Gender HCI: What About the Software?

Laura Beckwith, Margaret Burnett, and Valentina Grigoreanu
Oregon State University

Susan Wiedenbeck
Drexel University

Studies building upon theories and research from several domains investigate how gender differences interact with software.

In high school, Ashley planned to become a graphic designer. However, when attempting to tackle flash programming in the graphics design course, frustration set in. Although graphic designers use Flash, the original WYSIWYG programming style has largely given way to a Java-like language aimed at software developers. Then the need to do Web programming arose as well. Ashley decided that learning Flash and Web programming was too great a barrier, and instead majored in art.

What were the real causes of Ashley's difficulties? It's possible that a problem-solving style, learning style, or lack of confidence made learning these software tools seem more formidable than it would to someone else.

Gender differences in these and other domains, such as psychology, marketing, and neuroscience, strongly suggest that females process information and solve problems in different ways than males do.[1,2] We have been investigating whether software design should take these sorts of gender differences into account.

Recognition of the possibility that software could erect barriers to females has only recently emerged.[1,3] There is, however, research on other relationships between gender and computers, such as on gender differences with computer display hardware,[4] gender-oriented marketing strategies for Web-based shopping,[5] and computer game software content.[6] Other work has focused on factors affecting females' interest in computer science as a career choice, including the academic climate, educational strategies, human resource management, education, and social and cultural factors[7-9]—just about everything except how the software works.

We use the term *gender HCI* (human-computer interaction) to refer to research into how software relates to gender differences. Our particular focus is on how gender-neutral software works, not on gender-oriented content. Specifically, we have concentrated on software aimed at supporting everyday users doing problem solving. Examples of this sort of software include spreadsheets, CAD systems, macro builders, educational software authoring systems, and media authoring systems.

The issues involved in supporting both genders' use of problem-solving software are important for two reasons. First, such experiences could impact the pipeline of women in information technology, essentially closing it off. If a female's early experience with software that is supposed to support her problem-solving efforts is negative and discouraging, how likely is she to eventually choose a career in information technology?

Second, the productivity of end-user problem solvers, regardless of whether they might someday become software developers, also raises concerns. If they fail to take relevant gender differences into account in the design of problem-solving software, developers can introduce barriers into the software that interfere with the success of half the population the software is intended to support.

INVESTIGATIVE METHOD

Our method for conducting this investigation consists of four steps:

1. Draw from theory and previous empirical gender-difference work from other domains—such as computer confidence, perceived risk, information processing, computer gaming, and technology adoption models—to hypothesize gender issues and their causes that could arise from gender-based differences in the use of problem-solving software.
2. Use empirical methods to investigate whether these issues do actually arise in problem-solving software.
3. Use the results of the first two steps along with qualitative empirical

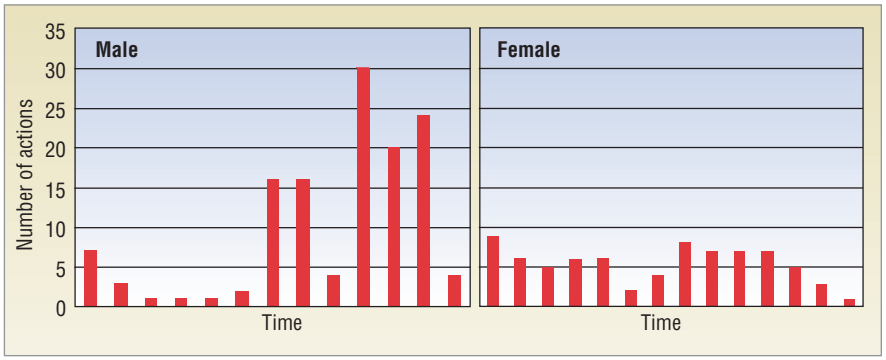

Figure 1. Activity profiles of one male and one female in a problem-solving software environment show that the male performed more actions and used more features than the female did.

work involving low-cost prototyping to derive and refine approaches to address the issues.
4. Use quantitative empirical methods to evaluate the approaches' effectiveness.

Several of the hypotheses we developed in the first step of our investigative method predict that in a software environment with problem-solving features, gender differences will have a significant impact on the adoption and use of these features, due at least in part to differences related to risk perception and confidence.

