In P. Romero, J. Good, E. Acosta Chaparro & S. Bryant (Eds). Proc. PPIG 17 Pages 242 - 254

Mining Qualitative Behavioral Data from Quantitative Data: A Case Study from the Gender HCI Project

Laura Beckwith[1], Thippaya Chintakovid[2],
Susan Wiedenbeck[2], and Margaret Burnett[1]

[1] Oregon State University
Department of Computer Science and Electrical Engineering
Corvallis, Oregon 97330
{beckwith, burnett}@eecs.orst.edu
[2] Drexel University
College of Information Science and Technology
Philadelphia, PA 19066
{thippaya.chintakovid, susan.wiedenbeck}@cis.drexel.edu

Abstract. Recent research has shown that gender differences exist that influence the ways that males and females work with problem-solving software. These gender differences may put females at a disadvantage in competing for jobs requiring these skills. Earlier research has shown the existence of gender differences in confidence that affects feature usage and adoption; however these findings have raised new questions. We are seeking answers to these questions through qualitative methods. The case study we present here documents our methodology and may be used as a guide for others embarking on similar qualitative analyses.

1 Introduction

Although there have been gender studies designed to understand and ameliorate the low representation of females in the computing field [9, 15], there has been little emphasis on software's *design attributes* and how these design attributes affect males' and females' performance in computing tasks. Building upon theories and research about gender differences from a number of domains [4], we have begun investigating whether there are features within software that interact with gender differences.

These investigations are just beginning, but there are already interesting results emerging. We carried out a study in which we gave male and female spreadsheet users two spreadsheet debugging tasks and an environment containing a number of features that support such debugging tasks. A summary of our three main findings is presented below. A more complete description of the experiment and the results can be found in [5].

- Females had lower self-efficacy (i.e. 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 were to accept the new debugging features. One reason females stated for this was that they thought the features would take them too long to learn. Yet, there was no real difference in the males' and females' ability to learn the new features.
- Although there was no gender difference in fixing the seeded bugs, females introduced more new bugs—which remained unfixed. This is probably explained by low acceptance of the debugging features: high effective usage was a significant predictor of ability to fix bugs.

The data collection mechanisms we used in the above study produced detailed data on the actions that participants engaged in and the time they spent on them while debugging. We decided that a more in-depth investigation of the participants' behaviors would provide further insights into gender differences surrounding their feature usage. Toward that end, we embarked on a qualitative investigation.

Looking at users' behavior using qualitative analyses (as in [23, 24]) normally includes users' verbal data sometimes in combination with their captured computer actions. However, in our study we have only users' computer actions, no verbal data as they worked through their task.

Going from a large collection of data gathered from the study to a qualitative investigation of some parts of it raises a number of issues to overcome and decisions to be made. In this paper, we present a case study of our journey down this path, with the issues we encountered and decisions we made highlighted along the way. The case study is of ongoing work, so the end of the story is not yet available. Still, in presenting the part of the work we have done so far, we hope to obtain useful feedback from others about the decisions we have made, and to share our experience with others who may find themselves in similar situations.

2 Experiment

A full description of the experimental design of our quantitative study can be found in [5]. Here we present only the portions of the methodology needed to understand the qualitative part of the study.

2.1 Participants and Procedures

The 27 male and 24 female participants (mostly business students) started by filling out a pre-session questionnaire which collected participant background data and included self-efficacy questions based on a slightly modified version of Compeau and Higgins' validated scale [12]. The following background data were collected: gender, major, year or degree completed, GPA, programming experience (to bar participants with more programming experience than is usual for business students), spreadsheet

experience, previous use of the study's prototype environment, and whether English was their primary language.

All participants received the same treatment; the only independent variable was gender. Each participant attended one session. The participants were seated one per computer in a small lab. After participants completed the questionnaire, we administered a 35-minute "hands-on" tutorial to familiarize participants with the environment. The participants were then given two spreadsheet debugging tasks. We captured their actions (mouse clicks, keystrokes, and the system's feedback) in electronic transcripts, as well as their final spreadsheets. At the conclusion of each task, we administered post-task questionnaires in which participants self-rated their performance on the task. The second task's post-session questionnaire also included questions assessing participants' comprehension of features in the environment.

