Designing Features for Both Genders in End-User Programming Environments

Laura Beckwith*, Shraddha Sorte*, Margaret Burnett*,
Susan Wiedenbeck†, Thippaya Chintakovid†, and Curtis Cook*

*Oregon State University
Corvallis, OR 97331
{beckwith, sortes, burnett, cook}@cs.orst.edu

†Drexel University
Philadelphia, PA 19104
{Susan.Wiedenbeck, Thippaya.Chintakovid}@cis.drexel.edu

Abstract

Previous research has revealed gender differences that impact females' willingness to adopt software features in end users' programming environments. Since these features have separately been shown to help end users problem solve, it is important to female end users' productivity that we find ways to make these features more acceptable to females. In this paper, we draw from our ongoing work with users to help inform our design of theory-based methods for encouraging effective feature usage by both genders. This design effort is the first to begin addressing the gender differences in the ways that people go about problem solving in end-user programming situations.

1. Introduction

Although there have been gender studies designed to understand and ameliorate the low representation of females in the computing field [12], there has been little emphasis on how software's *design* affects females' and males' performance in computing tasks. Building upon theories and research on gender differences from a number of domains [6], we have begun investigating whether there are features within software that interact with gender differences in the realm of end-user programming.

Our method for conducting this investigation is as follows: (1) use theory and previous gender difference empirical work from other domains to hypothesize gender issues and their causes that could arise in end-user programming environments, (2) use empirical methods to investigate whether these issues do actually arise in end-user programming environments, (3) use the results of the first two steps along with theory and qualitative empirical work involving low-cost prototyping to derive and refine approaches to address the issues, and (4) use quantitative empirical methods to evaluate the effectiveness of the approaches.

Our work on the first step was presented in [6]. In that paper, we derived a set of hypotheses from relevant research literature; the subset of those hypotheses of interest to this paper is given in Table 1. A particularly useful aspect of these hypotheses is that, because many of these hypotheses are theory-based, they tend to suggest a cause for the hypothesized effect. These causes potentially point the direction for our designs to take in addressing issues that are present.

Our work on the second step has so far concentrated on hypotheses H1 and H2 in the table. To investigate these hypotheses, we conducted a study [7] in which we gave male and female spreadsheet users two spreadsheet debugging tasks in an environment containing a number of features that support such debugging tasks. The hypotheses were confirmed by our investigation:

- 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

Table 1: Theory-based hypotheses about gender differences in end-user programming environments [6].

Basis: Confidence and Risk
H1: There will be gender differences in users' interest in (initially) exploring new features in end-user programming environments.
H2: Females' high perceptions of risk will render them less likely to make (genuine) use of unfamiliar devices in end-user programming environments.
Basis: Learning Theory, Problem-Solving Style, and Information Processing Style
H3: Gender differences in learning style will cause some software devices aiming to "teach" new features or procedures to be less effective for one gender than another.
H4: An end-user programming environment that restricts users to a linear (non-linear) approach will adversely impact females' (males') abilities to problem-solve effectively in that environment.
H5: Males will be less likely than females to thoroughly read complicated or lengthy "help" explanations.

Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)
0-7695-2443-5/05 $20.00 © 2005 **IEEE**

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.

In this paper, we report the results of the third step of our investigation as applied to the above findings related to H1 and H2: developing potential solutions to address the issues revealed. H3, H4, and H5 are also relevant to these design efforts, because we derived our approaches in part from the bases of those hypotheses. After reviewing our earlier work on gender and end-user programming (Section 2), we used theory-based approaches to derive two solutions, which we evaluated using low-cost prototyping (Sections 3 and 4).

This paper's contributions are two-fold. First, the paper shows the application of these particular theories to the design of potential solutions to address gender issues in end-user programming features. The second contribution is the potential solutions, which are, to our knowledge, the first reported approaches to target gender issues for end-user programming environments.

2. The Starting Point

For some time, we have been working on a concept we term "end-user software engineering" [11]. The essence of the end-user software engineering concept is to tightly intertwine into end-user programming environments features that aid end users in guarding against errors in the "programs" they create (spreadsheets in our case). In this section, we describe the end-user software engineering features as they existed in our prototype at the time of the empirical study that investigated H1 and H2.

