula errors. Users could also see dataflow relationship between cells using arrows.

Even though females have often been found to have lower computer-related self-efficacy than males [2, 6, 8, 9], few studies have considered *how* self-efficacy affects actual computer usage. Of the ones that have, most relied on self-reported usage [11, 13, 17]. In each of these studies, the males' and females' self-reported usage of the computer technologies was no different, but males always reported higher beliefs in their abilities than the females.

Recently, other researchers have begun to study males and females doing programming activities, looking specifically at gender. Kelleher et al.'s work on gender and programming environments [12] has focused on middle school girls, and the types of environments that encourage computer programming. They found that girls become more engaged in programming and enjoy it more when the programming environment is designed for story-telling [12].

A few interview-style studies of end-user programmers in their "real lives" have also considered gender, for example with end-user web developers and programming in the home [15, 16]. Rode et al.'s research on home programming found different categories of appliances that were more likely to be programmed by males (e.g. entertainment devices) and by females (e.g. kitchen appliances). Their study involved both a real-world environment (of various home appliances) and real-world users. Our study uses a real-world environment and real-world population, but the set-up is lab- and task-based, not interview-based.

4. Study Design

As mentioned in the Introduction, we replicated the gender-debugging study's procedures to the extent possible [2].

4.1 Participants

We recruited participants from Microsoft's repository of Seattle-area residents interested in being part of a study for compensation of a Microsoft product. In order to be eligible to participate in this study, each participant had to meet the requirements of Table 1.

In total our participants were 21 males, 23 females. There were no gender differences in any background measure including: age, spreadsheet experience, programming experience, and education. Most participants (33/44) considered themselves Excel "intermediates." Median ages were 48 for males and 44 for females. Education was primarily at the baccalaureate level. Only 9 participants (6 males and 3 females) had *never* created a spreadsheet for professional use.

Table 1: The minimum requirements participants had to meet in order to participate in the study.

Type	Requirement	Rationale
Age	20-60: 60 was the upper limit.	To generalize, we wanted a wide range of ages. Upper limit was set to avoid confounding factors due to deteriorating eyesight and other cognitive factors that occur with age.
Profession	Participants classifying themselves as a software developer, IT professional, computer engineer, or electrical engineer were disqualified.	Our interest was in end-user programmers. These professions are closer to professional programming than to end-user programming.
Excel experience	Participants could classify their Excel experience as: beginner, intermediate, advanced, or expert. Answering "no experience" disqualified them.	Since the population of interest to us is people already engaged in this type of end-user programming, some prior experience with Excel was required.
Programming background	Participants who had taken 2 or more courses in Java- and/or Perl- like programming were disqualified. (Web programming and Visual Basic were also allowed.)	Some programming coursework was allowed because, given modern business degree requirements, young business adults have usually taken 1-2 programming courses in high school and/or college.
Experience with macros	Participants who had programmed Excel macros were disqualified.	This level of sophistication with Excel is beyond that of many business users.
Disqualifying features	If participants had previously used data validation, the watch window, or evaluate formula they were disqualified.	We wanted to analyze the use of these specific features without the participants having prior knowledge of them.
Qualifying Features	Participants had to have used three or more of the functions from the following list: average, count, countif, hlookup, if, indirect, lookup, max, min, round, sum, and sumif.	To avoid spreadsheet illiteracy as a confound, it was important to ensure that participants had some experience with reasonably complicated formulas.

4.2 Environment

The experiment took place using the real-world environment of Microsoft Excel 2001. Because Excel is the mostly widely used end-user programming language, this environment is an ideal choice for examining the generalization of the gender-debugging study results.