2.2 Environment

The debugging features that were present in this experiment were part of WYSIWYT ("What You See Is What You Test"). WYSIWYT is a collection of testing and debugging features that allow users to incrementally "check off" or "X out" values that are correct or incorrect, respectively [8]. In addition, arrows that allow users to see the dataflow relationships between cells also reflect WYSIWYT "testedness" status at a finer level of detail.

The underlying assumption behind WYSIWYT is that, as a user incrementally develops a spreadsheet, he or she can also be testing incrementally. Figure 1 shows an example of WYSIWYT in Forms/3 [7], the research spreadsheet environment used in this experiment. In WYSIWYT, untested formula cells (i.e., cells with non-constant formulas) have red borders (light gray in this paper). Whenever users notice a correct value, they can place a checkmark (√) in the decision box at the corner of the cell they observe to be correct: this communicates a successful test. Behind the scenes, checkmarks increase the "testedness" of a cell according to a test adequacy criterion based on formula expression coverage (described in [21]), and this is depicted by the cell's border becoming more blue (more black in this paper). Also visible in the figure, the progress bar (top) reflects the testedness of the entire spreadsheet.

Instead of noticing that a cell's value is correct, the user might notice that the value

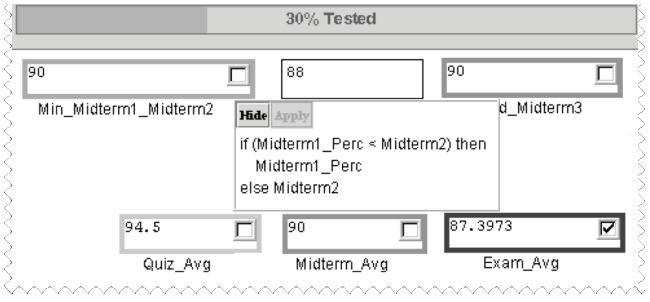

Figure 1. An example of WYSIWYT in Forms/3

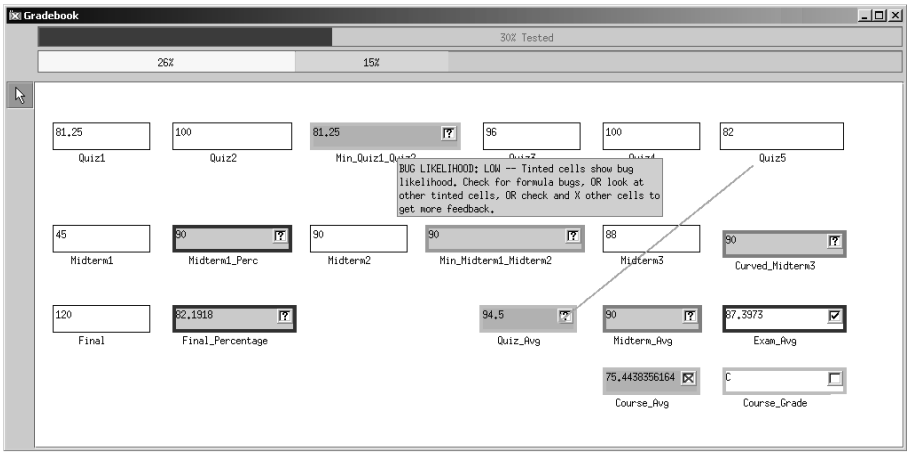

Figure 2. The user notices an incorrect value in `Course_Avg`—the value is obviously too low—and places an X-mark in the cell. As a result of this X and the checkmark in `Exam_Avg`, eight cells are identified as being possible sources of the incorrect value, with some deemed more likely than others. The (lower) progress bar reflects the current status of fault likelihood feedback

is incorrect. In this case, instead of checking off the value, the user can put an X-mark in the cell's decision box. X-marks trigger fault likelihood calculations, which cause cells suspected of containing faults to be colored in shades along a yellow-orange continuum (shades of gray in this paper), with darker orange shades given to cells with increased fault likelihood. Figure 2 shows an example of this behavior in one of the spreadsheets the participants debugged. The intent is to lead the user to the faulty cell (colored darkest orange).

