Interacting with Computers 23 (2011) 450–460

Contents lists available at ScienceDirect

Interacting with Computers

journal homepage: www.elsevier.com/locate/intcom

Gender pluralism in problem-solving software ☆

Margaret M. Burnett [a,*], Laura Beckwith [a,c], Susan Wiedenbeck [b], Scott D. Fleming [a], Jill Cao [a], Thomas H. Park [b], Valentina Grigoreanu [a,d], Kyle Rector [a,e]

[a] Oregon State University, 1148 Kelley Engineering Center, Corvallis, OR 97331, USA
[b] Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104, USA
[c] HCI Researcher, Sønder Boulevard 63, 4.th, 1720 Copenhagen, Denmark
[d] Microsoft, One Microsoft Way, Redmond, WA 98052, USA
[e] University of Washington, AC101 Paul G. Allen Center, Box 352350, 185 Stevens Way, Seattle, WA 98195, USA

ARTICLE INFO

Article history:
Received 2 June 2010
Received in revised form 25 June 2011
Accepted 25 June 2011
Available online 8 July 2011

Keywords:
Gender
Problem-solving software
Spreadsheet debugging

ABSTRACT

Although there has been significant research into gender regarding educational and workplace practices, there has been little awareness of gender differences as they pertain to software tools, such as spreadsheet applications, that try to support end users in problem-solving tasks. Although such software tools are intended to be gender agnostic, we believe that closer examination of this premise is warranted. Therefore, in this paper, we report an end-to-end investigation into gender differences with spreadsheet software. Our results showed gender differences in feature usage, feature-related confidence, and tinkering (playful exploration) with features. Then, drawing implications from these results, we designed and implemented features for our spreadsheet prototype that took the gender differences into account. The results of an evaluation on this prototype showed improvements for both males and females, and also decreased gender differences in some outcome measures, such as confidence. These results are encouraging, but also open new questions for investigation. We also discuss how our results compare to generalization studies performed with a variety of other software platforms and populations.

1. Introduction

An important cultural issue in the technological world is gender (Bardzell and Feminist, 2010; Margolis and Fisher, 2003; Zeldin and Pajares, 2000). From a feminist perspective, *pluralism* is the quality of designing artifacts that "resist any single, totalizing, or universal point of view" (Bardzell and Feminist, 2010). Pluralism implies that designers can produce more inclusive designs through sensitivity to marginal or marginalized users. To inform the design of gender-pluralist software, we are investigating gender differences in the ways males and females engage in problem solving and use problem-solving software.

Understanding gender differences in problem-solving software may provide important benefits to the users of such software. If males and females work differently with problem-solving features, then a software environment may be unintentionally biased to the needs of one gender while marginalizing the other. Understanding such differences would reveal ways to design environments that serve both genders more equally—ideally without disadvantaging either gender.

In this paper, we begin with a theory-based foundation (Section 2). Drawing from these theories, we then present investigations to understand whether and how gender differences manifest in the context of problem-solving software. Our core studies focus on spreadsheet testing and debugging because spreadsheets environments are ubiquitous and used by end users in tasks common for males and females in the workplace, such as accounting, budgeting, and calculating student grades. Moreover, researchers have observed high error rates in spreadsheets developed by end users (Panko and Aurigemma, 2010), which suggests that problem-solving in that software context is difficult for at least some users. We therefore investigate software-based problem-solving through three studies in the context of spreadsheet debugging (Sections 3 and 4) that together address the four research questions below. We then discuss several related studies (Section 5) that generalized the results from the first three research questions.

RQ1: Are there gender differences as to *which features* males and females use in problem-solving software?
RQ2: Are there gender differences as to males' and females' willingness to *tinker and explore*?

☆ This paper includes updated reports of the studies previously described individually in Beckwith et al. (2005, 2006) and Grigoreanu et al. (2008).

* Corresponding author.
E-mail addresses: burnett@eecs.oregonstate.edu (M.M. Burnett), beckwith@hciResearcher.com (L. Beckwith), sw53@drexel.edu (S. Wiedenbeck), sdf@eecs.oregonstate.edu (S.D. Fleming), caoch@eecs.oregonstate.edu (J. Cao), tp352@drexel.edu (T.H. Park), valeng@microsoft.com (V. Grigoreanu), rectorky@cs.washington.edu (K. Rector).

0953-5438/$ - see front matter © 2011 British Informatics Society Limited. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.intcom.2011.06.004

RQ3: Are any such differences related to males' or females' technical problem-solving *confidence*?
RQ4: If gender differences exist, is it possible to design problem-solving software that overcomes gender gaps *without* penalizing either gender?

The answers to RQ1–RQ3 may have considerable implications for software product design, which we consider in RQ4. For example, many software products are designed around the notions that the software's problem-solving features are gender neutral, and that when users notice new features relevant to their tasks, they will often have enough confidence and interest to tinker with the new features. If these assumptions are not true for both males and females, software product designers will want to consider changing their products. Making such changes is likely to help both genders: research has shown that studying the needs of one subpopulation can provide benefits that extend beyond that subpopulation (Ljungblad and Holmquist, 2007). For example, phenomena that affect large numbers of females are likely to affect some males as well.

As to *how* to follow up on implications of RQ1–RQ3, fortunately, theories offer not only explanations of why the differences we investigate here could occur, but also offer a useful framework for thinking about how to act upon the differences (RQ4). We therefore begin by considering these theoretical foundations.

2. Theoretical background

2.1. Self-efficacy theory

Self-efficacy, a type of confidence, is a person's belief in his or her ability to perform a specific task (Bandura, 1986). According to self-efficacy theory, self-efficacy distinguishes how individuals will approach and perform a task. In particular, self-efficacy affects performance when a task becomes challenging (Bandura, 1986). People with high self-efficacy tend to put in more effort, persist in challenging tasks, and have more genuine interest in the task than people with low self-efficacy. Ultimately self-efficacy affects performance outcomes, influencing whether or not an individual succeeds at the task.

According to the theory, individuals with high self-efficacy on a task possess several characteristics that aid their success and that individuals with low self-efficacy lack (Bandura, 1986). One such characteristic is propensity for generating and trying alternative strategies when the current strategy is failing. Another characteristic is willingness to abandon faulty strategies (which generally implies determining an alternative strategy). In contrast, individuals with low self-efficacy are less likely to abandon faulty strategies, and less likely to try alternative strategies.

Computer self-efficacy refers to a person's judgment of his or her capabilities to use computers in a variety of situations (Compeau and Higgins, 1995). Researchers have reported gender differences in computer self-efficacy across nationalities and across levels of computer expertise (e.g., Beyer et al., 2003; Busch, 1995; Colley and Comber, 2003; Durndell and Haag, 2002; Margolis and Fisher, 2003; Zeldin and Pajares, 2000).

Computer self-efficacy is particularly pertinent to software tool design because a user's computer self-efficacy influences how he or she will respond to design strategies intended, for instance, to encourage the user to learn about new or unfamiliar software features. For example, Lowenstein's information-gap perspective on curiosity states that users will become curious and investigate when appropriately surprised. Designers applying this perspective rely on users having a certain level of confidence for the surprise to generate curiosity and follow-up rather than anxiety and avoidance (Loewenstein, 1994). Thus, in the context of problem solving with software, low self-efficacy may interfere with the user's willingness to explore and ultimately adopt unfamiliar features (RQ1, RQ2). This potential effect of low self-efficacy also raises the question of whether any gender differences in feature usage may be due solely to differences in computer self-efficacy (RQ3).

2.2. Selectivity hypothesis

The selectivity hypothesis synthesizes earlier theories of gender differences with respect to information processing (Meyers-Levy, 1989). According to the hypothesis, males tend to process information in a heuristic manner, paying particular attention to cues that are highly available and particularly salient in the focal context. In contrast, females process information in a comprehensive manner, attempting to assimilate all available cues. The hypothesis goes onto suggest that males may focus on themselves when processing information. This self-focus helps to streamline processing because memory processing of information pertaining to oneself has a particularly well-developed and elaborate network of associations. According to the hypothesis, females may take a different approach, processing equally information relevant both to others (external world) and themselves. Meyers-Levy and others have presented empirical evidence that supports the hypothesis (Meyers-Levy, 1989).