Gender differences: Is there a case?

Gender differences regarding computer confidence have been widely studied, revealing that females—both computer science majors and end users—have lower confidence than males in their computer-related abilities.[1,8] Of particular pertinence is the concept of *self-efficacy*,[10] a person's judgment about his or her ability to carry out a specific course of action to achieve a goal.

According to self-efficacy theory, this trait is critical in problem solving because it influences the use of cognitive strategies, the amount of effort put forth, the coping strategies adopted in the face of obstacles, and the final performance outcome. This, combined with other research that found females more risk-averse than males, implies that females' low self-efficacy regarding computing tasks will render them less willing to explore and adopt new features.

We began our investigation into gender's possible impact on software-based problem-solving activities in Experiment 1, a qualitative reanalysis of feature usage from data collected in a prior experiment unrelated to gender. We recruited this study's participants, all noncomputer science students, primarily from the business school.

We found that differences in feature activity showed a striking tendency to align by gender. For example, Figure 1 shows the activity profiles of a male and a female user, both fairly representative of other experiment participants of the same gender. As the figure shows, overall, the males' amounts and types of feature usage differed noticeably from those of the females.

Feature acceptance

Following up on Experiment 1, Experiment 2 was a quantitative laboratory study in which 27 male and 24 female end users debugged spreadsheet formulas as their problem-solving task.[11] The research spreadsheet system contained a variety of features that have previously been shown to help end users test and debug spreadsheet formulas.[12]

We partitioned the features into three categories for analysis: familiar features such as editing formula text boxes, features taught during the experiment's tutorial such as checkmarks and arrows, and unfamiliar features that were untaught such as Xs. We sought to determine how much males and females used these three feature types, the relationships between their feature usage and their self-efficacy, and the implications of these findings for task success.

Table 1 shows several statistical results from this experiment. There were indeed gender differences in self-efficacy, with female results significantly lower. Further, females were significantly slower to try out the new features and also were significantly less likely to adopt them for repeated use. Females gravitated toward the familiar feature of editing formulas, whereas males were more likely to try less familiar features early.

Interestingly, the relationship between self-efficacy and feature usage differed for males and females. For females, low self-efficacy predicted low effective feature usage. For the males, however, self-efficacy was not a predictor.

An obvious question then arises: Were the females correct in believing they had such limited abilities? The evidence does not suggest this; rather, it points toward females' low self-efficacy and their low usage of the new problem-solving features as hindrances to their performance.

First, we found no significant difference in the males' and females' performance in fixing the bugs provided. However, females proved significantly more likely to introduce new bugs that they never fixed. This seems tied to their heavy reliance on formula editing instead of the other problem-solving features—formula editing is the only way users can introduce new bugs.

Second, females agreed significantly more than the males with this statement: "I was afraid I would take too long to learn [the untaught feature]." However, despite the gender differences in expectation of their ability to learn the untaught feature, the subjects showed no significant gender differences in their actual learning of it. It appears that the females perceived inappropriately low self-efficacy, which inhibited their use of a new fea-

ture and in turn prevented them from receiving its benefits. In effect, their inaccurate assessment of ability became a self-fulfilling prophecy.

Attitudes toward new features

Several other experiments—both qualitative and quantitative—that we have conducted all revealed some form of gender difference in self-efficacy. The qualitative experiments consistently illustrate a common attitude in the females: low self-efficacy seems to hinder their ability to cope with new, unfamiliar features.

For example, Experiment 3 involved a qualitative think-aloud study that added a *guards* feature to the research environment, with red circles akin to those of Excel's data validation feature, which circles out-of-range values. The comments of female participant F1 exemplify the attitude that has consistently emerged from females with low self-efficacy in these experiments:

> What's this little arrow doing? They're everywhere! So, I need to take this—oh, my goodness. Now what's happening? … too much happening.

Experiment 3 also revealed that the males and females perceived features differently. For example, while using the guards feature, female F3 said,

> I don't think that you can get a -5 on the homework. No, it can't be. So 0 to 100 [is the guard I'm entering], ok. Ok, hmm … So, it doesn't like the -5 [...]. They can get a 0, which gets rid of the angry red circle.