To as closely as possible replicate the purpose of the gender-debugging study, this study followed the same objective of end-user software engineering. We thus focused on factors and features in Excel that would promote the reliability of the spreadsheet formulas. Excel's audit toolbar feature has several features that aid users in ensuring reliability, and allow them to engage in end-user software engineering activities. As shown in Figure 2, the audit toolbar contains 12 buttons. Five of these (numbers 2-6 in Figure 2) support operations with dataflow arrows. Two (numbers 1 and 7) relate to Excel's error checking of cells flagged as being inconsistent or otherwise suspicious. Two (numbers 9 and 10) relate to Excel's "validation" feature, in which users can check if any of their cells' values violate expected ranges (also set by the user). Finally, number 11 is for watching cell's values that may be off-screen or on another sheet all together; and number 12, "evaluate formula," allows a user to go step-by-step in evaluating a formula.

Although we focused on the audit toolbar, we did not in any way restrict the participants to only these features. In comparison to the gender-debugging study (with only 4 features – see Figure 1), participants in this study had to choose between hundreds of Excel features.

4.3 Tutorial

The tutorial was designed under the same requirements as the gender-debugging study [2]. As with the gender-debugging study's tutorial, it was hands-on, and lasted about 30 minutes. Its purposes were to (1) focus participants' attention on the goal of formula reliability, (2) teach features in the "taught" category, and (3) also call attention to (but not teach) features in the "untaught" category.

The taught features were the arrows (numbers 2-6 from Figure 2). For these features, the instructor described how to use the feature and its feedback once used. The taught features were used multiple times during the study. The untaught features singled out by the instructor were the error checking buttons and the evaluate formula button (numbers 1 and 12). Users were encouraged to explore all audit toolbar features.

Throughout the tutorial, participants learned to use the taught features, and experimented as they wished with the untaught ones focusing on the reliability of the

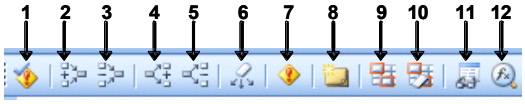

Figure 2: To stay with the theme of end-user software engineering the experiment emphasized the use of the features in the audit toolbar to aid formula reliability.

spreadsheet as they worked. They also learned about the "IF" function in Excel, because in previous studies, a number of participants have stumbled on its use, and we wanted to avoid introducing "noise" relating to misunderstandings of IF into our data.

The spreadsheet they worked on during the tutorial came from the EUSES corpus of real-world spreadsheets [10], with slight modifications for tutorial suitability. The spreadsheet was a learning styles questionnaire; participants' task during the tutorial was to introduce two new rows for two new learning styles questions – which would then have to be accounted for in several downstream formulas. One of these was completed step-by-step during the tutorial. This maintenance-style task was designed to be similar to one of the tasks in the main part of the experiment.

At the end of the tutorial, once one of the modifications had been made, the participants had several minutes to explore the features they had just learned about and to make the second modification (add the next question) to the spreadsheet.

4.4 Main spreadsheet and tasks

The main experiment required participants to make two modifications to a grade book spreadsheet. This spreadsheet, also obtained from the EUSES Spreadsheet Corpus of real-world spreadsheets [10], is shown in Figure 3.

We chose to make the tasks modification tasks—instead of debugging as in the gender-debugging study—for generalization purposes. Modification includes debugging, and hence covers both the "create" and the "debug" phase of end-user programming.

The modification tasks were designed with two criteria in mind. First, they needed to be grounded in the real world. For this reason, we drew the spreadsheet and the task ideas from the EUSES Corpus of real-world spreadsheets. Second, the tasks needed to be complicated enough to warrant use of the auditing toolbar features. If the modification tasks were too easy, we feared there would be no reason for participants to consider use of these features.