In the context of computer-based problem-solving environments, the selectivity hypothesis implies that males and females will respond differently to different cues and features in the environment, which should impact problem-solving decisions. Moreover, these differences imply that both *what* information the environment presents and *how* the environment presents information will impact the tool's effectiveness differently for males versus females. With respect to our investigations, the theory not only predicts gender differences in feature usage (RQ1), but also suggests design remedies (RQ4) based on supporting both genders' information processing styles.

2.3. Attention investment model

In contrast to the above theories, the attention investment model specifically attends to users' decisions about technology usage. The model is an analytical model of problem-solving behavior that accounts for the perceived costs, benefits, and risks that a user considers in deciding how to complete a task in a software environment (Blackwell, 2002).

Gender comes into play because it potentially influences the perception of cost, benefit, and risk. For example, due to the self-efficacy differences discussed in Section 2.1, a female's perception of the cost of learning a new feature may be higher than a male's perceived cost to learn the same feature. Researchers have also pointed to differences in motivations to use technology (Brunner et al., 1998), which suggest gender differences in perceived benefits.

Regarding the risk factor, the model suggests that a user's perception of risk may influence exploration of unfamiliar features while problem solving. For example, a user might notice an unfamiliar but potentially useful feature in the environment. However, the user may not try the feature if he or she perceives a substantial risk, such as the significant loss of time trying to learn the feature without ultimately succeeding.

Gender differences in perceived risk have a rich literature. Research on risk perception suggests that females perceive more risk from everyday life decisions and situations than do males (Barke et al., 1997; Blais and Weber, 2001; Byrnes et al., 1999; Finucane et al., 2000; Hudgens and Fatkin, 2001). Researchers have also found that females are more risk averse than males in making financial decisions (Jiankoplos and Bernasek, 1998) and in making informed guesses when a wrong guess produces negative consequences (Byrnes et al., 1999).

These gender differences in perceptions of cost and risk suggest that females may be less likely to explore unfamiliar features than males (RQ2) and may therefore use different features than males (RQ1). The attention investment model also suggests a design remedy (RQ4) for users whose perceptions of costs or risks are inappropriately high: a software environment could help such users bring their perceptions of costs, benefits, and risks in line with reality, resulting in a better-informed decision as to whether to try an unfamiliar feature.

3. Studies #1 and #2

Based on our survey of the theoretical literature, we conducted two studies to understand how gender differences manifest in the context of testing and debugging spreadsheets.

3.1. Study #1: feature use and self-efficacy

Our first study investigated gender differences in feature use (RQ1) and the influence of confidence on those differences (RQ3). In particular, we checked for differences in males' and females' willingness to use familiar versus unfamiliar features for testing spreadsheets. We also looked at how the differences we found related to self-efficacy in finding and fixing spreadsheet errors.

3.1.1. Spreadsheet environment

We conducted the study in the context of the Forms/3 spreadsheet environment, which includes a number of features that have been previously shown to help users find and fix errors in spreadsheet formulas (Burnett et al., 2004).

In Forms/3 (Fig. 1), users can "check off" cells whose computed values they decide are correct (e.g., `Exam_Avg`). These decisions constitute "tests". As the user checks off values as correct, cell border colors update to indicate how fully tested the cell is. Although users may not realize it, these "testedness" colors reflect the use of a dataflow test adequacy criterion that measures the interrelationships in the formulas that have been covered by the users' tests (Rothermel et al., 2001). A cell is fully tested if all its interrelationships have been covered; if only some have been covered then the cell is partially tested. Along the top of the Forms/3 window, a progress bar shows the average percent "testedness" of all cells in the spreadsheet (e.g., 30% tested).

The user can also "X off" cells with incorrect values (e.g., `Course_Avg`). The interior color of a cell indicates the likelihood that the cell contains a fault. A fault-likelihood bar (below the testedness bar) shows how much of the spreadsheet has particular levels of fault likelihood.

Arrows display dataflow relationships among cells (e.g., the arrow from `Quiz5` to `Quiz_Avg`). The arrows follow the testedness color scheme to show how thoroughly a relationship between two cells has been tested. In Fig. 1, the user has displayed `Quiz5`'s arrow showing that the formula in `Quiz_Avg` references `Quiz5`. The color of the arrow indicates that the user has not tested the relationship.

3.1.2. Participants and procedures

The participants were 27 male and 24 female undergraduates, all of whom were familiar with spreadsheets. Most of the participants were business students, and none were studying computer science. Background data collected with a pre-experiment questionnaire indicated that the gender groups did not have significant differences in major, grade point average (GPA), or experience with spreadsheets, programming (as measured by number of classes and years of experience at the high school, college, and professional level), or the study's spreadsheet environment. (We did not collect background data relating to affinity for mathematics or computing because spreadsheets are expected to be used successfully in the workplace by males and females regardless of potential gender differences in such affinities.) The females were academically younger than the males by a small but significant margin (ANOVA: $F(1, 48) = 4.53$, $p < .04$). Post-hoc analyses, however, showed that academic age was not predictive of any outcome measures.

Prior to the main study session, we assessed participants' pre-task self-efficacy (*pre-self-efficacy*) by adapting Compeau and Higgins' computer self-efficacy questionnaire for spreadsheet-debugging tasks (Compeau and Higgins, 1995). The original questionnaire consisted of a prompt, "I could complete the job using the software package if ..." followed by ten different completions of the sentence, such as "... I had used similar packages before this

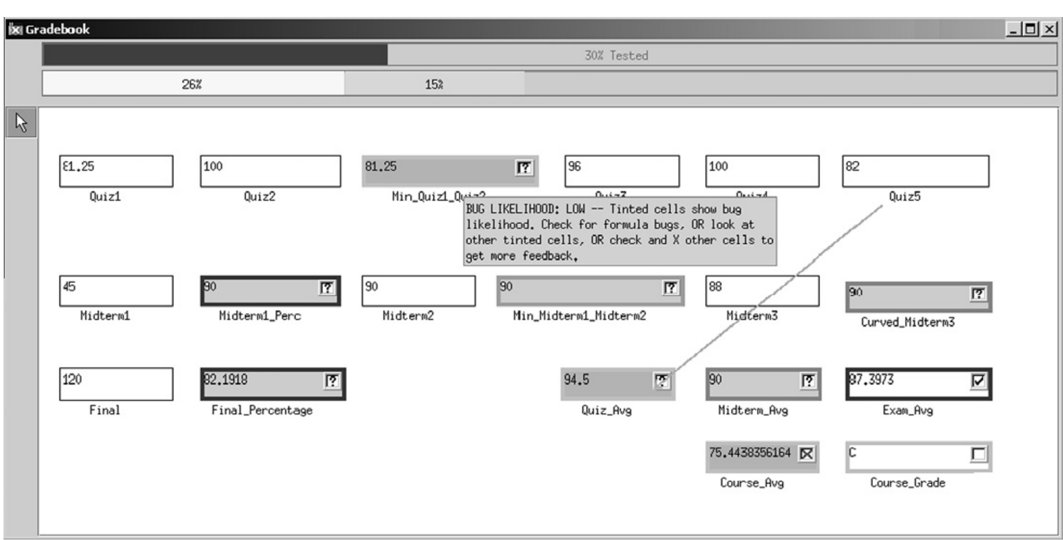

Fig. 1. The user notices an incorrect value in Course_Avg—the value is obviously too low—and places an X-mark in the cell. As a result, Forms/3 colors the eight cells that may have directly contributed to the error.

one to do the same job.'' Each completion reflected a different level of self-efficacy, and allowed responses on a 10-point scale from ''not at all confident'' to ''totally confident''. To adapt this questionnaire to spreadsheet debugging, we changed the prompt to ''Given a spreadsheet which performs common tasks (such as calculating course grades or payroll) I could find and fix errors if . . .'' In each completion, we changed the word *job* to *task* and *package* to *spreadsheet* (e.g. ''. . . I had used similar *spreadsheets* before this one to do the same *task*''). For consistency with other questions on our questionnaire, we also changed the response scale to a 5-point scale from ''strongly disagree'' to ''strongly agree''. (Note that this scale change is a departure from the Compeau/Higgins questionnaire, implying that our results should not be compared with other researchers' uses of this questionnaire. Therefore we compare our self-efficacy results within only our own participants and our own studies.)