In contrast to F3's focus on the guard feature as a way to get her spreadsheet to work correctly, male M4 initially focused on the feature itself:

> The first thing I'm going to do is go through and check the guards for everything, just to make sure none of the entered values are above or below any of the ranges specified. So, I'm going to put guards on everything because I feel like it. I don't even know if this is really necessary, but it's fun.

Table 1. Experiment 2's statistical results, with bold values showing statistical significance at $p < .05$.

Result type	Outcome
Self-efficacy	Mann Whitney: ***p* <.018** (males higher)
Self-efficacy predicts effective feature usage:	Linear regression (males no, females yes):
Type taught	Males: $p < .551$, F(1,25)=.365, R2=.015
Type untaught	Females: ***p* <.046**, F(1,22)=4.52, R2=.177
Likelihood of initially trying new features:	ANOVA (males used earlier):
Type taught	Taught: ***p* <.005**, F(1,49)=8.69
Type untaught	Untaught: $p < .073$, F(1,40)=3.40
Likelihood of adopting new features for repeated use:	Various methods (males more):
Type taught	ANOVA: ***p* <.03**, F(1,49)=4.971
Type untaught	Fisher's exact test: ***p* <.01**
Bugs fixed	Mann Whitney: $p < .651$ (no difference)
Bugs introduced	Fisher's exact test: ***p* <.015** (females more)
Thought features would take too long to learn	Mann Whitney: ***p* <.017** (females more)

In fact, these gender differences in views toward the same feature are consistent with reports of gender differences regarding motivation for using technology, majoring in computer science, and how children talk about the use of technology.[6,8] In particular, the male participant's comment about using the guards "because I feel like it" is similar to oft-reported reasons males give for majoring in computer science: technology for the fun of it.

The tinkering factor

Male M4's playful experimentation—tinkering—is often encouraged in educational settings because of its expected educational benefits.[13] Tinkering with problem-solving software could yield similar positive benefits if it encourages users to incidentally and iteratively gain knowledge of the software's features, as M4 did. But the education literature reports that tinkering is a strategy that males adopt more often than females,[14] possibly leaving females without the advantages they could gain from this strategy.

An exploratory analysis of Experiment 2's data showed that males did significantly more tinkering with the untaught feature than females did, which might explain the males' greater interest in it. Given the strongly suggestive evidence from Experiments 2 and 3 of tinkering differences by gender, we embarked on Experiment 4 to investigate whether the self-efficacy factor would discourage females from tinkering and whether this would be tied to negative outcomes for them.

Experiment 4's design[15] resembled Experiment 2's, but added a comparison with a second software environment that added features intended to provide greater support. We term the second software environment *high support*, and the original software environment *low cost*. For both genders, we measured participants' tinkering, self-efficacy, and effectiveness in the high-support and low-cost environments.

The results revealed several differences between the genders in their tinkering practices and how these related to self-efficacy and to effectiveness. Males tended to be comfortable with tinkering, which seemed to match many of their problem-solving styles. But this was not wholly a good thing for the males: they also had a tendency to tinker to excess.

Figure 2 depicts these tinkering tendencies and their relationships to effective outcomes. On the female side of Figure 2, starting at the bottom, the

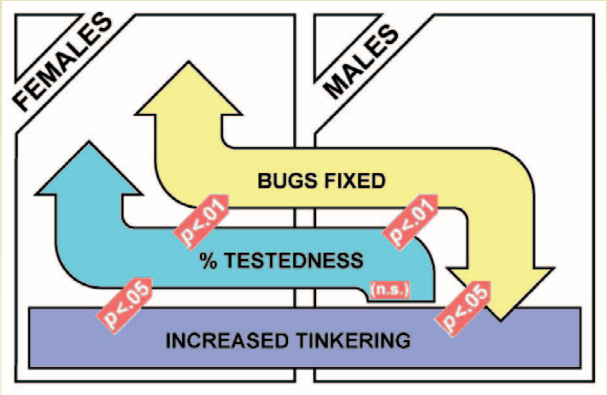

Figure 2. Tinkering effectiveness. Males' and females' tinkering affected their debugging effectiveness in unexpectedly opposite ways. The direction of the stylized arrows depicts a measure's increase or decrease, while the red arrows show the significance of the regression relationships between measures.

rectangle depicts the amount of tinkering. Regardless of software environment, increased tinkering was significantly predictive of females' testing effectiveness, as the red arrow pointing up shows. Following the next red arrow up shows that, just as in the previous study, increasing testing effectiveness was significantly predictive of increased bugs fixed for the females.