The optional dataflow arrows are colored to reflect testedness of specific relationships between cells and subexpressions. (The user can turn these arrows on/off at will.) In Figure 2, the user has popped up Quiz5's arrow, which shows both that Quiz5 is referenced in Quiz_Avg's formula and that this relationship is not yet tested.

The way these features are supported is via the Surprise-Explain-Reward strategy [19, 22, 26]. If a user is surprised by or becomes curious about any of the feedback of the debugging features, such as cell border color or interior cell coloring, he or she can seek an explanation, available via tool tips (Figure 2). The aim of the strategy is that, if the user follows up as advised in the explanation, rewards will ensue [22]. Some of the potential rewards are functional—such as being led directly to a bug—and some are affective—such as increased progress in the progress bar. One aspect of interest in our quantitative experiment was whether, if gender differences in confidence were present, they might impact Surprise-Explain-Reward's success in encouraging users to approach and adopt new features.

2.3 Tutorial

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

Half of the tutorial sessions were presented by a male graduate student and half were presented by a female graduate student. This design ensured that approximately 50% of males were instructed by a same-gender instructor and 50% by an opposite-gender instructor (and likewise for the females) [25], serving to distribute any gender effect of the tutorial presenter equally over the two genders.

2.4 Tasks

The experiment consisted of two spreadsheets, `Gradebook` and `Payroll` (Figure 2 and Figure 3). To make the spreadsheets representative of real end-user spreadsheets, `Gradebook` was derived from an Excel spreadsheet of an (end-user) instructor, which we ported into an equivalent Forms/3 spreadsheet. `Payroll` was designed by two Forms/3 researchers using a payroll description from a real company.

These spreadsheets were each seeded with five faults created by real end users. From the collection of faults left in these end users' final spreadsheets, we chose five that provided coverage of the categories in Panko's classification system [18] (based upon Allwood's classification system [1]).

The participants were provided these `Gradebook` and `Payroll` spreadsheets and descriptions, with time limits of 22 and 35 minutes, respectively. The experiment was counterbalanced with respect to task order so as to distribute learning effects evenly. The participants were instructed, "Test the … spreadsheet to see if it works correctly and correct any errors you find."

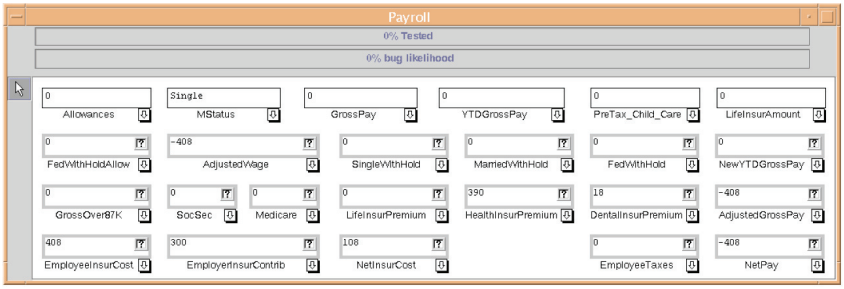

Figure 3. The Payroll spreadsheet

3 Development of Research Questions and Codes

The development of the research questions for our qualitative study was intertwined with and partially driven by the development of a set of codes to apply to the transcripts of participants' actions obtained from our study. The coding was a way of assigning each action, or set of actions, into categories that could later be used to answer questions about participants' behaviors. Developing the categories necessitated deriving research questions which we then used to determine the codes that would best allow us to answer those questions.

To develop our research questions, beyond our more basic questions of why we found differences, we relied on two procedures. The first was to develop our research questions based on casual observations the researchers had made during the quantitative study itself as the participants were working on the tasks and on the types of data collected in the transcripts. For example, when looking at why males had greater usage of checkmarks, we knew the transcript data would allow us to answer questions about whether the males placed checkmarks in cells even when doing so did not help them make testing progress. As our second procedure for generating research questions, we derived questions based on the theories that govern the way people problem solve with software. We drew from six theories (see Table 1) that suggested new research questions or tied into our existing research questions. The list of our research questions can be found in Table 2.