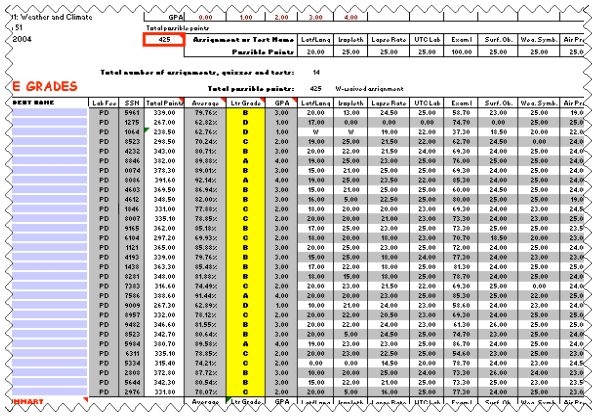

Figure 3. A snapshot of part of the grade book spreadsheet from the EUSES Spreadsheet Corpus [10]. Participants' tasks were to make modification to formulas, and add a new lab section. Figure is shrunk and cropped to give a sense of the overall size.

The first modification task (#1) was drawn directly from a second real-world spreadsheet from the corpus, in which the teacher was incorporating lab assignments into the students' grade. The second modification (#2) was to solve an error proneness problem with the current spreadsheet. Without the second modification, teachers would have to manually override formulas for students with waived homework assignments; the modification was thus to instead change the formulas so that they could calculate the grades for any student with or without waived homeworks.

4.5 Questionnaires

A pre-session questionnaire collected participant background data. It also included self-efficacy questions based on a slightly modified version of Compeau and Higgins' validated scale [7]; the modifications made the questionnaire task-specific to spreadsheet modifications. Participants were asked to answer on a five-point Likert scale their level of agreement with the statements. For example, "…I could modify [a] spreadsheet to change how it calculates formulas and ensure it works properly… if there was no one around to tell me what to do as I go…" and "…if I had seen someone else using it before trying it myself."

The following background data were collected: gender, degree program, highest degree completed, current job title, programming experience, previous spreadsheet experience, professional spreadsheet experience, 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), regardless of any other talents they may have.

Following the experiment a post-session questionnaire assessed comprehension of the audit toolbar features with a set of 22 multiple choice and true/false questions.

5. Results that Generalized

5.1 Effectiveness

Gender-Debugging Study Result: Females' self-efficacy was predictive of their effectiveness at using the debugging features (which was not the case for the males).

To analyze this question, we used the measure possible with the data that was most related to effectiveness with the debugging features. This was a measure of success on the task – specifically how much of the two tasks were attempted and/or completed. To determine "how much" we divided the tasks into small subtasks. Points were assigned for correctly completing tasks, with partial credit for (incorrectly) attempting completion of the subtask. The sum of points is the "task success."

Females' self-efficacy was a predictive indicator of their task success (linear regression: $F(1,21)=7.2$, $ß=0.47$, $R^2=0.26$, $p<0.01$). Males' self-efficacy, however, was not a significant predictor of their task success (linear regression: $F(1,19)=2.19$, $ß=0.15$, $R^2=0.10$, $p<0.16$). Figure 4 shows the males' and females' relationships between self-efficacy and task success. These findings are consistent with the above gender-debugging study result.

Self-efficacy was also a significant predictor of task success for all participants ($F(1,42)=6.99$, $ß=0.24$, $R^2=0.14$, $p<0.01$), but this was due to the females. This is indicated by the R^2 values—a measure of how much of the variance in task success self-efficacy described—showing that the separate analysis of the preceding paragraph provides a better fit to the female data than with the combined group, and that most of the male outcomes were not explained by self-efficacy.

5.2 Comprehension

Gender-Debugging Study Result: No difference in the males' and females' ability to learn the new features.

One possible explanation for the result of Section 5.1 is that the females made better judgments than the males did regarding their abilities to understand and use the features effectively.

In the gender-debugging study, this was not the case. A comprehension post-test showed that there was no difference in males' and females' understanding of how the debugging features worked, or interpretation of their feedback, etc. Females' low self-efficacy was a self-fulfilling prophecy: low belief in their ability impacted their willingness to engage with the features, although their understanding would not have predicted this difference.