As the results of H1 and H2 showed, the environment was not as effective for females as it was for males. As one specific example, females' self-efficacy was a significant predictor of their effectiveness testing spreadsheet formulas, as the positively sloping line in Figure 1 shows. For the males, however, this was not the case. In short, for females, low self-efficacy was tied to low usage of useful features, creating a barrier to effectively testing and debugging their spreadsheet formulas.

2.1 The Features: WYSIWYT with Fault Localization

Two end-user software engineering features in our environment are the WYSIWYT ("What You See Is What You Test") features that allow users to incrementally "check off" ("√" in Figure 2) or "X out" ("X" in Figure 2)

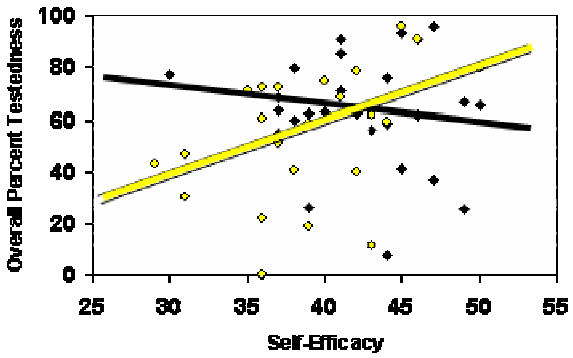


Figure 1: Light/dark colors represent the females/males, respectively. There was a significant relationship between females' self-efficacy and spreadsheet testedness (a measure of effective feature usage).

values that are correct or incorrect, respectively [20, 21]. A third features is optional dataflow arrows for making relationships among the cells explicit (the arrow in Figure 2). These three features were present when the above empirical results were obtained.

Marking values correct and incorrect allows the system to track the "testedness" and estimate the fault likelihood of all the cells contributing to those correct and incorrect values. Untested cells start with red borders (light gray in this paper), and as a cell becomes more tested, the cell's border becomes more blue (more black in this paper). X-marks trigger fault likelihood calculations, which cause the interiors of cells suspected of containing faults to be colored along a yellow-orange continuum (shades of gray in this paper), with darker orange shades given to cells with increased fault likelihood. The overall goal of these features is to encourage the user to test the spreadsheet thoroughly and, if the testing reveals incorrect values, to lead the user to the faulty formula(s).

2.2 Surprise-Explain-Reward

These environment features are supported via the Surprise-Explain-Reward strategy [23]. If a user is surprised by or becomes curious about any of the feedback of the debugging features, such as cell border color or interior cell coloring, he or she can seek an explanation, available via tool tips (as in Figure 2). If the user follows up as advised in the explanation, rewards potentially ensue.

Our empirical results with end-user programming as supported by Surprise-Explain-Reward have been encouraging [11, 23]. Still, the results of our investigation into H1 and H2 [7] suggest that the Surprise-Explain-Reward strategy was not as effective at enticing females as it was males to use the features. This was the case not only for adopting and using the features, but even for initially approaching the features. Our theory-based hypotheses H1 and H2 suggest that females' lower confidence and higher