Participants then completed a hands-on tutorial to become familiar with Forms/3. The tutorial thoroughly covered Forms/3's checkmarks and arrows features, but merely mentioned X-marks. Varying the coverage of these features enabled us to check for three levels of feature usage by participants: (1) use of previously familiar features (formula editing), (2) use of unfamiliar but taught features (checkmarks and arrows), and (3) use of unfamiliar and untaught features (X-marks).

For the main study tasks, participants were to find and fix errors in two spreadsheets, each seeded with five bugs. One spreadsheet, *Gradebook*, contained a teacher's gradebook, and the other, *Payroll*, captured a real company's method of calculating payroll checks. For each task, we instructed participants to ''Test the . . . spreadsheet to see if it works correctly and correct any errors you find.'' Participants had 22 min to complete the Gradebook task and 35 min to complete the Payroll task. We assigned the tasks in random order.

3.1.3. Results

To address RQ1, we measured usage of familiar and taught features by counting number of uses (formula edits and feature toggles). From the taught-features count, we excluded repeated toggles on the same cell after ceasing formula edits because such toggling suggested an absence of intellectual involvement. We considered usage of the untaught features when a participant placed more than one X-mark and subsequently edited a colored cell's formula. Because only about 60% of the participants exhibited this behavior and their frequency of usage was low (1 or 2 was typical), we classified participants as having used or not used untaught features rather than counting instances of use. Analysis revealed that females relied on familiar features significantly more than males (ANOVA: $F(1, 49) = 4.979$, $p < .03$) (Table 1). In contrast, males used both taught features (ANOVA: $F(1, 49) = 4.971$, $p < .03$) (Table 2) and untaught features (Fisher's Exact Test: $p < .001$) (Table 3) significantly more than females.

How did the differences in feature usage play out in terms of success? There was no significant gender difference in the number of bugs fixed (Mann–Whitney: $U = 300.5$, $p < .651$). However, females introduced significantly more bugs than males (Mann–Whitney: $U = 227.5$, $p < .011$), and were significantly more likely than males to introduce bugs (Fisher's Exact Test: $p < .015$). Table 4 summarizes these results. Females may have introduced more bugs than males simply because females edited formulas more.

Table 1
Males versus females' use of familiar features.

Gender	*n*	Mean	Std. dev.
Males	27	23.8	9.58
Females	24	29.8	9.66

Table 2
Males versus females' use of taught features.

Gender	*n*	Mean	Std. dev.
Males	27	123.41	68.27
Females	24	87.54	47.67

Table 3
Males versus females' use of untaught features.

Gender	*n*	Num who used	Pct who used (%)
Males	27	22	81.5
Females	24	11	45.8

Note that editing formulas is the *only* way to introduce bugs in a spreadsheet. Thus, it stands to reason that if females had spent more time with the unfamiliar debugging features and less with formula editing, then they would have had fewer chances to introduce bugs.

To address RQ3, we investigated whether self-efficacy played a role in participants' feature usage. We analyzed the pre-self-efficacy questionnaires by summing each participant's answers to all 10 questions and found that females had significantly lower pre-self-efficacy scores than males (Mann–Whitney: $U = 181$, tied $p < .018$). Fig. 2 depicts this difference.

For females, self-efficacy was a good predictor of feature usage. In particular, self-efficacy significantly predicted the final percent testedness for females (linear regression: $F(1, 22) = 4.52$, $\beta = 2.09$, $R^2 = .177$, $p < .046$) (but not for males). Fig. 3 depicts the relationships by gender. Percent testedness was a good measure of feature usage because it increased only if the participant strategically checked off values, indicating deliberate feature use. Spreadsheet testedness further predicted the number of bugs fixed by both males and females (linear regression: males: $F(1, 25) = 16.60$, $\beta = .632$, $R^2 = .399$, $p < .0004$; females: $F(1, 22) = 6.818$, $\beta = .486$, $R^2 = .237$, $p < .016$), suggesting important ties between self-efficacy, feature use, and success.

The results of Study #1 suggest that designers of problem-solving software should look for new ways to encourage users to consider relevant features they have not tried before. The (predominantly female) participants who favored the formula editor may have been drawn to the feature's familiarity. Since many of the users who favored the familiar formula editor had low self-efficacy, possibilities include finding ways to reinforce users' confidence in their success or finding ways to suggest that high certainty is not important in the use of particular features. We will discuss techniques that leverage these ideas in Section 4.

3.2. Study #2: tinkering and self-efficacy

Researchers in education (Rowe, 1978) and other domains (Martocchio and Webster, 1992; Webster and Martocchio, 1993) have found tinkering (playful exploration) in an environment to be beneficial for learning. Education research further suggests that gender may influence a person's willingness to tinker; for instance, males reportedly tinker more frequently than females (e.g., Jones et al., 2000). However, such differences have not previously been investigated in problem-solving software.

To understand the relationship between gender and tinkering in problem-solving software (RQ2), we empirically studied the tinkering behavior of males and females in the context of testing and debugging spreadsheets. Building upon Study #1, Study #2 also investigated the relationship between self-efficacy and tinkering (RQ3). We thus used a 2-by-2 between-subjects design with two factors: gender and spreadsheet environment. The two

Table 4
Mean (standard deviation) performance of males and females on bugs fixed and new bugs introduced that still remained at the end of the task.

Gender	*n*	Seeded bugs fixed (10 possible)		New bugs introduced		Num who introduced bugs
		Mean	Std. dev.	Mean	Std. dev.	
Males	27	5.815	2.167	0.111	0.424	2
Females	24	5.667	2.014	0.583	0.974	9

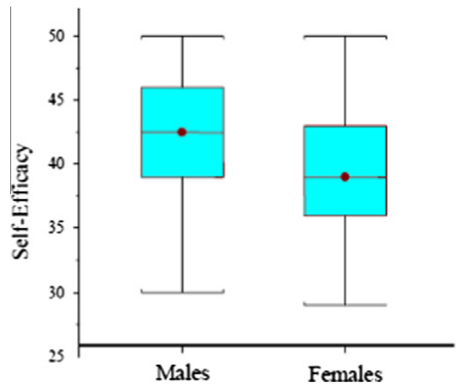

Fig. 2. Males' and females' pre-session self-efficacy (maximum was 50). The center line of each box represents the median score. The boxes show the ranges encompassed by 50% of the scores of each gender. The whiskers extending above and below the boxes show the remaining upper and lower 25%.

variants of the spreadsheet environment were (1) a *low-cost* version and (2) a *high-support* version. The low-cost version encouraged tinkering for users by minimizing the cost of manipulating features (e.g., in terms of number of clicks). The high-support version provided greater support for debugging, but also increased the cost of tinkering. Comparing males and females' tinkering in the low-cost environment versus the high-support environment enabled us to observe the influence that feature design can have on tinkering.

3.2.1. Low-cost and high-support spreadsheet environments

The low-cost variant of Forms/3 is the version described in Section 3.1.1. Its acting/tinkering cost was low: it required only a single action to toggle (or undo) a testing decision for cells (left-click for checkmark and right-click for X-mark), and it kept tooltip explanations as brief as possible.

The high-support variant emphasized support for the debugging features, but also increased the cost of acting/tinkering. The X-mark/checkmark box on each cell now enabled users to denote "maybe" values, but at a cost of two clicks (Fig. 4). Expandable tooltip explanations provided information about the current state of the spreadsheet and possible actions the user might take (Fig. 5). A "Help Me Test" feature could recommend test inputs to improve coverage of untested logic in the spreadsheet.