Turning to the current study, the comprehension post-test also showed no difference in males' and females' comprehension of the audit toolbar features with females scoring a median of 12 points (22 possible), males a median of 11 (t-test: $t=0.72$, $df=42$, $p<0.47$). Furthermore, self-efficacy also did not predict comprehension for either gender (linear regression: males: $F(1,19)=0.16$, $ß=-0.04$, $R^2=0.008$, $p<0.69$; females: $F(1,21)=0.56$, $ß=0.12$, $R^2=0.03$, $p<0.46$). These results suggest that, as in the gender-debugging study, females' self-efficacy was more of a self-fulfilling prophecy than an accurate assessment of abilities.

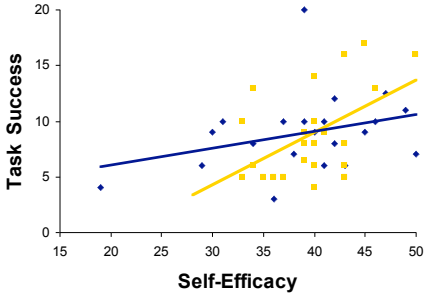

Figure 4. For the females (light), self-efficacy predicted task performance. This relationship did not for the males (dark).

5.3 Familiar Features

Gender-Debugging Study Result: Females had a higher adoption rate of the Type Familiar feature (formula edits) than the males did.

Familiar features were those features not defined as taught or untaught features (see Section 4.3). The category included formula and value manual edits (i.e., without using replicate features), and basic features such as bold, copy/paste, and insert function. A t-test revealed no gender differences in the overall use of the familiar features (t-test: $t=0.51$, $df=42$, $p<0.62$). But, as Figure 5 clearly suggests, females' and males' relationships between their self-efficacy and use of the familiar features differed. Females' self-efficacy was inversely predictive of their use of these features: as their self-efficacy decreased their use of the familiar features increased (linear regression: $F(1,21)=10.08$, $ß=-15.8$, $R^2=0.32$, $p<0.005$). For males, the relationship is not significant ($F(1,19)=0.49$, $ß=-2.65$, $R^2=0.03$, $p<0.49$).

The regression relationship for the females is consistent with the gender-debugging study. Specifically, low self-efficacy females concentrated more of their

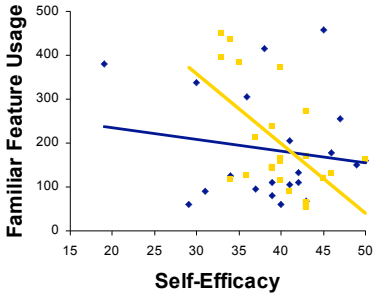

Figure 5. For the females (light), lower self-efficacy was a significant predictor of higher usage of familiar features. For the males (dark), there was no relationship.

efforts on the familiar features, particularly when compared to the high self-efficacy females.

Closer scrutiny of these relationships showed that they were almost entirely due to (manual) value edits. Unlike the gender-debugging study, which had only one feature classified as familiar—formula edits, not value edits—in this study, there were several groups classified as such, as shown in Table 2. (The role of values was different in the previous study, due to the connection with a testing tool.)

Females' self-efficacy inversely predicted use of these manual value edits (but no other sub-category), as shown in Figure 6 (linear regression: edit values: $F(1,21)=12.08$, $ß=-17.22$, $R^2=0.37$, $p<0.002$; edit formulas: $F(1,21)=2.10$, $ß=1.14$, $R^2=0.09$, $p<0.16$; other features: $F(1,21)=0.03$, $ß=0.14$, $R^2=0.001$, $p<0.86$). For males, self-efficacy did not predict any use of the three sub-categories (linear regression: edit values: $F(1,19)=0.32$, $ß=-1.9$, $R^2=0.02$, $p<0.58$; edit formulas: $F(1,19)=0.21$, $ß=-0.44$, $R^2=0.01$, $p<0.65$; other features: $F(1,19)=0.08$, $ß=-0.29$, $R^2=0.004$, $p<0.78$).