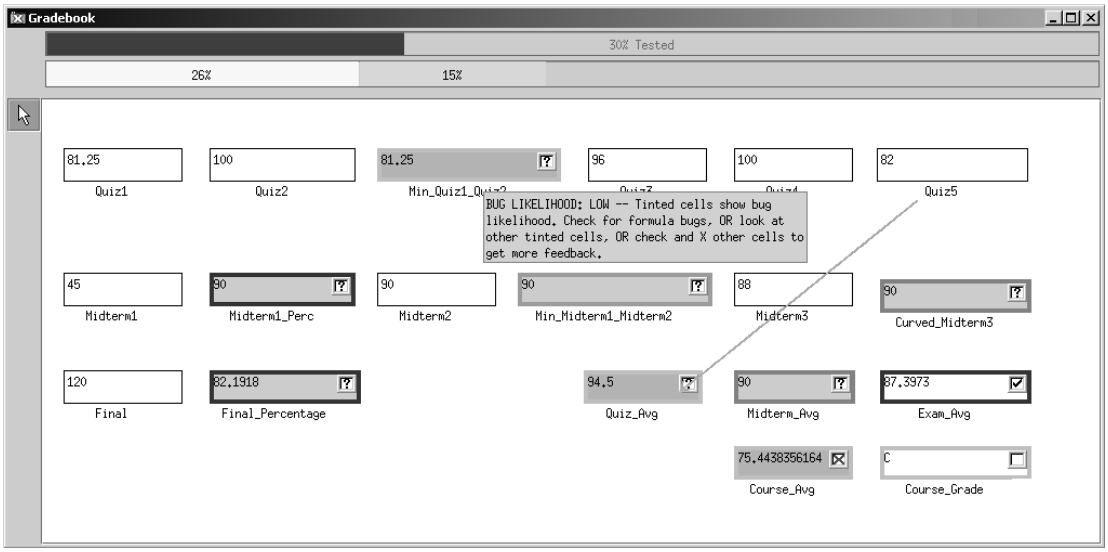


Figure 2. WYSIWYT with fault localization as prototyped in Forms/3 [10]. The user notices an incorrect value in Course_Avg and places an X-mark in the cell. As a result of this X and the checkmark in Exam_Avg, eight cells are identified as being possible reasons for the incorrect value, with some deemed more likely than others.

perception of risk may be causes. Also, since much of our strategy rests upon explanations, it seems likely that the bases for H3-H5 (learning theory, problem-solving style, and information processing style) may also be contributors. We next consider specific barriers that may be contributing to these results, and how to remove them.

3. From Problem to Solution 1: "No Confidence Required"

From a high-level design perspective, we are dealing with an "ill-structured" [22] problem. In such problems, formulating the problem and the solution are not entirely separate issues, because each attempt to solve the problem changes the researchers' understanding of the problem. The potential solutions are not well-defined, theory is incomplete, and information upon which a solution can be based is also incomplete.

In our own ill-structured setting we drew from a combination of existing empirical results, theories, and human-computer interaction (HCI) design techniques. Following Ko et al.'s example [16], we use the concept of "barriers" to help organize the problem space. Table 2 lists possible barriers and potential solutions to help females overcome these barriers. The hypotheses of Table 1, the associated research contributing to those hypotheses, and our empirical results on H1 and H2 were the sources of the barriers and of the potential solutions.

3.1 Barriers and Potential Solutions

As we have already pointed out, Barrier 1, low confidence in females in computer-related tasks, has been widely reported, as has risk aversion in females [6].

According to the attention investment model [8], users will take an action if they believe that the action's benefits

Table 2: Barriers females faced related to the findings of H1 and H2 and potential solutions.

Barrier	Potential Solutions
Barrier 1: Low computer-related confidence in females (as measured in [7] and numerous other sources).	Emphasize low risk nature of judgments by providing a way to make it acceptable to express less confident judgments. (For example: not very sure to very sure)
	Provide a "what if these cells were wrong" feature, where users can get feedback, but do not have to commit to saying that the cells are definitely wrong.
	Experience helps in increasing confidence.
Barrier 2: Low feature usage by females [7].	A WYSIWYT Skill Builder (similar to a Wizard, but set up to facilitate learning without being overly directive) to introduce users and lead them to greater skills.
Barrier 3: Perception that it will take too long to learn the X-mark feature (reported by females in [7]).	Clearly state X-mark's usefulness, to emphasize the value of learning the X-mark.
	Watch someone else use X-marks.
	Enhance fault localization feedback to help users understand how fault localization narrows down the potentially faulty formulas.
	Expand content of explanations to help users make more accurate assessment of risks and benefits of using the X-mark feature.

are greater than their perceived costs and are likely to materialize given the perceived risks. This implies that our approach should emphasize the low risk nature of checkmarks and X-marks. Taking this into account in conjunction with females' low confidence led to two low-risk, low-confidence design ideas, in which users need not be 100% certain of the correctness of their judgments in order to make these marks (the first two potential solutions listed in Table 2). The third potential solution, increasing experience to help increase confidence, is based on Bandura's self-efficacy theory [4]. Bandura argues that the best way to increase self-efficacy is to give the low-confidence individual more experience in personally accomplishing the task. This solution does not seem very useful by itself—seeming to come down to "the best way to increase feature usage is to increase feature usage"—but it could magnify the effects of solutions, such as the first two, that encourage users to get at least a little experience in the course of trying out the features.