3.2.2. Participants and procedures

Participants comprised 36 males and 40 females with spreadsheet experience and minimal programming experience. They were recruited from a university community, had a variety of educational backgrounds, and ranged in age from under 20 to older than 60. Prior to the main study session, participants completed a pre-experiment questionnaire that collected background data on GPA, spreadsheet experience, and programming experience, and also measured pre-self-efficacy. We randomly assigned the males and females each into the two treatment groups (low-cost and high-support), using the background data only as needed to balance the groups. The low-cost group comprised 20 males and 17 females, and the high-support group comprised 16 males and 23 females. The groups showed no significant differences in the background data.

The main study session began with a 35-min hands-on tutorial to familiarize participants with the appropriate version of Forms/3. Next, participants performed the same two tasks as in Study #1 (randomly ordered): debugging the Gradebook and Payroll spreadsheets. Participants had 22 min and 35 min to work on Gradebook and Payroll tasks, respectively. After each task, participants filled out questionnaires regarding how they perceived their performance. Finally, participants completed a post-experiment questionnaire that measured post-self-efficacy and comprehension of the debugging features.

3.2.3. Results

To address RQ2, we analyzed whether the relationship between tinkering and performance on task was the same for females as for males. In educational settings, such exploration has been encouraged for improved performance (Rowe, 1978). Although our context was different, we expected the trend to hold—that is, we expected that participants who tinkered more would exhibit better performance.

To measure tinkering, we operationally defined a *tinkering instance* as turning a feature on and then immediately turning it off, e.g., placing a checkmark on a cell and then removing the

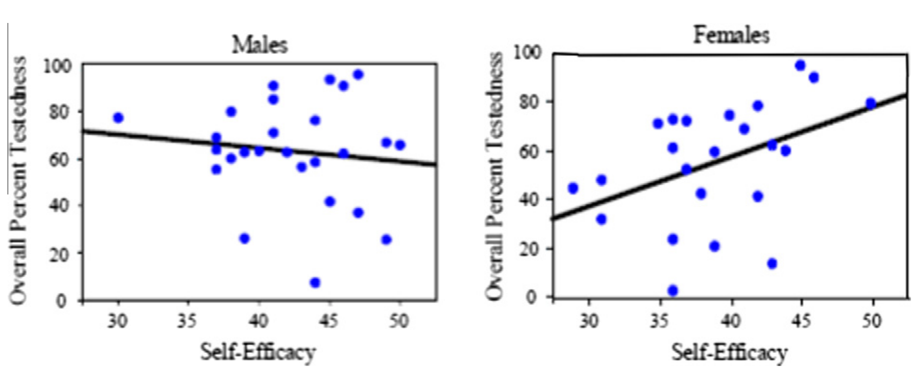

Fig. 3. Self-efficacy as a predictor of final spreadsheet testedness. The regression lines show the females' positive relationship of self-efficacy to spreadsheet testedness. For males, the relationship was not significant.

Fig. 4. Clicking on the decision box provides four decision options. The user can then click (from left to right): it is wrong, it seems wrong, it seems right, it is right.

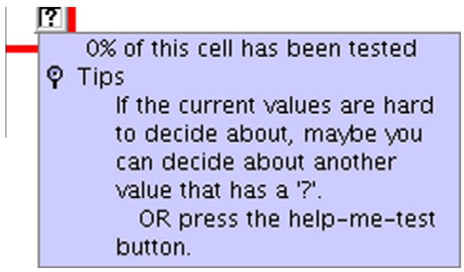

Fig. 5. From the non-expanded tool tip (the first line) a user can click on the tips expander to get the rest of the tool tip—which remains on the screen until dismissed.

checkmark. A participant's *tinkering frequency* for a task comprised the total number of tinkering instances. A *tinkering episode* comprised a sequence of one or more tinkering instances, terminated by a cell edit or the end of the task. A participant's episode count served as a measure of his or her consistent usage of tinkering. Finally, a participant's *tinkering rate* averaged his or her tinkering frequency per episode and was interpreted as commitment to tinkering within episodes. The first three rows of Table 5 summarize the tinkering results.

We considered three measures of performance: bugs fixed, percent testedness, and understanding of debugging features (as measured on the post-experiment questionnaire). The last three lines of Table 5 summarize the results for these task outcomes.

Correlating tinkering with performance, we were surprised to find that the relationship for females was essentially *the opposite* of the relationship for males. Tinkering positively predicted debugging effectiveness for females, whereas it *negatively* predicted debugging effectiveness for males. For females, tinkering episodes significantly predicted percent testedness (linear regression: $F(1, 38) = 4.63$, $R^2 = .11$, $p < .05$). In contrast, no measure of tinkering predicted percent testedness for males; however, tinkering rate negatively predicted bugs fixed for them (linear regression: $F(1, 34) = 8.04$, $R^2 = .19$, $p < .01$). For females, tinkering episodes also predicted understanding (linear regression, frequency: $F(1, 38) = 4.56$, $R^2 = .11$, $p < .05$; episodes: $F(1, 38) = 4.44$, $R^2 = .10$, $p < .05$). In contrast, no measure of tinkering correlated with males' understanding of the features.

To understand why males' tinkering correlated negatively with their performance, we considered two types of tinkering: exploratory and repeated. *Repeated tinkering instances* are the number of tinkering instances in a sequence of two or more consecutive tinkering instances on the same feature and the same cell. For example, turning the arrows for a cell on and then immediately back off again three times in a row is three repeated tinkering instances. Hence, the repetitions can only repeat information already revealed by the previous tinkering instances. *Exploratory tinkering instances*, which we define as the difference between total tinkering instances and repeated tinkering instances, potentially can impart new information.

Table 5
Results for tinkering and task outcomes: means (standard deviations).

Dependent variable	Low cost		High support	
	Male	Female	Male	Female
Tinkering frequency	27.3 (27.0)	13.7 (9.2)	10.1 (8.0)	11.7 (9.0)
Tinkering episodes	9.5 (5.8)	7.9 (4.0)	5.1 (3.1)	6.0 (3.7)
Tinkering rate	2.6 (1.9)	1.7 (0.7)	1.8 (0.7)	1.9 (0.9)
Bugs fixed	5.9 (3.1)	6.3 (3.1)	6.5 (2)	5.3 (3.2)
Percent testedness	61.9 (25.8)	67.9 (20.3)	63.2 (17.2)	65.3 (22.2)
Understanding of features	7.4 (2.8)	7.2 (2.3)	7.3 (3.1)	7.5 (2.7)

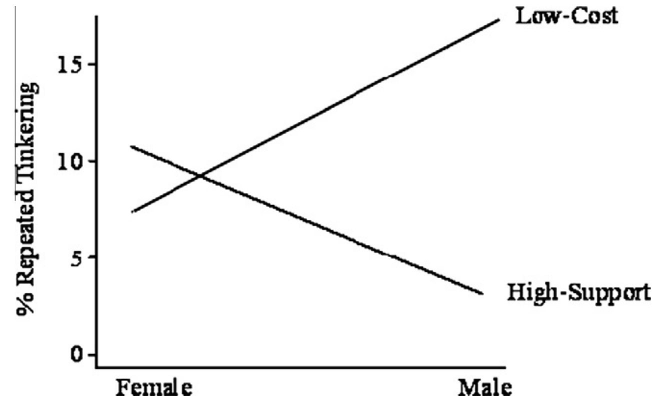

Fig. 6. This interaction plot (of means only) illustrates the gender × treatment interaction in percentage of repeated tinkering.

An increase in female exploratory tinkering (which accounted for 91% of their overall tinkering) was a significant predictor of increased understanding (linear regression: $F(1, 38) = 4.61$, $R^2 = .11$, $p < .05$). In contrast, males' exploratory tinkering oddly was *not* statistically predictive of understanding or of any of the effectiveness measures.