The distinction between values entered manually and those entered using replicate (e.g., by copy/pasting to several cells, fill functions, and the cross bar in the lower-right corner of a cell) provided insights into the females' behaviors. In fact, values entered using replicate were predicted by self-efficacy for the females, but the relationship is *opposite* of the manual value edits. Shown in Figure 7, self-efficacy is a predictive indicator for the percentage of females' total value edits that are filled using replicate, but not so for the males (linear regression: males: $F(1,19)=2.37$, $ß=1.17$, $R^2=0.11$, $p<0.14$; females: $F(1,21)=6.26$, $ß=3.9$, $R^2=0.23$, $p<0.02$). This same pattern is consistent for formula edits entered using replicate (linear regression: males: $F(1,19)=0.94$, $ß=2.19$, $R^2=0.05$, $p<0.34$; females: $F(1,21)=9.05$, $ß=14.6$, $R^2=0.30$, $p<0.007$).

Self-efficacy theory provides an interpretation for this difference. According to self-efficacy theory "people tend to avoid tasks and situations they believe exceed their capabilities, but they undertake and perform assuredly activities they judge themselves capable of handling" [1]. From this perspective, low self-efficacy females spending their time manually entering values may be avoiding a challenging part of the task.

For example, one aspect of task #1 (see Section 4.4), providing additional columns, could be interpreted to mean that many data values should be typed in. Another subtask was to write an "IF" formula, arguably the most challenging sub-task of task #1. Of the 10 low self-efficacy females (defined as females with self-efficacy below the median of 40), only 1 attempted the "IF" (unsuccessfully), compared with the high self-efficacy females of whom 6 of the 13 made the change correctly: a statistically significant result (t-test: $t=2.3$, $df=21$, $p<0.03$). This result suggests that low self-efficacy females, in avoiding a subtask they believed exceeded their capabilities, focused on the part of the task they knew they could do—manually entering values.

Table 2. Mean (std. dev.) for the components of the familiar features. No significant gender differences in usage.

Familiar Features	Males	Females
Value (manual) edits	125.1 (111.8)	151.7 (124.3)
Formula (manual) edits	29.5 (31.5)	26.0 (16.5)
Other basic features	28.0 (33.5)	25.2 (15.8)

6. Unconfirmed Results

Some of the gender-debugging study results were not confirmed. Those are discussed briefly here.

Gender-Debugging Study Result: females had significantly lower self-efficacy than the males.

In the current study, both males and females had a median self-efficacy of 40 (t-test: $t=0.41$, $df=42$, $p<0.68$). We also found no gender differences in self-efficacy in a previous study [3]. For researchers and designers concerned about gender differences, the bottom line is that no assumptions should be made regarding whether females will or will not have lower self-efficacy than the males. Even so, for females, low self-efficacy when present had more detrimental effects than for low self-efficacy males.

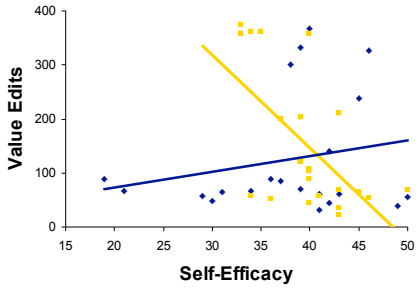

Figure 6. Self-efficacy (inversely) predicts value edits for females (light), but not for males (dark).

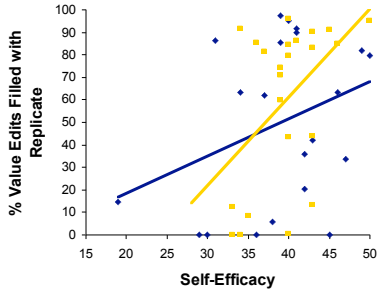

Figure 7. Self-efficacy predicts percentage of value edits filled using replicate for the females (light), but not the males (dark).