Barrier 2, low feature usage by females, is not independent of the other barriers, but is present in our table because it encourages thinking directly about usage, rather than concentrating only on underlying causes, as in the other barriers. A proposed solution is to provide a "wizard-like" entity, such as Excel's Chart wizard, to facilitate feature usage and to build skills. This approach draws from minimalist learning [19, 13], which advises that new system features should be introduced by engaging users in activity and providing scaffolding to help them gradually increase their skills. As this learning theory advocates, the scaffolding would avoid being so overly directive that users blindly follow instructions; thus the device would be somewhat different from traditional wizards, which tend to be very directive.

Barrier 3, females' perceptions that it takes too long to learn the X-mark feature has several possible solutions. The first is ensuring the usefulness of the feature is clearly stated. The attention investment model's benefits component suggests that, if benefits of placing X-marks are not obvious to users, they are not likely to see learning the feature as a good use of their time, especially if they expect that amount of time to be large. The second solution, drawn from self-efficacy theory [4], indicates that observing peers accomplishing the task is an important source of self-efficacy. This would mean that a low self-efficacy female should observe another female peer.

It is also possible that the feedback about the results of X-marks led to Barrier 3. If so, then enhancing the feedback would help reduce the barrier. From a theoretical perspective, Norman's action cycle [18] points out that to carry out a task successfully, users must correctly interpret feedback on their actions. Arroyo [3] and Beck et al. [5] support interactivity in learning to understand tasks, and both studies revealed useful information about gender. Arroyo's study suggested that concrete and interactive hints helped females to perform better and learn more. Beck et al.'s study further indicated that highly interactive hints helped increase females' confidence.

3.2 Claims Analyses

For each solution in Table 2 we performed a claims analysis. Claims analysis [14] is a technique for evaluating design solutions. In claims analysis the researchers identify positive and negative consequences of each solution with respect to the intended users. Our claims analyses were instrumental in helping us to improve our solutions and to choose which solutions to implement. For example, the claims analysis for the first solution in the table (which became our "Solution 1") is shown in Table 3.

3.3 Solution 1's Prototype

Solution 1's goal was to communicate to users that they did not have to be confident to judge the correctness or incorrectness of values. Thus, in our prototype, instead of having only two possible actions—checking off or X'ing out values—there are now four possible actions: the original two ("it's right" and "it's wrong") plus "seems right *maybe*" checkmarks and "seems wrong *maybe*" X-marks. See Figure 3. The lighter colored marks are for lower confidence judgments, as their tool tips explain.

One small but important detail: another way this change differs from the previous prototype is that in the previous version, the checkmark was done with a left click and the X-mark with a right click. Removing the need for a right click, which we have observed is not often used by less experienced users, may make X-marks more accessible to those with less experience.

The lower confidence marks result in feedback at lower saturations. That is, a lower confidence checkmark produces lower saturations of border colors reflecting the affected cells' "testedness." Similarly, a lower confidence

Table 3: The claims analysis for Barrier 1 of Table 2.

Problem (re: Barrier 1): Females might use checks or X-marks only when they are confident about their judgments.
Potential Solution: Emphasize low risk nature of judgments by providing a way to make it acceptable to express less confident judgments.
Pros: + may increase willingness to use checkmarks or X-marks. + user receives feedback that encourages placing a mark at the moment he/she questions a cell. + optional—user not forced to use it—yet noticeable.
Cons: - another step for users to perform, taking more time. - may be seen as greater complexity. - might be too many "environments" to keep track of.

Figure 3: Clicking on the checkbox turns it into the four choices. The tool tips over the choices, starting with the left-most X, are "it's wrong," "seems wrong maybe," "seems right maybe," "it's right."

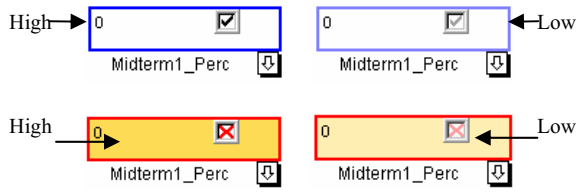


Figure 4: Saturation of border color (top) and interior color (bottom) reflect confidence of user judgments of values being correct or incorrect.

X-mark produces lower saturations of interior colors reflecting the affected cells' fault likelihood. See Figure 4. Like the increases/decreases in testedness and fault likelihood that arise from the correctness judgments communicated through checkmarks and X-marks, the confidence of these judgments are also propagated to all affected cells.

3.4 Feedback from Users

As the prototype evolved, we brought in end users with no programming experience, one at a time, (two males and six females) to use our prototype, in order to inform our design of the prototype changes. Each participant was asked to "think aloud" while working on the same tasks as in [7]. The tasks were followed by interviews.