As Fig. 6 shows, repeated tinkering instances accounted for a significantly greater proportion of the low-cost males' tinkering than in any other group, with a significant effect of interaction between gender and treatment (ANOVA: $F(1, 72) = 5.82$, $p < .05$). In fact, nearly 17% of the low-cost males' tinkering instances were of the repeated type, almost twice as many as for the next highest group.

The high-support treatment, with its increased tinkering cost, not only reduced males' tinkering, but also *selectively* reduced their ineffective repeated type of tinkering. Because it did not affect the females' tinkering effectiveness, this approach appears to be the better of the two treatments for both genders, albeit for different reasons.

Still, the different types of tinkering alone do not explain the inverse relationship between tinkering and performance for males. Education research suggests a second reason: the males may not have reflected enough. Research has shown that when students have a "wait-time" of 3 s or more to process a classroom response, their critical thinking improves (Rowe, 1978). To see whether such wait-times also apply in this context, we likewise defined *pauses* as three or more seconds of inactivity after a user action.[1]

Overall, females had significantly more pauses than males (ANOVA: $M_{Male} = 109$, $M_{Female} = 220$, $F(1, 74) = 4.22$, $p < .05$). Thus, males did not take time that might have been used to reflect upon the feedback from their actions as often as the females did. These pauses mattered; participants who paused long enough to reflect on the environment's responses showed evidence of understanding the features better and used them more effectively. For both genders, more frequent pauses were predictive of greater effectiveness

[1] We considered pauses after *any* action (not just tinkering instances), because every action provides feedback.

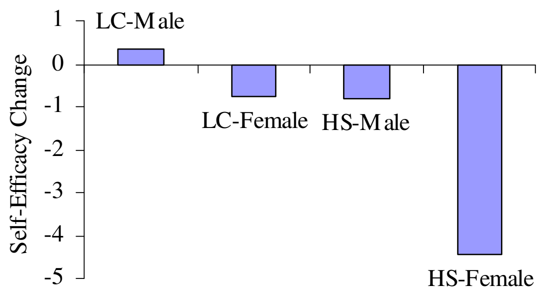

Fig. 7. The change from pre-self-efficacy to post-self-efficacy. The high-support females' drop was significant.

as measured by bugs fixed (linear regression: $F(1, 74) = 5.36$, $R^2 = .07$, $p < .05$), final percent testedness (linear regression: $F(1, 74) = 31.3$, $R^2 = .30$, $p < .01$), and understanding of features (linear regression: $F(1, 74) = 14.11$, $R^2 = .16$, $p < .01$). Males' tendency to tinker more appears to be useful only when they make regular use of pauses. In our study, their tendency to pause too little may have interfered with the potential benefits of tinkering.

To address RQ3, we considered changes in participants' self-efficacy before and after they completed the tasks. We expected that increased tinkering would increase females' post-self-efficacy, as would the set of features in the high-support treatment. Surprisingly, females using the high-support treatment experienced a dramatic fall in self-efficacy (Fig. 7) that was significant (paired t test: DF = 22, $t = 3.19$, $p < .01$). The other groups showed little to no change in their self-efficacy between pre-self-efficacy and post-self-efficacy. The relationships between tinkering and post-self-efficacy were also surprising. For high-support females, the rate of tinkering per episode was predictive of the *drop* in self-efficacy reported above (linear regression: $F(1, 21) = 6.32$, $R^2 = .23$, $p < .05$).

One possible reason for the high-support females' extreme drop in self-efficacy is that they did not perceive tinkering as helpful for understanding how their debugging environment worked. Therefore, the more they tinkered, the more they reinforced this perception of their inability to understand what was happening in the environment. Taken in combination with the other tinkering/effectiveness results of this study, it appears that the tinkering issue for females is complex. Females were better than the males at consistently extracting problem-solving benefits from tinkering, but were worse than the males at maintaining their self-efficacy levels.

Since tinkering was not a straightforward win for females, one implication for products in which there is an underlying assumption that adopting new features will start by people tinkering with them is that designers could provide alternative ways to enable users to become comfortable with new features, such as by tutorial snippets. Additional design remedy possibilities include finding ways to encourage pauses for reflection during exploratory tinkering while reducing the incidence of repetitive tinkering, which our results suggest are particularly important for males. We consider these possibilities in the next section.

4. Study #3: gender-pluralist design

Our third study investigated whether it is possible to close the gender gaps revealed in Studies #1 and #2 in ways that do not penalize either gender (RQ4). We introduced variants of the testing/debugging features described earlier and investigated their influence on the feature use, effectiveness, and confidence of male and female problem-solvers.

The experiment was a 2-by-2 between-subjects design with two factors: gender and version of the spreadsheet environment (treatment versus control). The treatment and control groups both

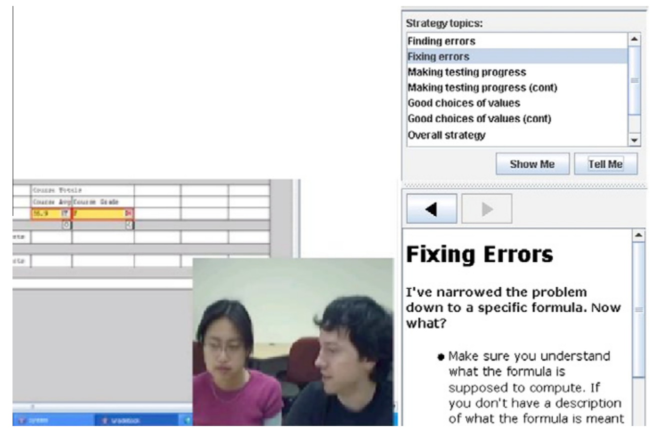

Fig. 8. The strategy-explanation feature includes video snippets of two people working on a spreadsheet (left) and hypertext content (right).

used the Forms/3 environment, but the treatment group's version contained the two additional features: "maybe" marks (described in Section 3.2.1) and a strategy-explanation feature.

4.1. Gender-conscious spreadsheet environment

The design of the treatment features was motivated by both theoretical and empirical research relating to gender differences in problem-solving environments.

We designed a strategy-explanation feature with video-snippet and hypertext media (Fig. 8), drawing inspiration from research reporting that tutorial materials benefit females' performance in software development (Kelleher et al., 2007; Subrahmaniyan et al., 2007). The explanations emphasized strategic debugging in the software environment, not feature-by-feature usage. The videos were developed according to self-efficacy theory, which states that one way for people to gain confidence is through vicarious experience, i.e., watching other people like themselves succeed at a similar task. They aimed primarily to help low self-efficacy females gain confidence. Each video snippet involved a male/female pair and showed the female struggle and then succeed with a debugging situation. Also taking into account the attention investment model, we began each video with explicit lead-ins to make explicit the benefit of viewing the video. We labeled the video snippets with their viewing times (usually a minute or less) to align users' perceived costs with actual costs of viewing the videos. We also included hypertext variants of the same text as the videos to accommodate users and situations for which text was perceived as less costly or otherwise preferable. We combined this feature with "maybe" marks. ("Maybe" marks had also been used in Study #2, but were not explicitly evaluated in that study.) "Maybe" marks were intended to beckon to low self-efficacy users by suggesting that being certain was not required to be "qualified" to pare down possible locations of spreadsheet errors.

4.2. Participants and procedures

Participants comprised 67 male and 65 female undergraduate students from a variety of majors, about half of whom were science/engineering majors (e.g., mechanical and nuclear engineering) and about half non-technical majors (e.g., Spanish, theater arts, business administration). No computer science students were allowed to participate. Prior to the experiment, we gathered background data that included GPA, age, year in school, programming experience, spreadsheet experience, and self-efficacy, and we randomly assigned participants to treatment groups (balanced by

Table 6
Tinkering and use of debugging features. Thirty-five treatment females, 30 control females, 34 treatment males, 33 control males.