Gender-Debugging Study Result: Males were more willing to adopt the new features: they performed significantly more Type Taught actions than females. Furthermore, significantly more males used Type Untaught features than females did.

For the type taught and untaught features, there were no statically significant gender differences in usage (t-test: taught: t=-0.76, df=42, p<0.45; untaught: t=0.065, df=42, p<0.95). However, the relationship between self-efficacy and untaught feature usage is revealing. Figure 8 shows the suggestive relationships, and how those differ for the males and females. For the females, their self-efficacy was suggestively predictive of untaught feature usage, but for the males suggestive relationship is the opposite (linear regression: males: $F(1,19)=0.41$, ß=-0.08, $R^2=0.02$, p<0.53; females: $F(1,21)=1.92$, ß=0.21, $R^2=0.084$, p<0.18). In essence, gender differences in the use of untaught features were not confirmed for a commercial spreadsheet environment.

Gender-Debugging Study Result: no significant difference between the females' and males' performance in fixing seeded bugs, but the females introduced significantly more bugs than the males did.

There was no gender difference in task success: males and females scored a median of 10 and 9 points respectively (t-test: t=0.46, df=42, p<0.65). In the gender-debugging study the gender differences in introduced bugs may be due to the females' lower self-efficacy in that study. A subsequent study also found no gender differences in performance [3]. These findings, in combination with our original results, amount to this: when there are no differences in self-efficacy, there is no evidence of lower task performance by female end-user programmers.

7. Discussion

This work has generalized one particular study. Another recent study we did investigated "tinkering" as another interesting behavior in end-user programming environments [3]. We would have liked to consider its applicability to Excel features, but there was not enough activity on any one feature. Overall, participants used 61 different Excel features, with a median of 12 different features per participant.

A central outcome from the current study is that neither gender alone nor self-efficacy alone was a particularly useful predictor of the outcomes for this task of spreadsheet maintenance. Rather, the impact of self-efficacy on behavior was different for male than for female end-user programmers. This outcome is consistent with our other studies of end-user software engineering tasks as well [2, 3, 4].

Because this phenomenon is consistently present in

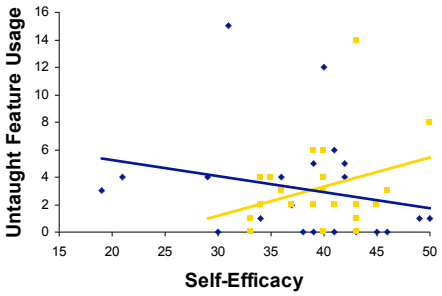

Figure 8. Notice that the suggested relationship between self-efficacy and untaught feature usage is nearly the opposite for the males (dark) and females (light).

1. Choose an environment, preferably one that supports logging (for easy data capture).
   Our study: We chose Excel.
   Alternatives for replication/generalization: Other environments.

2. Choose participants. Ensure that their experience level does not interfere with your choice of features to study.
   Our study: Real end-user developers, familiar with Excel, subject to the limitations described in Section 4.1.
   Alternatives for replication/generalization: End-user developers with at least some prior experience with the environment.

3. Choose a task for the participants to complete.
   Our study: Modification task, with both the original program and the modification ideas drawn from real-world spreadsheets from the EUSES Spreadsheet Corpus.
   Alternatives for replication: Different spreadsheets from the same corpus; programs drawn from some other corpus of software developed by end users.
   Alternatives for generalization: Debugging, with bugs harvested from other end users; some other end-user software development task, such as comprehending, reusing, testing, ...

4. Create a tutorial that teaches the "taught" features, briefly calls attention to the "untaught" features, and illustrates the correspondence between the usefulness of features and the task the participants are doing.
   Our study: Described in Section 4.3.
   Alternatives for generalization: An on-line self-study guide.