Only three participants used the low-confidence marks, but in general the participants did seem to be more willing to make judgments than they had been in previous studies. This change seemed especially apparent with the X-marks. Thus, the changes may have indeed succeeded in communicating the low risk and acceptability of low confidence. However, without a statistical study, we cannot be sure that such a change occurred.

For example, one female (S4) used the approach exactly as we had hoped. Here is what she said while contemplating a cell's value:

> S4 (thinking aloud): "I am not sure if this cell's value is right so maybe I'll mark it gray and come back to it later."

However S3, a female, did not use the low-confidence marks and later told us she did not see their importance:

> S3 (interview): "I didn't use the 'maybe' marks because I thought that they might not help me any more than the other ones in my task."

S3 also made some revealing comments relating to Barrier 3:

> S3 (interview): "I didn't know what was wrong when it seemed correct to me ...why it showed 50 and not 100 [% tested]."
> Interviewer: "Weren't the tool tips helpful?"
> S3 (interview): "Yeah, they were good but sometimes I didn't find the answer that I wanted …I needed more answers than were present."

Comments such as this one pointed us toward the path to Solution 2.

4. Solution 2: Explanations

The addition of low-confidence marks may have helped with the usage of marks, but the evidence is not overwhelming. To strengthen our approach, we decided to tackle Barrier 3 (Table 2), perceived difficulty of learning, via the learning vehicle in the system, explanations.

As pointed out in Section 2.2, explanations are a critical part of the Surprise-Explain-Reward strategy [23]. They connect surprises with rewards by providing users with a low-cost mechanism (tool tips) to explore objects that arouse their curiosity.

Until the work we report here, explanations were as follows: each explanation described the semantics, the action users should try, and a potential reward. They were designed with minimalist learning theory in mind, with the goal of encouraging users to learn by doing and to stay connected to the task they were working on when they sought the explanations. Therefore, we kept the explanations short—typically one to three very short lines.

4.1 Requirements on Types of Explanation Content

We used the theory that generated the hypotheses of Table 1 to also help develop requirements on the solutions for both Solution 1 and Solution 2. For example, one important influence on the redesign of our explanations' content was the evidence suggesting that the above short explanations may not be well suited to females. According to research in information processing and in education, short explanations such as these are closer matches to the type of information processing and learning environments in which males thrive, not females [3, 5, 17].

Anson's essay on minimalist learning theory, a second important influence on Solution 2, discusses content and delivery of minimalist documentation [2]. Content is described using the terms *conceptual*, *procedural*, and *problem solving*. These terms provide a useful framework for organizing requirements on explanations' content types. Anson did not provide precise definitions, but we use the term "conceptual" for content relating to concepts and semantics, "procedural" for how to perform actions, and "problem solving" for higher-level strategies directed toward "big picture" goals. Together, these terms form completeness requirements for our content *types*; that is,

we require explanations to be available with conceptual, procedural, and problem-solving content.

A third influence on Solution 2 was Ko et al.'s work on learning barriers [16]. We used these learning barriers to cross-check our list of content type requirements for completeness and to solidify each requirement's aim.

A final influence came from research on learning [15] and problem-solving [1] styles. These works have found that females' styles tend to be non-linear (not necessarily sequential in nature), whereas males' tend to be linear (sequential). As a result, we required that our redesigned explanations support both linear and non-linear styles.

4.2 Applying the Requirements

The content type requirements of Section 4.1 led initially to three additional components in the explanations: a "what" component to fulfill the conceptual requirement, a "how should..." component, to fulfill the procedural requirement, and an "advice" component to fulfill the problem-solving requirement. Eventually, we subdivided the conceptual component for clarity of labeling: a "what" component with declarative information and a "how did..." component that explains how the current state came about (emphasizing system responses to user actions). Users of our low-cost prototype experienced the new components primarily in the form of paper augmentations to our executable prototype, as shown in Figure 5.

In addition, the actual content of each type necessitated an orthogonal set of requirements. Table 4 lists the requirements, along with the originating theories.

4.2.1 Conceptual: The "What" Component

> S7 (thinking aloud): "I don't understand why this [cell] is not 100% tested when it appears to have the right value."