Once we had our questions we developed the codes. We kept the codes as simple as possible, containing information that would answer as many questions as possible with the fewest number of codes. It was also important that the codes be easy to apply, so that the two raters would be clear on which code applied to participants' actions, and further that no two codes with contradictory meaning would apply to the same participants' actions.

Before each rater began coding the transcripts, one of the raters tested the codes by applying them to one participant's transcript. With the insights gained from this procedure we refined the codes slightly to make them as straightforward to apply as possible.

In our experiment, two raters coded participants' actions taken to debug the spreadsheets and the changes that occurred subsequent to the debugging actions. For instance, our codings indicated when participants correctly or incorrectly placed checkmarks or X-marks and whether the checkmarks/X-marks were placed on a decision box with a questionmark (indicating that a user can make testing progress by making a decision about the cell's value), a blank decision box (indicating a decision has been made for a situation like this one), or a decision box with either a checkmark or X-mark (a decision has previously been made for this value). Moreover, the raters coded when participants introduced or fixed bugs, and whether participants read tooltip-based explanations (as opposed to merely causing explanations to appear by resting their mouse near them). The raters also coded changes in testedness of cells, as indicated by cell borders, and the percentage of likelihood of bugs in the spreadsheet appearing in the progress bar. Based on the codings, we expected to gain insights into the participants' debugging approach.

Table 1. A summary of the theories used for the development of our research questions

Theory	Description
Minimalist Learning	Minimalist learning is a method of introducing users to new aspects of a system through engaging them in activity. The theory is based on addressing motivation and cognitive issues within software development. The designer's focus is on the user's desire to accomplish a real task and balances this goal with the user's need to learn (make sense of) other helpful features of the software [10].
Self-efficacy	Self-efficacy is a person's judgment about his or her ability to carry out a course of action to achieve a certain type of performance. Bandura argues that achieving a desired type of performance depends on two factors, the skills needed to carry out the task and the perception of efficacy that will allow the individuals to use their skills effectively. High self-efficacy is critical in problem solving because self-efficacy influences the use of cognitive strategies, the amount of effort put forth, the level of persistence, the coping strategies adopted in the face of obstacles, and the final performance outcome [2, 3].
Attention Investment	The Model of Attention Investment is an analytic model of user problem-solving behavior that models the perceived costs, benefits, and risks users weigh in deciding how to complete a task [6].
Norman's Action Cycle	Norman's Action Cycle considers problem solving in two steps: *Execution* and *Evaluation/Feedback*. In the former users look to the environment for possible actions to move them closer to their goal; in the latter users determine if the result of their actions had the desired effect. When users do not have the necessary information to complete a step this is referred to as a *gulf* [17].
Diffusion of Innovation	The Diffusion of Innovation theory [20] describes how a new technology is adopted by society over time. People fall into 5 groups based on how soon they choose to adopt the new technology: Innovators, Early Adopters, Early Majority, Late Majority, and Laggards. Innovators are risk-taking technology enthusiasts, and Early Adopters tend to be visionaries who respond to the potential of the emerging technology. On the other end of the spectrum, the Late Majority is cautious, responding to pressure from peers and economic necessity; Laggards are even more skeptical and cautious, questioning both the intrinsic value of the technology and their own ability to benefit from it.
Information Gap	According to the Information Gap theory, when a person realizes they have an *information gap*, their curiosity about that information increases. At an optimal level of curiosity a person tries to fill their information gap by gaining more knowledge [14]. We take advantage of this theory in our Surprise-Explain-Reward methodology, attempting to surprise the users to raise their curiosity in some aspect of the software [26].