Moving across to the male side of Figure 2, the predictive relationship between effective testing and bugs fixed also held true for the males, but unlike the females, the tie between males' tinkering and testing effectiveness was insignificant. Further, males' tinkering was inversely predictive of bugs fixed.

The predictive relationship between effective testing and bugs fixed also held true for the males, but unlike the females, the tie between males' tinkering and testing effectiveness was insignificant. Further, males' tinkering was inversely predictive of bugs fixed.

The differences between the males' and females' tinkering behaviors, and the benefits that derived from them, appear to be tied to two factors. First, some males exhibited a tendency to tinker without pausing much. In fact, males paused significantly less often than the females, and, when they did such pauseless tinkering, their effectiveness suffered. Research from the education field has shown that if students are given a wait time of three seconds or more after a classroom response, their critical thinking about that response improves.[13] We found similar results: Pauses after any action were predictive of more understanding and more effective use of problem-solving features.

Second, the differences in the software environments affected male and female tinkering behavior differently. Females tinkered about the same amount in the two software environments, but males' tinkering—roughly comparable to the females' in the high-support environment—skyrocketed in the low-cost environment, where tinkering was easy to do and males were willing and eager to tinker to excess. This was not the case for females.

The results suggest that tinkering in either environment helped females gain valuable information about the features, leading to positive outcomes. However, there were both positive and negative ties between tinkering and females' self-efficacy.

As with our other experiments, pretask self-efficacy was an important predictor of effectively using the features. One type of tinkering in the high-support environment—a pattern of intermittently tinkering in fits and starts—was predictive of a drop in post-task self-efficacy. The more the females tinkered intermittently in this environment, the larger their drop in self-efficacy.

One possible interpretation of the findings from Experiments 2 and 3 is that as the females received increased feedback from the high-support environment through their tinkering, they became more convinced that they would not be able to master its more complex features. Another possibility is that in the less complex environment, there was suggestive evidence linking increased tinkering frequency to increased self-efficacy for the females after the task.

This suggests that females' self-efficacy could be more sensitive than males' to tinkering with software in which the user interface seems to contain obstacles, such as excessive complexity. The males did not experience changes in self-efficacy tied to tinkering in either environment.

The males' biggest tinkering-related problem was that too much tinkering interfered with their effectiveness. The females' biggest problem was that, while they found tinkering helpful in some circumstances, under others it interfered with their self-efficacy.

GUARDING AGAINST SOFTWARE BARRIERS

To maintain its creative edge, an organization must encourage a diversity of ideas. This holds true for the employees who create the software an organization builds, and also for the employees who use software in problem-solving activities—like Ashley.

In fact, Ashley is a *male*. His story is true. Ashley went on to a college career in art, and ultimately won the most prestigious academic award his university bestows. He enjoys art, but regrets his decision not to pursue graphic design. Now that Ashley has graduated, the barriers to making the switch back to graphic design loom even higher because he no longer has access to the educational support structures available to students. Still, at home after work, he strives to overcome the barriers that prevent infor-

mation technology from being a good fit to his strengths.

Although gender differences in self-efficacy, motivation, problem-solving styles, learning styles, and information-processing styles are all implicated in this issue, we must remember that no single female will likely have every trait statistically associated with females, nor will any single male likely have every trait statistically associated with males. For example, some males process information in the comprehensive style statistically associated with females, and some females process information in the more linear style associated with males. Thus, designing software in ways that support these differences does not penalize either gender—it helps everyone.

Ample evidence suggests that gender differences exist in the ways people solve problems. Our results show that these differences are highly relevant to users' ability to gain benefits from the features that exist in software. As we continue this work, we hope to ultimately identify specific features and attributes of software that present barriers so that alternative choices can be made available to the software users. ■

Acknowledgments

We thank Alan Blackwell, Thippaya Chintakovid, Curtis Cook, Xiaoli Fern, Michelle Hastings, Sienna Hiebert, Derek Inman, Cory Kissinger, Joseph Lawrance, Vaishnavi Narayanan, Kyle Rector, Shraddha Sorte, and Sherry Yang for their significant contributions to this research. Special thanks also to Alan Blackwell regarding Figure 2. This work was supported in part by Microsoft Research, by NSF grant CNS-0420533, by an IBM Faculty Award, and by the EUSES Consortium via NSF grants ITR-0325273 and CCR-0324844.