Figure 6 shows an example of a short explanation ("50% of this cell has been tested") and the additional components. The goal of the "what" component is to communicate the semantics of the object in more detail than the short explanation:

> The purple border means that this cell has been partially tested, but that other situations still need to be tested. The √ says you have tested this cell's value.

The first two sentences of this "what" component demonstrate Requirement 5 well (Table 4). This theory suggests that information be presented to users only when the information is relevant [13]. Separating the "what" component from the "how" and "advice" components is one way we applied this theory, because it gives the user a way to communicate what question they are wondering about. We also applied this theory by tying the explanations to specific objects, where users, through their hovering actions, get information on exactly *which* object, in *which* state, they are curious about.

The last sentence of this component, "Trying more

Figure 5. In our low-cost prototype, the user's request for an additional explanation component (bottom) caused the examiner to add it to the screen (top). Note the support for non-linear approaches—a user can view many unrelated components simultaneously.

situations helps you find errors" demonstrates Requirements 8 and 11, by relating the object's current situation to the big picture and keeping the rewards clear.

Note the emphasis on testing, rather than on the actions and feedback. Several learning theories dissuade giving users information that is too directive (Requirement 3), resulting in users simply taking the action without thinking or learning from it [13, 9]. Thus, we elected to stress testing situations, rather than checking cells off to achieve a blue cell border as the goal.

4.2.2 Conceptual: The "How did…" Component

> S8 (thinking aloud): "...how did I do that?"

The "how did" component explains what steps the system or user took to get the object to its current state:

> The purple border and the √ means you previously decided that this cell's value(s) was correct, and checked it off.

This component was particularly influenced by Requirements 4 and 7. Its tie to Requirement 4 is simply that it helps the user to interpret the meaning of the feedback. Requirement 7, which comes from various learning theories, allows omission of information the user may well already know (and if not, they can always ask again via "how should"). According to these theories, this encourages users to make ties among the different explanation components and their experience using the spreadsheet features. These interconnections help them learn.

For S8, who proceeded to open this component in order to answer her question above, the "how did…" content provided her with the information she needed:

> S8 (thinking aloud): "Oh yeah, I should test it more."

4.2.3 Procedural: The "How should…" Component

> S8 (thinking aloud): "How should I test it more?"

The "how should…" component suggests action(s) users can take to make progress on their task:

> You can get into a new situation by changing some of the in-

put values. Looking for new testing opportunities (marked by ?s) helps you make progress testing.

The second sentence of the above example aligns especially with two main themes of minimalist learning theory: keeping the user task-oriented and active (Requirements 1 and 2, respectively). The example component above reminds the user of the focus — testing, and suggests specific actions they can take to make progress on this task. Note that it also reminds them of the meaning of the "?" feedback device (Requirement 4), to help them evaluate the result of the action if they do take it.

Table 4: The explanation content requirements.

Content Requirements	Sources
1. Is task oriented.	Minimalist learning [13]
2. Keeps user active.	Various learning theories [9, 13]
3. Explanation not too directive.	Various learning theories [9, 13]
4. Explains how to evaluate whether an action taken was the right one to take.	Norman's action cycle [18]
5. Is context-appropriate: User should care about information when presented.	Minimalist learning [13]
6. Suggests strategies for a difficult task.	Minimalist learning [19]
7. Encourages user to take advantage of prior knowledge.	Various learning theories [19, 9]
8. Explains why task is meaningful (relate to big picture).	Motivation (summarized in [6])
9. Provides enough information for users to accurately assess risks and benefits.	Risk, information processing (summarized in [6]), Attention investment [8]
10. Makes obvious the actions that need to be taken.	Minimalist learning [13]
11. Makes sure rewards are clear.	Attention investment [8]

4.2.4 Problem Solving: The "Advice" Component

The "advice" component provides ideas about higher-level strategies to achieve the "big picture" goals. One of the purposes is to help orient the user to this feature within the context of their overall task.