Table 2. Our research questions

General Question	Detailed Questions
1. Why did females introduce more bugs than males?	a. Is the time spent around a formula edit the same for both males and females?
	b. Do users attempt to fix introduced bugs?
	c. Do users test (using checkmarks and/or X-marks) cells with wrong values after a wrong formula edit?
	d. How long before the user comes back to the original correct formula?
2. Why did females use fewer checkmarks and arrows?	a. Do users place checkmarks on all cells (whether they could make testing progress by doing so)?
	b. What is users' mode of testing?
	c. Do users appear to be careful in checkmark placement?
	d. Do users get into a "testing" mode after making an edit?
	e. How are arrows used? (e.g., for testing purposes, to understand relationships among cells – unrelated to testing)
3. Why were females less engaged with X-marks?	a. Do users read explanations about interior colors after placing an X-mark?
	b. How long from viewing a tool tip about an interior color do users wait to take one of the suggested actions?
	c. What kind of feedback do users see on the screen after they place an X-mark? Does this appear to effect their decisions?
	d. Do users appear to contemplate more before placing an X-mark versus placing a checkmark?
	e. How long is the X-mark left onscreen (before the user removes it)?
	f. Do users appear to be wavering in their decision about the correctness of a cell (by hesitating between marking it correct and incorrect)?

4 Selection of Transcripts for Coding

Coding and analyzing data one-by-one for each of the original 51 participants would have required a huge amount of time, and did not seem likely to add valuable information beyond what we could learn from a subset of the original participants. Fortunately, unlike the random selection of participants for quantitative research, selecting participants for qualitative research is far less restrictive; researchers can seek out those participants with the greatest differences in specific areas of interest [16]. We selected our participants based on two main characteristics: (1) checkmark and arrow usage and (2) X-mark usage. Since we were most interested in participants with extreme usage patterns in checkmarks, arrows, and X-marks, we selected participants with high and low usage in these areas, without knowledge of the gender of the selected participants. The gender of these chosen participants was then checked by

another researcher not involved with applying the codings, to ensure a reasonable distribution by gender. We believed it was important that the two raters not know the gender of the participants in order to avoid bias in applying the codings and doing the early analysis of the data.

In the original statistical study that produced these data, there were 51 participants. Through the method described above we selected 22 participants. Both raters coded all the transcripts from the 22 participants, which took approximately 160 hours in total.

The raters applied the codes by stepping through each participant's actions (as documented in the transcripts) while watching the feedback the participants observed by "replaying" the transcripts. For example, if the participant's action was to place a checkmark on a cell, the transcript shows that the participant placed this mark, and by observing what the participant's spreadsheet looked like after that action, the rater was able to determine whether the checkmark was placed correctly or incorrectly. A second example is that the transcripts contained the information about when participants changed a cell's formula, and the raters would state whether this was an edit on a formula that had previously been correct (therefore, participants *introduced* a bug), or whether it was on a buggy cell, but the edit did not fix the bug (referred to as an *attempted fix*).

5 Inter-Rater Reliability

Once both raters had coded each transcript, one of the raters began the task of checking for reliability between the coded transcripts. *Inter-rater reliability* is a measure of the level of consistency among raters applying the same codes to the same data. It is calculated by counting the agreements and disagreements between the raters [11]. Differences can occur for many reasons, such as a poorly specified coding or differences in opinions. Determining the reliability is an important aspect of the qualitative analysis, since all future statements and conclusions on the meaning of the data are determined based on the codings. If the two raters have little agreement in their codings, the corresponding results and conclusions will not be valid.

In our coding we found an example of a code that was not specific enough and led to differences in the way each rater applied the code. The two raters applied the code for tool-tip-based explanations differently; one rater was more systematic in the application of the code while the other rater chose only to apply the code in the places where it was clear the participant was or was not reading the explanation. (When more than three explanations were displayed in one second we reasoned that the participants were probably not reading these. This is also supported by observations made by the researchers during the quantitative study itself.) If the specification of the code had been more specific, this difference in code application might not have occurred.

However, this unclear specification also brought to our attention the inferences and interpretation the raters had to make regarding the explanations. This same level of inference/interpretation was not required with the other codes.

In light of the differences in the way the raters applied the codings of tool-tip explanations and the interpretation required, we decided to determine the inter-rater reliability both including and excluding these explanations. The inferences on the part of the raters in applying the explanation codes meant that answering research questions relevant to explanations was based on data less reliable than the other research questions.

We conducted the reliability analysis in three steps (we are currently working on steps 1 and 2):

1. We counted agreements and disagreements between original codings by Rater1 and Rater2 (agreements were counted as coding the same line of the transcript exactly the same – all others were disagreements).
2. The two raters independently reviewed their decisions where disagreements existed. This review was done to detect and eliminate simple slips in coding. Changes were then made to the codings where slips occurred.
3. Cohen's Kappa statistic was applied to the changed codings.