	TF versus CF	TM versus CM	Test
Playful checkmarks/minute	TF more: $F(2, 62) = 2.779$, $p < .022^{*}$	Not significant	ANCOVA
Playful X-marks/minute	TF more: $Z = -2.47$, $p < .014^{*}$	TM more: $Z = -2.02$, $p < .044^{*}$	Wilcoxon
Lasting checkmarks/minute	Not significant	Not significant	ANCOVA
Lasting X-marks/minute	TF more: $Z = -2.70$, $p < .070^{\sim}$	TM more: $Z = -2.09$, $p < .037^{*}$	Wilcoxon
	TF versus TM		
Low-conf marks/minute	TF more: $F(1, 67) = 3.977$, $p < .0503^{\sim}$		ANOVA

* $p < .05$.
** $p < .01$.
$^{\sim}$ $p < .1$.

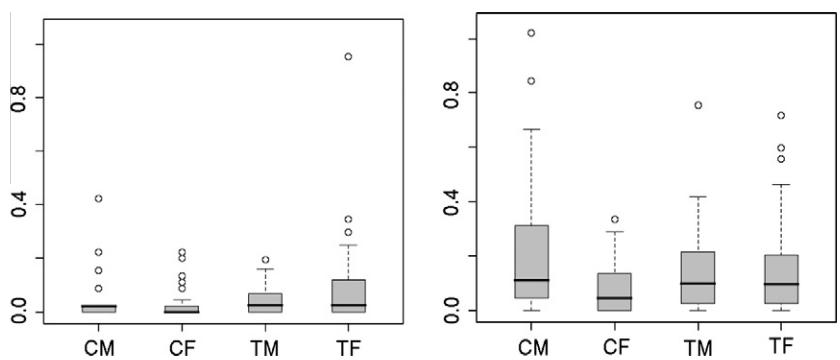

Fig. 9. Tinkering with X-marks (left) and checkmarks (right), in marks applied per debugging minute. Note the gender gaps between the control females' and males' medians. These gaps disappear in the treatment group.

gender). A post-hoc analysis showed no significant differences between genders or treatment groups on any of these background data except self-efficacy. (We handled the self-efficacy difference by taking it into account statistically, as detailed below.)

For the experiment, participants first completed a tutorial on their group's variant of Forms/3 (30 min). They then debugged the payroll spreadsheet from the first two studies (45-min limit). For this study, we seeded the spreadsheet with six bugs. To ensure treatment participants used the strategy-explanation feature, we interrupted them after 30 min and asked them to view either a video or hypertext explanation of their choosing. Finally, participants completed a questionnaire that asked them to rate the usefulness of features, to describe how parts of the software affected their confidence in fixing bugs (open-ended question), and to answer the self-efficacy questions again.

4.3. Results

Our analysis emphasized three comparison groups. First, we compared treatment females with control females to see if the treatment affected females' outcomes. Second, we compared treatment males with control males to see if the features influenced males' outcomes. Third, we compared treatment male/female gender gaps with control male/female gender gaps to see if the features affected the gender gap.

Although males fixed significantly more bugs than females in this study (females: $M = 2.88$, $SD = 1.75$; males: $M = 3.84$, $SD = 1.46$; ANOVA: $F(1, 130) = 15.67$, $p < .00013$), this is not surprising given that the first two studies revealed barriers to females' success in debugging spreadsheets. Complicating the situation, a self-efficacy background difference was present in this study. Thus, in making the comparisons, we statistically took into account potential confounds: gender/treatment group differences in *pre-self-efficacy*, and individual differences in *minutes* available for debugging. Females in the treatment group had significantly lower pre-self-efficacy than females in the control group (treatment females: $M = 36.82$, $SD = 5.22$; control females: $M = 40.13$, $SD = 4.28$; ANOVA: $F(1, 63) = 7.96$, $p < .0064$). Furthermore, control participants had more time to debug than the treatment participants because the latter spent time viewing explanations. All of our subsequent statistical tests accounted for these differences except where indicated otherwise. Taking these factors into account, our results showed that the feature changes helped to close the gender gap in *usage of debugging features*. Specifically, treatment females used both the checkmarks and X-marks significantly more than control females and closer to the males' level (Table 6; illustrated in Fig. 9) due to increased tinkering. We distinguished playful usage (e.g., quickly checking and unchecking a box) from lasting usage. We used ANCOVA (with pre-self-efficacy as a covariate) to test the difference in checkmark tinkering, whereas we used Wilcoxon rank-sum to test the difference for X-marks because the number of ties at zero made the distribution non-normal.

The increased feature usage of the treatment group is noteworthy because females who used the features more achieved greater success. Specifically, the total (playful plus lasting) number of checkmarks used per debugging minute (accounting for pre-self-efficacy) predicted the maximum percent testedness per debugging minute achieved by females in the control group (total checkmarks per debugging minute: $M = 0.59$, $SD = 0.43$; ANCOVA: $F(2, 27) = 18.04$, $p < .00001$) as well as in the treatment group (total checkmarks per debugging minute: $M = 0.63$, $SD = 0.53$; ANCOVA: $F(2, 32) = 31.11$, $p < .00001$). We used testedness as a measure of success because it was a significant factor (accounting for pre-self-efficacy) in the number of bugs fixed (maximum percent testedness: $M = 0.57$ (56.5% testedness), $SD = 0.22$; bugs fixed: $M = 3.36$, $SD = 1.65$; ANCOVA: $F(2, 129) = 8.88$, $p < .00024$). This experiment did not tease apart the effects of the nuanced ("maybe") judgments feature from the effects of the strategy explanations on success. We attempted to analyze the relationships between the number of minutes spent viewing explanations and measures of each participant's success. However, assessing ties to success involved the interaction of participants' use of strategies and their

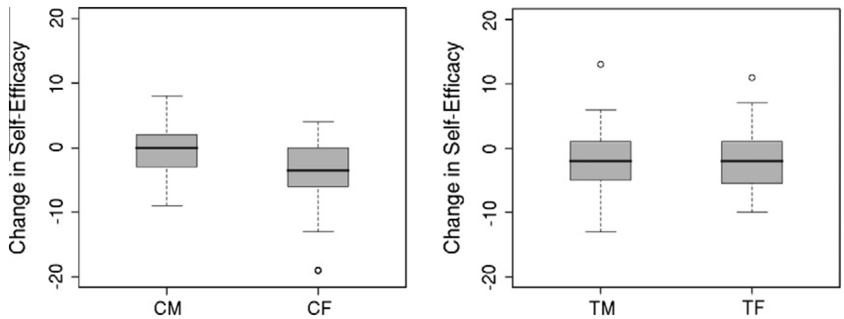

Fig. 10. Change in self-efficacy for (left) control participants and (right) treatment participants.

Table 7
Means, standard deviations, and results of linear regression tests comparing whether bugs fixed per debugging minute is predictive of post-self-efficacy for control females, control males, treatment females, and treatment males.

CF	CM	TF	TM
Bugs: $M = 0.067$, $SD = 0.042$	Bugs: $M = 0.092$, $SD = 0.026$	Bugs: $M = 0.065$, $SD = 0.039$	Bugs: $M = 0.084$, $SD = 0.040$
Post-SE: $M = 36.12$, $SD = 5.55$	Post-SE: $M = 39.50$, $SD = 4.70$	Post-SE: $M = 34.86$, $SD = 6.49$	Post-SE: $M = 39.12$, $SD = 5.51$
$F(1, 28) = 2.07$	$F(1, 31) = 6.92$	$F(1, 33) = 3.62$	$F(1, 32) = 6.33$
$\beta = 34.46$	$\beta = 76.79$	$\beta = 52.86$	$\beta = 56.49$
$R^2 = .069$	$R^2 = .18$	$R^2 = .099$	$R^2 = .17$
$p < .17$	$p < .014^{*}$	$p < .066^{\sim}$	$p < .018^{*}$

Table 8
Means, standard deviations, and results of Wilcoxon rank-sum tests of 35 treatment females' and 30 control females' responses regarding what influenced their confidence.