5. Create a pre-task and post-task questionnaire.
   Our study: Described in Section 4.5.
   Alternatives for generalization: Questionnaire could be tailored for different research goals.

Figure 9. How to replicate or generalize the experiment in other environments.

our studies, including this one showing its presence in a commercial environment, there is now significant evidence that it is real, at least for spreadsheets. To encourage other researchers interested in exploring its applicability to other end-user programming environments, Figure 9 summarizes how to repeat it.

8. Conclusion

We have presented our process and subsequent results of replicating and generalizing the gender-debugging study that first revealed gender differences in end-user programming environments. We have found that several of the results from that study generalize to the commercial environment, namely:

- Females' self-efficacy predicted task success, but the same did not hold true for the males.
- Low self-efficacy females were more engaged with the type familiar features, particularly value edits. Self-efficacy theory suggests that they may have avoided more challenging aspects of the tasks. (Males' usage did not suggest this same explanation.)
- The above results cannot be attributed to females being better judges of their weaknesses: Females' comprehension of the software features were no different than the males' and were not predicted by self-efficacy.

This is the first study in a commercial environment, but the fourth study in total [2, 3, 4], in which we have found that the effects of self-efficacy play out differently for male and female end-user programmers.

Acknowledgements

We thank Mary Czerwinski and her team at Microsoft Research for ideas, assistance, and hosting this study. This work was also supported by Microsoft Research UK, by NSF grant CNS-0420533, and by the EUSES Consortium via NSF grant ITR-0325273.

References

[1] Bandura, A. *Social foundations of thought and action: a social cognitive theory*, Prentice-Hall, Englewood Cliffs, N.J. 1986.

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

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

[4] Beckwith, L., Burnett, M., Grigoreanu, V. & Wiedenbeck, S. Gender HCI: What about the software? *IEEE Computer 39*(11), 2006, 83-87.

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

[6] Busch, T. Gender differences in self-efficacy and attitudes toward computers. *J. Educational Computing Research 12*, 1995, 147-158.

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

[8] Corston, R., & Colman, A.M., Gender and social facilitation effects on computer competence and attitudes toward computers, *Journal of Educational Computing Research 14*(2) 1996, 171-183.

[9] Durndell, A. & Haag, Z. Computer self-efficacy, computer anxiety, attitudes toward the Internet and reported experience with the Internet, by gender, in an East European sample. *Computers in Human Behavior 18*, 2002, 521-535.

[10] Fisher II, M. & Rothermel, G. The EUSES Spreadsheet Corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanism. *WEUSE05: 1st Workshop on End-User Software Engineering*, 2005, 47-51.

[11] Hargittai, E. & Shafer, S. Differences in actual and perceived online skills: The role of gender. *Social Science Quarterly 87*(2), 2006, 432-448.

[12] Kelleher, C., Pausch, R., & Kiesler, S. Storytelling Alice motivates middle school girls to learn computer programming. *ACM Conf. Human-Computer Interaction*, 2007 (to appear).

[13] McCoy, L.P., Heafner, T.L., Burdick, M.G. & Nagle, L.N. Gender differences in computer use and attitudes on a ubiquitous computing campus. *AERA Annual Meeting*, 2001, 2-7.

[14] *Merriam-Webster Online Dictionary.* 2006-2007. http://www.merriam-webster.com (accessed: March 18, 2007).

[15] Rode, J.A., Toye, E.F. & Blackwell, A. The Fuzzy Felt Ethnography – understanding the programming patterns of domestic appliances. *2nd International Conference on Appliance Design*, 2004, 10-22.

[16] Rosson, M.B., Ballin, J. & Nash, H. Everyday programming: Challenges and opportunities for informal web development. *Visual Languages and Human-Centric Computing,* 2001, 123-130.

[17] Spotts, T.H., Bowman, M.A., & Mertz, C., Gender and use of instructional technologies: A study of university faculty. *Higher Education 34*, 1997, 421-436.