Due to the considerable number of ratings made (around 100 per transcript) we expect there to be some substantial initial differences in Step 1, but we expect these differences will decrease in Step 2.

The formal statistical measure applied to the data is the Cohen's Kappa statistic. This statistic works by comparing the agreements and disagreements in ratings, and depends on the types of ratings. As a simple example, if there are three codes that can be applied, A, B, and C, then all of the agreements of AA coding need to be counted and all the disagreements of AB and AC counted separately. After counting these, some simple calculations can be applied to determine the overall reliability [11].

Our codes fall into two categories: objective and subjective. The objective codes are those that can be determined directly from the transcripts without the rater needing to make a judgment call on what the participant is doing. An example of this type of code is the type of mark a user placed, whether it was an X-mark or checkmark. A subjective code is one that cannot be directly determined from the transcripts and requires the rater's eyes and reasoning to make a judgment about the participant's actions. For example, whether or not a checkmark placed was correct or not is a subjective code because this information needed to be determined by the raters. For calculating Cohen's Kappa we will use only the subjective ratings the raters made and ignore the objective ratings.

The next step of analyzing the coded data relies on both the objective and subjective codes.

6 Analysis of Coded Data

Many methods for analyzing data exist [13, 16] that vary in their detail and level of interpretation of the data (e.g., whether to let the data speak for itself or use the data for the further development of theory). When we engage in our data analysis, we plan to follow the latter of the above descriptions, using the constant comparative method [16] to relate our findings to the theories in Table 1. Constant comparison is a process in which each participant is compared to each previously analyzed participant and

then sorted into categories (the categories are formed throughout the constant comparative analysis). As new participants are analyzed, further theories are generated. For our research questions we expect to link the categories directly to our theories (see Table 1), and additionally consider relationships between categories on different research questions.

Although we just referred to our unit of comparison as a participant, for some of the research questions a better-quality unit of comparison may be some set of actions that a participant may complete zero or more times. Before beginning our constant comparative analysis the unit of comparison for each research question will first be established. For example, for research question 2a the unit of comparison will be the set of actions where users are placing checkmarks on many cells all at once. In this example each participant may have zero or more units.

The following is our method for constant comparison analysis (these steps will be followed for each research question):

1. For the first unit placed into a category, describe the specifics of that category (what about this instance makes it standout)
2. For each unit after the first:
    a. Compare it with all other units already analyzed
    b. If it fits into an already created category, add it (specifying any changes made to that category to accommodate this specific unit). If it does not fit into an existing category, create a new category with the specific differences between this and the other existing categories.
3. Explore and document relationships and patterns across categories
4. Integrate data and theories to yield understanding of findings in relation to theories presented in Table 1.

7 Conclusion

This paper describes the methodology we followed (and are still engaged with) in a qualitative analysis of gender differences in problem-solving software features. This analysis is a follow-up investigation to previous quantitative research highlighting these gender differences. Since we were working with data collected before the design of the qualitative analysis some of the typical qualitative procedures needed to be adapted to accommodate the data we had collected. For example, collection of data in traditional qualitative studies can be modified as the study progresses to answer questions generated by the earlier findings. However, in our study all the data we had available to us had been collected during our quantitative analysis.

During each step of the analysis we minded reliability issues. In particular we took reliability in the raters' codings seriously since all future analysis relies upon these codings. Another reason for being careful and intentional in small choices is the enormous amount of time several of the steps took to complete. Both the codings and the inter-rater reliability combined will have taken us hundreds of hours to complete (when finished), which is prior to any of the analysis of the coded data.

Despite the extensive time involved in qualitative analysis, we expect a payoff that will enhance the data obtained from the quantitative study. We anticipate that our

findings will considerably increase our understanding of how the existing theories apply to gender differences in end-user problem-solving software.