	CF	TF	Result
Positive statements overall	$M = 0.63$, $SD = 0.93$	$M = 1.26$, $SD = 1.20$	TF more: $Z = -2.29$, $p < .022^{*}$
Negative statements about experiment	$M = 0.40$, $SD = 0.50$	$M = 0.20$, $SD = 0.47$	TF less: $Z = 1.92$, $p < .055^{\sim}$
Positive statements about information	$M = 0.03$, $SD = 0.18$	$M = 0.37$, $SD = 0.65$	TF more: $Z = -2.70$. $p < .007^{**}$
Positive statements about features	$M = 0.10$, $SD = 0.31$	$M = 0.29$, $SD = 0.52$	TF more: $Z = -1.63$, $p < .103$

self-efficacy. These relationships were complex and non-linear, making interpretation difficult. However, although we were not able to separate the impacts of nuance from strategy explanations, we know that the combination was tied to quantifiable benefits for females.

Turning to self-efficacy, we hypothesized that the features would positively influence female self-efficacy, but not lead to inappropriately high self-efficacy. Low self-efficacy is only appropriate when it reflects an accurate assessment of the person's ability. Thus, *inappropriately* low (underconfidence) or high (overconfidence) self-efficacy was our target because they can lead to problems with persistence and strategy decisions (Bandura, 1986). Recall also that Studies #1 and #2 suggested that females' debugging outcomes were particularly susceptible to low self-efficacy.

Indeed, the results suggest that the features provided the desired benefits to self-efficacy. Although nearly all participants' self-efficacy decreased during the experiment (see Fig. 10), we found suggestive evidence that treatment females' decrease in self-efficacy was less than control females' (control females: $M = -4.05$, $SD = 5.00$; treatment females: $M = -1.91$, $SD = 5.14$; ANOVA: $F(1, 63) = 2.86$, $p < .096$). Furthermore, Table 7 shows that treatment females (as well as both male groups) displayed appropriate post-self-efficacy given their debugging performance. In contrast, we found no such evidence that control females had appropriate confidence.

Given the suggestive, but not overwhelming, evidence of positive effects on treatment females' self-efficacy, we triangulated these results with the participants' free-text responses on the post-questionnaire. In particular, we focused on questions that asked which parts of the software affected participants' confidence in their ability to fix bugs. To analyze the responses, we coded participant comments as positive or negative and as related to environmental conditions, user features, software feedback, information given, software usability, or experiment setup. Two researchers individually coded the same 20 participants' answers using this scheme, achieving a 91% agreement rate. Given the high agreement, a single researcher coded the remaining participants' answers.

Participant responses confirmed that the treatment environment fostered females' confidence better than did the control environment. Treatment females said significantly more positive things than control females about the features and information's effects on their confidence. Table 8 summarizes the analysis results for females.

Additionally, the results suggest that the features did not hinder treatment males. Overall, treatment males' comments were not significantly more positive or negative than control males'. Moreover, treatment males spoke more positively about information than their control counterparts did (Wilcoxon rank-sum test: $Z = -2.21$, $n = 67$, $p < .028$).

5. Discussion: generality

The three studies reported here together define our "core" results. However, they were all done in the context of one research prototype, allowing controlled manipulation of features of the software at a fine-grained level, but raising the issue of generality: whether our results would also apply to other software environments, other tasks, and other populations. Therefore, we ran a number of studies in other problem-solving contexts to assess the extent to which these results can be replicated in other environments and with other populations. We briefly summarize those results here.

In our first such study, we closely replicated the procedure of Study #1, but in the context of Excel (Beckwith et al., 2007). We generalized the population also, incorporating a population of Seattle-area experienced Excel practitioners rather than college students as in Study #1. We found that several of the results from Study #1 generalized to Excel. First, females' self-efficacy predicted task success, but the same did not hold true for the males. Second, low self-efficacy females relied more heavily on the "familiar" type of features, particularly value edits: a relationship that did not hold for males. Third, these results could not be attributed to females being better judges of their weaknesses. Females' comprehension of the software features was no different than the males' and was not predicted by self-efficacy. This was the first study in a commercial environment, but the fourth study in total (including the three reported in this paper) in which we found that the effects of self-efficacy playing out differently for males and females.

Moving beyond spreadsheets, we have also investigated gender differences in the context of problem solving about the learned behaviors from machine learning systems. For example, email spam filters and recommender systems learn rules of computational behavior particular to one end user and these learned rules (which we term "learned programs") sometimes require adjustment ("debugging"). We performed two studies in the context of an email sorter that learns how to help users sort emails into folders. We found gender differences in both of them. In our first study in this domain (Stumpf et al., 2008), males and females handled the same number of messages, but females spent significantly more time doing so. This was because females used the provided features more comprehensively (recall the selectivity hypothesis (Meyers-Levy, 1989), notifying the system of significantly more keywords than males did. Our second study in this domain (Kulesza et al., 2009) found that females had significantly lower self-efficacy, encountered more "selection" barriers (knowing what to do, but not which feature to use) and "design" barriers (cognitive difficulties, separate from the features) with the environment's debugging features, and responded differently than males did in their attempts to overcome those barriers. For example, when females encountered a selection barrier, their next barrier was more likely than the males' to be a selection barrier. Thus, although these two studies did not consider all the issues of the spreadsheet studies reported in this paper, the questions that they did consider produced results consistent with those found in the spreadsheet studies.

Turning to the paradigm of web automation, in a study of end-user mashup programmers (Cao et al., 2010), as with the above studies, females had lower self-efficacy and focused their efforts on familiar webservice features (versus unfamiliar webservice features) significantly more than the males did. Rosson et al.'s study of web developers also showed suggestive gender differences in the use of novel web-based database features that are consistent with these findings (Rosson et al., 2007).

Finally, we performed a multi-study that generalized beyond end-user problem solvers to a wide range of problem solvers ranging from administrators to professional programmers (Burnett et al., 2010). Specifically, in a gender-based analysis of almost 3000 participants from multiple studies' data at a large software company, we considered the same three research questions as RQ1–RQ3 in the core studies of this paper, but across multiple platforms and populations. The multi-study covered a study of technical problem-solving practices of multiple populations (from administrators to professional software developers), two studies of hobbyist programmers using Visual Studio Express, and two studies of professional software developers using Visual Studio as well as a variety of other platforms. As with the three studies reported in this paper, we found significant gender differences across all programming environments and populations as to which features males and females elected to use, as to males' and females' willingness to tinker and explore, and between males' or females' technical problem-solving confidence. Furthermore, as with the other studies reported in this paper, the confidence differences were not the sole explanation for the differences in feature usage and tinkering.

In summary, we have incorporated gender analyses in a variety of studies spanning multiple software environments and populations, and these studies have all found gender differences similar to those reported in this paper: in feature usage, in tinkering, and in how confidence plays out for males versus females.

6. Conclusions

In this paper, we have reported three core studies investigating gender differences in problem-solving environments. We also point to evidence that many of these results generalize, with similar results spanning ten other studies, seven problem-solving platforms, and thousands of participants ranging from end users to professional programmers. Our results were as follows:

RQ1: There were significant gender differences as to *which features* males and females use in problem-solving software.
RQ2: There were significant gender differences as to males' and females' willingness to *tinker and explore*.
RQ3: Although there were significant differences between males' or females' technical problem-solving *confidence*, these differences clearly were not the sole explanation for the differences in feature usage and tinkering.
RQ4: Early results suggest that it is possible to design problem-solving software in a way that takes gender differences into account, such that it overcomes gender gaps *without* penalizing either gender.

Our findings underscore the importance of taking gender differences into account when designing problem-solving software. For instance, software that assumes that users will discover advanced features solely through tinkering and exploring will likely marginalize females. Fortunately, our RQ4 results suggest that pluralist designs need not sacrifice the needs of one gender to serve the other: researchers have shown that taking gender differences into account in designing software features can benefit *both* genders. As an additional example, Tan et al. showed that displaying optical flow cues helped close a gender gap in virtual world navigation while benefiting both females and males (Tan et al., 2003). Furthermore, education researchers found that pair programming, which was expected to help female computer-science students, not only reduced the gender gap but also increased success and reduced attrition among both male and female students (Berenson et al., 2004; McDowell et al., 2003).