References

1. L. Beckwith and M. Burnett, "Gender: An Important Factor in Problem-Solving Software?" *Proc. IEEE Symp. Visual Languages and Human-Centric Computing Languages and Environments*, IEEE Press, 2004, pp. 107-114.
2. K. Kucian et al., "Gender Differences in Brain Activation Patterns during Mental Rotation and Number-Related Cognitive Tasks," *Psychology Science*, vol. 47, no. 1, 2005, pp. 112-131.
3. C. Huff, "Gender, Software Design, and Occupational Equity," *SIGCSE Bull.*, vol. 34, no. 2, 2002, pp. 112-115.
4. D. Tan, M. Czerwinski, and G. Robertson, "Women Go with the (Optical) Flow," *Proc. ACM Conf. Human Factors in Computing Systems*, ACM Press, 2003, pp. 209-215.
5. C. Van Slyke, C. Comunale, and F. Belanger, "Gender Differences in Perceptions of Web-Based Shopping," *Comm. ACM*, Aug. 2002, pp. 82-86.
6. J. Cassell and H. Jenkins, eds., *From Barbie to Mortal Kombat: Gender and Computer Games*, MIT Press, 1998.
7. J. Cohoon, "Toward Improving Female Retention in the Computer Science Major," *Comm. ACM*, May 2001, pp. 108-114.
8. J. Margolis and A. Fisher, *Unlocking the Clubhouse: Women in Computing*, MIT Press, 2002.
9. M. Othman and R. Latih, "Women in Computer Science: No Shortage Here!" *Comm. ACM*, Mar. 2006, pp. 111-114.
10. A. Bandura, "Self-Efficacy: Toward a Unifying Theory of Behavioral Change," *Psychological Rev.*, vol. 8, no. 2, 1977, pp. 191-215.
11. L. Beckwith et al., "Effectiveness of End-User Debugging Features: Are There Gender Issues?" *Proc. ACM Conf. Human Factors in Computing Systems*, ACM Press, Apr. 2005, pp. 869-878.
12. M. Burnett, C. Cook, and G. Rothermel, "End-User Software Engineering," *Comm. ACM*, Sept. 2004, pp. 53-58.
13. M.D. Rowe, *Teaching Science as Continuous Inquiry: A Basic*, 2nd ed., McGraw-Hill, 1978.
14. M.G. Jones et al., "Tool Time: Gender and Students' Use of Tools, Control, and Authority," *J. Research in Science Teaching*, vol. 37, no. 8, 2000, pp. 760-783.
15. L. Beckwith et al., "Tinkering and Gender in End-User Programmers' Debugging," *Proc. ACM Conf. Human Factors in Computing Systems*, ACM Press, Apr. 2006, pp. 231-240.

Laura Beckwith *is a PhD candidate in the School of Electrical Engineering and Computer Science at Oregon State University. She received an MS in computer science from Oregon State University. Contact her at beckwith@ eecs.orst.edu.*

Margaret Burnett *is a professor in the School of Electrical Engineering and Computer Science at Oregon State University. She received a PhD in computer science from the University of Kansas. She is a member of the IEEE Computer Society, a senior member of the IEEE, and a member of the ACM. Contact her at burnett@eecs.orst.edu.*

Susan Wiedenbeck *is a professor in the College of Information Science and Technology at Drexel University. She received a PhD in information science from the University of Pittsburgh. She is a member of the ACM. Contact her at susan.wiedenbeck@cis.drexel.edu.*

Valentina Grigoreanu *is a PhD student in the School of Electrical Engineering and Computer Science at Oregon State University. She received a BA in computer science/mathematics and a BA in communication from Lewis and Clark College. She is a member of the IEEE and the ACM. Contact her at grigorev@ eecs.orst.edu.*

Series editor: Juan E. Gilbert, Dept. Computer Science and Software Engineering, Auburn University; bpc@computer.org