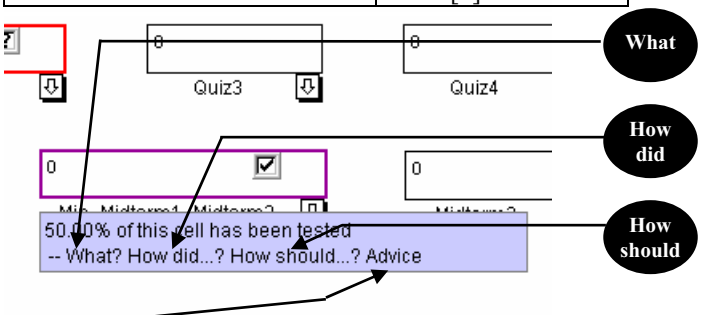
> You can use the border colors to systematically test your spreadsheet. If you can make a decision about a cell's value (correct or wrong) you can (1) test this cell given different inputs, or (2) move on to testing another cell, or (3) if there are tinted cells, which indicate possible locations of errors, follow the system's guidance (cells with darkest tints) to find the cause(s).
> Border colors reflect the number of √s on this or related cells, and tints on the entire cell reflect the number of Xs (in relation to the number of √s) on this or related cells.

The "advice" component satisfies Requirement 6, which is important when the complexity of a task is high and users need ideas on how to approach the task. In this example, the advice component suggests three strategies.

There is a fine balance in the advice components between providing enough information (Requirements 9, 10, and 11) without providing too much (Requirements 3 and 5). As Requirement 10 clarifies, it is important for the user to know how to follow through. Further, pertinent to satisfying Requirement 9, research on gender differences in perceived risk, risk aversion, and the way the females process information suggests that females may need many details before taking an action. Finally, according to the attention investment model, the explanation component may be important in decreasing users' perceptions of risk and/or increasing their perceptions of benefits.

5. Conclusion

In this paper, we describe our third step of investigation into gender issues in end-user programming environments. The first two steps of our four-step investiga-



What

The purple border means that this cell has been partially tested, but that other situations still need to be tested. The √ says you have tested this cell's value. Trying more situations helps you find errors.

How did

The purple border and the √ mean you previously decided that this cell's value(s) was correct, and checked it off.

How should

You can get into a new situation by changing some of the input values.
Looking for new testing opportunities (marked by ?s) helps you make progress testing. Testing helps you find errors.

Advice

You can use the border colors to systematically test your spreadsheet. If you can make a decision about a cell's value (correct or wrong) you can (1) test this cell given different inputs, or (2) move on to testing another cell, or (3) if there are tinted cells, which indicate possible locations of errors, follow the system's guidance (cells with darkest tints) to find the cause(s).
Border colors reflect the number of √s on this or related cells, and tints on the entire cell reflect the number of Xs (in relation to the number of √s) on this or related cells.

Figure 6. The top line of the tool tip contains a very short explanation. The expansion components will be clickable via the "What?", "How did...?", "How should...?", and "Advice" labels.

tion method were to use theory and previous empirical work to derive specific hypotheses (see Table 1) related to gender issues in such environments, and to investigate whether these hypothesized issues really do arise in end-user programming. The result of the second step was confirmation that two hypothesized gender issues indeed exist in end-user programming (although the underlying cause for these gender differences is unknown). The third step, reported in this paper, was to develop solutions to address these issues. The changes were specifically aimed at the females who appeared to be facing barriers in using our earlier design, although we suspect that our changes will help both genders. The fourth step, using quantitative empirical methods to evaluate the effectiveness of the solutions, will be the subject of an upcoming study.

Our work resulted in two complementary solutions: a single-mouse-button "no confidence required" device to elicit inputs from low-confidence users that were then reflected in the feedback devices, and changes to our explanation system to support user-driven, non-linear exploration of the end-user programming devices in the system.

Our procedure for developing these solutions used theory, low-cost prototyping, and qualitative empirical work. Specifically, we showed how theories such as self-efficacy theory, minimalist learning theory, Norman's action cycle, and attention investment can be used to help understand barriers, derive requirements, and ultimately derive design ideas to address gender issues in end-user programming. Using the theory-derived design ideas, coupled with design techniques originally developed in HCI, we designed the potential specifics of our solutions, evaluated them analytically and through rapid prototyping, and informed our emerging approaches with a small stream of users. The solutions that resulted are the first to begin addressing gender differences through the design of features in end-user programming environments.

Acknowledgments

Our removal of the right click in Solution 1 builds on an earlier design proposed by Joseph Ruthruff. This work was supported in part by Microsoft Research, by NSF grant CNS-0420533 and by the EUSES Consortium via NSF grants ITR-0325273 and CCR-0324844.