Finally, these gender differences do not suggest that males are somehow "better" software users than females. For instance, using unfamiliar features is not always better than using familiar ones, and tinkering is not always productive. Furthermore, although gender differences in feature usage, willingness to tinker, and confidence are all implicated in our results, an individual male or female generally does not bear every trait associated with his or her gender. Although our statistical analyses revealed gender differences, a portion of the males still exhibited traits associated with the majority of females, and vice versa. Thus, informing the design of problem-solving software based on the differences revealed by our investigation need not penalize either gender—doing so can help everyone.

Acknowledgment

This work was supported in part by NSF 0917366 and 0324844.

References

Bandura, A., 1986. Social Foundations of Thought and Action: A Social Cognitive Theory. Prentice-Hall.

Bardzell, S., Feminist, H.C.I., 2010. Taking stock and outlining an agenda for design. In: ACM Conf. Human Factors in Computing Systems, pp. 1301–1310.

Barke, R., Jenkins-Smith, H., Slovic, P., 1997. Risk perceptions of men and women scientists. Social Science Quarterly 78 (1), 167–176.

Beckwith, L., Burnett, M., Wiedenbeck, S., Cook, C., Sorte, S., Hastings, M., 2005. Effectiveness of end-user debugging software features: are there gender issues? In: ACM Conf. Human Factors in Computing Systems, pp. 869–878.

Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., Cook, C., 2006. Tinkering and gender in end-user programmers' debugging. In: ACM Conf. Human Factors in Computing Systems, pp. 231–240.

Beckwith, L., Inman, D., Rector, K., Burnett, M., 2007. On to the real world: gender and self-efficacy in Excel, In: IEEE Symp. Visual Languages and Human-Centric Computing, pp. 119–126.

Berenson, S., Slaten, K., Williams, L., Ho, C.-W., 2004. Voices of women in a software engineering course: reflections on collaboration. Journal on Educational Resources in Computing 4 (1).

Beyer, S., Rynes, K., Perrault, J., Hay, K., Haller, S., 2003. Gender Differences in Computer Science Students. SIGCSE: Special Interest Group on Computer Science Education, pp. 49–53.

Blackwell, A., 2002. First steps in programming: a rationale for attention investment models. In: IEEE Symp. Human-Centric Computing Languages and Environments, pp. 2–10.

Blais, A.-R., Weber, E., 2001. Domain-specificity and gender differences in decision making. Risk Decision and Policy 6, 47–69.

Brunner, C., Bennett, D., Honey, M., 1998. Girl Games and Technological Desire. From Barbie to Mortal Kombat: Gender and Computer Games. MIT Press, pp. 72–88.

Burnett, M., Cook, C., Rothermel, G., 2004. End-user software engineering. Communications of the ACM 47 (9), 53–58.

Burnett, M., Fleming, S., Iqbal, S., Venolia, G., Rajaram, V., Farooq, U., Grigoreanu, V., Czerwinski, M., 2010. Gender differences and programming environments: across programming populations. In: IEEE Int'l Symp. Empirical Software Engineering and Measurement, September 2010.

Busch, T., 1995. Gender differences in self-efficacy and attitudes toward computers. Journal of Educational Computing Research 12 (2), 147–158.

Byrnes, J., Miller, D., Schafer, W., 1999. Gender differences in risk taking: a meta-analysis. Psychological Bulletin 125, 367–383.

Cao, J., Rector, K., Park, T., Fleming, S., Burnett, M., Wiedenbeck, S., 2010. A debugging perspective on end-user mashup programming. In: IEEE Symp. Visual Languages and Human-Centric Computing, September 2010.

Colley, A., Comber, C., 2003. Age and gender differences in computer use and attitudes among secondary school students: what has changed? Educational Research 45 (2), 155–165.

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

Durndell, A., Haag, Z., 2002. 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, 521–535.

Finucane, M., Slovic, P., Merz, C.-K., Flynn, J., Satterfield, T., 2000. Gender, race and perceived risk: the white male effect. Health, Risk and Society 2 (2), 159–172.

Grigoreanu, V., Cao, J., Kulesza, T., Bogart, C., Rector, K., Burnett, M., Wiedenbeck, S., 2008. Can feature design reduce the gender gap in end-user software development environments? In: IEEE Symp. Visual Languages and Human-Centric Computing, pp. 149–156.

Hudgens, G., Fatkin, L., 2001. Sex differences in risk taking: repeated sessions on a computer-simulated task. The Journal of Psychology 119 (3), 197–206.

Jiankoplos, N., Bernasek, A., 1998. Are women more risk averse? Economic Inquiry 36 (4), 620–630.

Jones, M., Brader-Araje, L., Carboni, L., Carter, G., Rua, M., Banilower, E., Hatch, H., 2000. Tool time: gender and students' use of tools, control, and authority. Journal of Research in Science Teaching 37 (8), 760–783.

Kelleher, C., Pausch, R., Kiesler, S., 2007. Storytelling Alice motivates middle school girls to learn computer programming. In: ACM Conf. Human Factors in Computing Systems, pp. 1455–1464.

Kulesza, T., Wong, W.-K., Stumpf, S., Perona, S., White, R., Burnett, M., Oberst, I., Ko, A., 2009. Fixing the program my computer learned: barriers for end users, challenges for the machine. In: ACM Conf. Intelligent User Interfaces, pp. 187–196.

Ljungblad, S., Holmquist, L., 2007. Transfer scenarios: grounding innovation with marginal practices. In: ACM Conf. Human Factors in Computing Systems, pp. 737–746.

Loewenstein, G., 1994. The psychology of curiosity: a review and reinterpretation. Psychology Bulletin 116 (1), 75–98.

Margolis, J., Fisher, A., 2003. Unlocking the Clubhouse. MIT Press.

Martocchio, J., Webster, J., 1992. Effects of feedback and playfulness on performance in microcomputer software training. Personnel Psychology 45, 553–578.

McDowell, C., Werner, L., Bullock, H.E., Fernald, J., 2003. The impact of pair programming on student performance, perception and persistence. In: 25th Int'l Conf. Software Engineering.

Meyers-Levy, J., 1989. Gender Differences in Information Processing: A Selectivity Interpretation. Cognitive and Affective Responses to Advertising. Lexington Books, pp. 219–260.

Panko, R., Aurigemma, S., 2010. Revising the Panko-Halverson Taxonomy of Spreadsheet Errors, Decision Support Systems.

Rosson, M.B., Sinha, H., Bhattacharya, M., Zhao, D., 2007. Design planning in end-user web development. In: IEEE Symp. Visual Languages and Human-Centric Computing, pp. 189–196.

Rothermel, G., Burnett, M., Li, L., DuPuis, C., Sheretov, S., 2001. A methodology for testing spreadsheets. ACM Transactions on Software Engineering and Methodology 10 (1), 110–147.

Rowe, M., 1978. Teaching Science as Continuous Inquiry. McGraw-Hill.

Stumpf, S., Sullivan, E., Fitzhenry, E., Oberst, I., Wong, W.-K., Burnett, M., 2008. Integrating rich user feedback into intelligent user interfaces. In: ACM Conf. Intelligent User Interfaces, pp. 50–59.

Subrahmaniyan, N., Kissinger, C., Rector, K., Inman, D., Kaplan, J., Beckwith, L., Burnett, M., 2007. Explaining debugging strategies to end-user programmers. In: IEEE Symp. Visual Languages and Human-Centric Computing, pp. 127–134.

Tan, D., Czerwinski, M., Robertson, G., 2003. Women go with the (optical) flow. In: ACM Conf. Human Factors in Computing Systems, pp. 209–215.

Webster, J., Martocchio, J., 1993. Turning work into play: implications for microcomputer software training. Journal of Management 19 (1), 127–146.

Zeldin, A., Pajares, F., 2000. Against the odds: self-efficacy beliefs of women in mathematical, scientific, and technological careers. American Educational Research Journal 37, 215–246.