8 Acknowledgements

We would like to thank Shraddha Sorte for her help in participant selection, and Curtis Cook for his assistance with general ideas regarding analysis. 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. Allwood, C.: Error detection processes in statistical problem solving. *Cognitive Science 8,* 4 (1984) 413-437
2. Bandura, A.: Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review 8*, 2 (1977) 191-215
3. Bandura, A.: *Social Foundations of Thought and Action*. Englewood Cliffs NJ: Prentice Hall, (1986)
4. Beckwith, L. and Burnett M.: Gender: An important factor in end-user programming environments? In Proc. *IEEE Symposium on Visual Languages and Human-Centric Computing* (2004) 107-114
5. Beckwith, L., Burnett, M., Wiedenbeck, S., Cook, C., Sorte, S., and Hastings, M.: Effectiveness of end-user debugging software features: are there gender issues? *ACM Conference on Human Factors in Computing Systems*, April 2005 (to appear)
6. Blackwell, A., First steps in programming: a rationale for Attention Investment models. In Proc. *IEEE Human-Centric Computing Languages and Environments,* (2002) 2-10
7. Burnett, M., Atwood, J., Djang, R., Gottfried, H., Reichwein, J. and Yang, S.: Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming 11*, 2 (2001) 155-206
8. Burnett, M., Cook, C. and Rothermel G.: End-user software engineering. *Communications of the ACM 47*, 9 (2004) 53-58
9. Camp, T.: The incredible shrinking pipeline. *Communications of the ACM 40*, 10 (1997) 103-110
10. Carroll, J.M. (ed.): *Minimalism beyond "The Nurnberg Funnel"*. Cambridge, MA: M.I.T. Press (1998)
11. Cohen's Kappa, http://www-class.unl.edu/psycrs/handcomp/hckappa.PDF accessed: January 28, 2005
12. Compeau, D. and Higgins, C.: Computer self-efficacy: development of a measure and initial test. *MIS Quarterly 19*, 2 (1995) 189-211
13. Dey, I.: *Qualitative data analysis: A user-friendly guide for social scientists*. London: Routledge, (1993)
14. Lowenstein, G.: The psychology of curiosity, *Psychological Bulletin 116*, 1 (1994) 75-98
15. Margolis, J., Fisher, A., Miller, F.: Caring about connections: Gender and computing, *IEEE Technology and Society Magazine 18*, 4 (1999) 13-20
16. Maykut, P, and Morehouse, R.: *Beginning Qualitative Research*, London: The Falmer Press, (1994)

17. Norman, D. A.: *The Invisible Computer: Why Good Products Can Fail, The Personal Computer Is So Complex, and Information Appliances Are the Solution*. Cambridge, MA, MIT Press, (1998)
18. Panko, R.: What we know about spreadsheet errors. *Journal of End User Computing 10*, 2 (1998) 15-21
19. Robertson, T. J., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, J., Beckwith, L. and Phalgune, A.: Impact of interruption style on end-user debugging. In Proc. *CHI 2004*, ACM Press (2004) 287-294
20. Rogers, E. M.: *Diffusion of Technology*. 4th Edition. New York, the Free Press, 1995.
21. Rothermel G., Burnett M., Li L., Dupuis, C. and Sheretov, A. A methodology for testing spreadsheets, *ACM Transactions on Software Engineering and Methodology 10*, 1 (2001) 110-147
22. Ruthruff, J., Phalgune, A., Beckwith, L., Burnett, M. and Cook, C.: Rewarding 'good' behavior: End-user debugging and rewards. In Proc. *IEEE Visual Languages and Human-Centric Computing* (2004) 115-122
23. Scholtz, J. and Wiedenbeck, S.: Using unfamiliar programming languages: The effects of expertise. *Interacting with Computers 5,* 1 (1993) 13-30.
24. von Mayrhauser, A. and Vans, A.M.: Program understanding behavior during debugging of large scale software. In Proc. *Empirical Studies of Programmers*, ACM Press (1997) 157-179.
25. Whitworth, J. E., Price, B. A. and Randall, C. H.: Factors that affect college of business student opinion of teaching and learning. *Journal of Education for Business 77*, 5 (2002) 282-289
26. Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., Durham, M. and Rothermel, G.: Harnessing curiosity to increase correctness in end-user programming. In Proc. *CHI 2003*, ACM Press (2003) 305–312