References

[1] P. Ames, "Gender and learning styles interactions in student's computer attitudes", *J. Educational Computing Research*, 28, 3, 2003, 231-244.

[2] P. Anson, "Exploring minimalistic technical documentation design today: a view from the practitioner's window", In J. M. Carroll (Ed.), *Minimalism Beyond the Nurnberg Funnel*, MIT Press, Cambridge, MA, 1998, 91-117.

[3] I. Arroyo, "Quantitative evaluation of gender differences, cognitive development differences and software effectiveness for an elementary mathematics intelligent tutoring system", PhD Thesis, Univ. Mass. Amherst 2003, http://ccbit.cs.umass.edu/people/ivon/Dissertation80.pdf

[4] A. Bandura, "Self-efficacy: Toward a unifying theory of behavioral change", *Psychological Review*, 8, 1977, 191-215.

[5] J.E. Beck, I. Arroyo, B.P. Woolf, and C. Beal, "An ablative evaluation", *Ninth Int. Conf. Artificial Intelligence in Education*, 1999, 611-613.

[6] L. Beckwith and M. Burnett, "Gender: An important factor in end-user programming environments?" *IEEE Symp. Visual Languages and Human-Centric Computing*, 2004, 107-114.

[7] L. Beckwith, M. Burnett, S. Wiedenbeck, C. Cook, S. Sorte, and M. Hastings, "Effectiveness of end-user debugging software features: Are there gender issues?" *ACM Conf. Human-Computer Interaction*, 2005, 869-878.

[8] A. Blackwell, "First steps in programming: a rationale for Attention Investment models", *IEEE Symp. Human-Centric Computing Languages and Environments*, 2002, 2-10.

[9] J.D. Bransford, A.L. Brown, and R.R. Cocking, *How People Learn: Brain, Mind, Experience, and School*, National Academy Press, Washington DC, 1999.

[10] M. Burnett, J. Atwood, R. Djang, H. Gottfried, J. Reichwein, and S. Yang, "Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm", *J. Fun. Programming*, 11, 2, 2001, 155-206.

[11] M. Burnett, C. Cook, and G. Rothermel, "End-user software engineering", *Comm. ACM*, 47, 9, 2004, 53-58.

[12] T. Camp, "The incredible shrinking pipeline", *Comm. ACM*, 40, 10, 1997, 103-110.

[13] J.M. Carroll (Ed.), *Minimalism Beyond "The Nurnberg Funnel"*, MIT Press, Cambridge, MA: 1998.

[14] J.M. Carroll and M.B. Rosson, "Getting around the task-artifact cycle: how to make claims and design by scenarios", *ACM Trans. Information Systems*, 10, 2, 1992, 181-212.

[15] C. Gorriz and C. Medina, "Engaging girls with computers through software games", *Comm. ACM*, 43, 1, 2000, 42-49.

[16] A.J. Ko, B.A. Myers, and H.H. Aung, "Six learning barriers in end-user programming systems", *IEEE Symp. Visual Languages and Human-Centric Computing*, 2004, 199-206.

[17] J. Meyers-Levy and B. Sternthal, "Gender differences in the use of message cues and judgments", *J. Marketing Research*, 28, Feb 1991, 84-96.

[18] D.A. Norman, *The Design of Everyday Things*, Basic Books, New York, 1988.

[19] M.B. Rosson, J.M. Carroll, and R.K.E. Bellamy, "Smalltalk scaffolding: a case study of minimalist instruction", *ACM Conf. Human-Computer Interaction*, 1990, 423-429.

[20] G. Rothermel, M. Burnett, L. Li, C. Dupuis, and A. Sheretov, "A methodology for testing spreadsheets", *ACM Trans. Software Eng.ineering and Methodology*, 10, 1, 2001, 110-147.

[21] J.R. Ruthruff, S. Prabhakararao, J. Reichwein, C. Cook, E. Creswick, and M. Burnett, "Interactive, visual fault localization support for end-user programmers", *J. Visual Languages and Computing*, 16, 1-2, 2005, 3-40.

[22] H.A. Simon, "The structure of ill-structured problems", *Artificial Intelligence*, 4, 1973, 181-202.

[23] A. Wilson, M. Burnett, L. Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, and G. Rothermel, "Harnessing curiosity to increase correctness in end-user programming", *ACM Conf. Human Factors in Computing Systems*, 2003, 305-312.