AN ABSTRACT OF THE THESIS OF

Valentina Grigoreanu for the degree of Master of Science in Computer Science presented on June 19, 2007.

Title: Complex Patterns in Gender HCI: A Data Mining Study of Factors Leading To End-User Debugging Success for Females and Males.

Abstract approved:

_____

Margaret M. Burnett

Most of the work so far in the subfield of Gender HCI has followed a theory-driven approach. Established theories, however, do not take into account specific issues that arise in end-user debugging. We suspected that there may be important information that we were overlooking. We therefore employed a methodology change: turning to data mining techniques to find hidden patterns and relationships in females' and males' feature usage patterns. This thesis reports two data mining studies to help discover complex ties among static, dynamic, and success data collected in end-user debugging sessions. Study 1 was our first step, and was used to derive new hypotheses about females' and males' strategies and behaviors. In Study 2, we then applied different data mining algorithms to a larger data set to describe, summarize, segment, and detect interesting patterns. We found that most of the factors that tied with females' success in debugging were different than those that tied with males' success in debugging and vice versa. The results will ultimately help Gender HCI researchers better support end-user debuggers of both genders.

Complex Patterns in Gender HCI: A Data Mining Study of Factors Leading To End-User Debugging Success for Females and Males

by

Valentina Grigoreanu

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 19, 2007
Commencement June 2008

Master of Science thesis of Valentina Grigoreanu presented on June 19, 2007

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Valentina Grigoreanu, Author

ACKNOWLEDGEMENTS

To my advisor, Dr. Margaret Burnett, thank you for your continued support and friendship through both good and bad times. You have every quality that I wanted my advisor to have. Thank you especially for throwing me into diverse high-responsibility tasks from the very beginning. The lessons I have learned from you about research are indispensible.

Study 1 of this Thesis was a joint effort by researchers at Oregon State University and the Oregon Institute of Technology. Thank you to Dr. Xiaoli Fern and Chaitanya Komireddy for conducting the data mining part of Study 1. Also thank you to Dr. Margaret Burnett, Dr. Curtis Cook, Laura Beckwith, Sherry Yang, and Vaishnavi Narayanan for their hard work and insights on the Gender HCI part of the thesis. Also thank you to Dr. Susan Wiedenbeck for her help related to this work and ongoing wisdom throughout our Gender HCI project.

To Dr. Carlos Jensen, thank you for all of your support and great feedback on my Thesis. Thank you also for introducing me to different areas in Gender HCI where similar data mining techniques could also be applied. Dr. Cook, thank you for the all of the thoughtful input for both Study 1 and Study 2 of my Thesis. You have been a consistent source of great ideas over the past two years. Also thank you to Dr. Robert Higdon for your feedback and contributions as a part of my committee.

Most importantly, thank you to my parents. Nothing would have been possible without their unconditional love and kind words and deeds. 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 grant ITR-0325273.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX TABLES

COMPLEX PATTERNS IN GENDER HCI:
A DATA MINING STUDY OF FACTORS LEADING TO END-USER DEBUGGING SUCCESS FOR FEMALES AND MALES

1. INTRODUCTION

1.1. Gender HCI: Motivation and Overview

Research from several domains has shown gender differences that are relevant to computer usage (Beckwith, Burnett and Wiedenbeck, et al. 2005), (Busch 1995), (Huff 2002). Although there has been a fairly wide interest in gender differences in computing professions and education, as well as in gaming, there has not been much research on how gender differences interact with end users' use of purportedly gender-neutral software features.

Beckwith and Burnett first defined the term *Gender HCI* in 2004 (Beckwith and Burnett 2004). *Gender HCI* is a subfield of Human-Computer Interaction that focuses on design and evaluation of interactive systems for humans, with emphasis on differences in how males and females interact with computers. It investigates ways in which attributes of software (or even hardware) can interact with gender differences.

Beckwith et al. began a line of Gender HCI work whose focus is on features in tools used for end-user software development, aiming to learn how to design end-user programming environments that support end-user programmers of both genders. This thesis continues this effort through two studies on gender differences in feature usage in end-user programming environments. Our analyses were conducted using data mining.

1.2. Data Mining: Motivation and Overview

Most of our work so far in this area has followed a theory-driven approach, in which theories from psychology, education, and HCI have been used to generate hypotheses which have then been investigated via empirical studies. However, a disadvantage in deriving empirical hypotheses from only established theories is that these theories do not take into account the specific needs and issues that arise in end-user programming. Research situations such as this are often referred to as "ill-structured" problems (Simon 1973). Such problems contain uncertainty about which concepts, rules, and principles are pertinent to it. Further, the "best" solutions to ill-structured problems depend on the priorities underlying the situation. In such problems, in addition to hypothesis testing and application, there is also the need for hypothesis generation. Such problems are candidates for ultimately deriving new theories from data and patterns.

Toward this aim, our research group previously used manual qualitative analysis techniques (Strauss and Corbin 1998), inspecting data on software feature usage in search of useful patterns leading to hypotheses. Although the results of these efforts have been fruitful, still, as humans we are fallible, especially given large amounts of detailed data. We suspected that there may be important information that we were overlooking. Therefore, we employed a methodology change: turning to data mining techniques to find feature usage patterns that we may have missed; we conducted two Gender HCI data mining studies.

In Study 1, we reported the results of revisiting data we had already analyzed in a previous study, using a data mining approach. Our aim was to derive new hypotheses about females' and males' strategies, adding to the growing foundation for

understanding gender differences in end-user programming situations—by "listening" to the participants, through their data, from the ground up (Grigoreanu, et al. 2006).

In Study 2, we further applied data mining algorithms to describe, summarize, segment, and find other interesting patterns in our data (see Chapter 3). Briefly, the main goals of Study 2 were to: (1) use a bigger set of data, (2) employ different data mining methods, (3) decrease the amount of grouping, and (4) employ statistical methods to validate the trustworthiness of the resulting models.

Employing data mining techniques to analyze Gender HCI data should provide a deeper understanding of hidden patterns and relationships that would otherwise be hard to hypothesize about. These patterns will help Gender HCI researchers better understand how females and males problem-solve differently.

1.3. Using a Standardized Model

A further difference between Study 1 and Study 2 is that we chose to use a standardized data mining process model for Study 2. The general data mining steps are agreed upon by researchers and many perform these steps in a similar order without following a standard data mining process model. My reasons for using a standardized data mining process model in Study 2 are that: (1) standardized processes help both experts and non-experts alike by providing a checklist of steps to not overlook and (2) this checklist doubles as a set of guidelines for conducting data mining studies.

In addition, this process allowed me to create a specific instance of the process model for future Gender HCI researchers to employ. The data mining process model that we customized to fit the needs of Study 2 is called the Cross Industry Process Model for Data Mining and is described in the next section.

1.3.1. The CRISP-DM Standardized Model

A standardized six-step model for data mining processes exists, with minor variations in some cases. The Cross Industry Process Model for Data Mining is the most prevalent of such models (KDnuggets 2002). CRISP-DM was developed by DaimlerChrysler AG, SPSS, NCR, and OHRA (Wirth and Hipp 2000). A very similar process model is described by SQL Server Books Online. Figure 1 below shows both of these models.

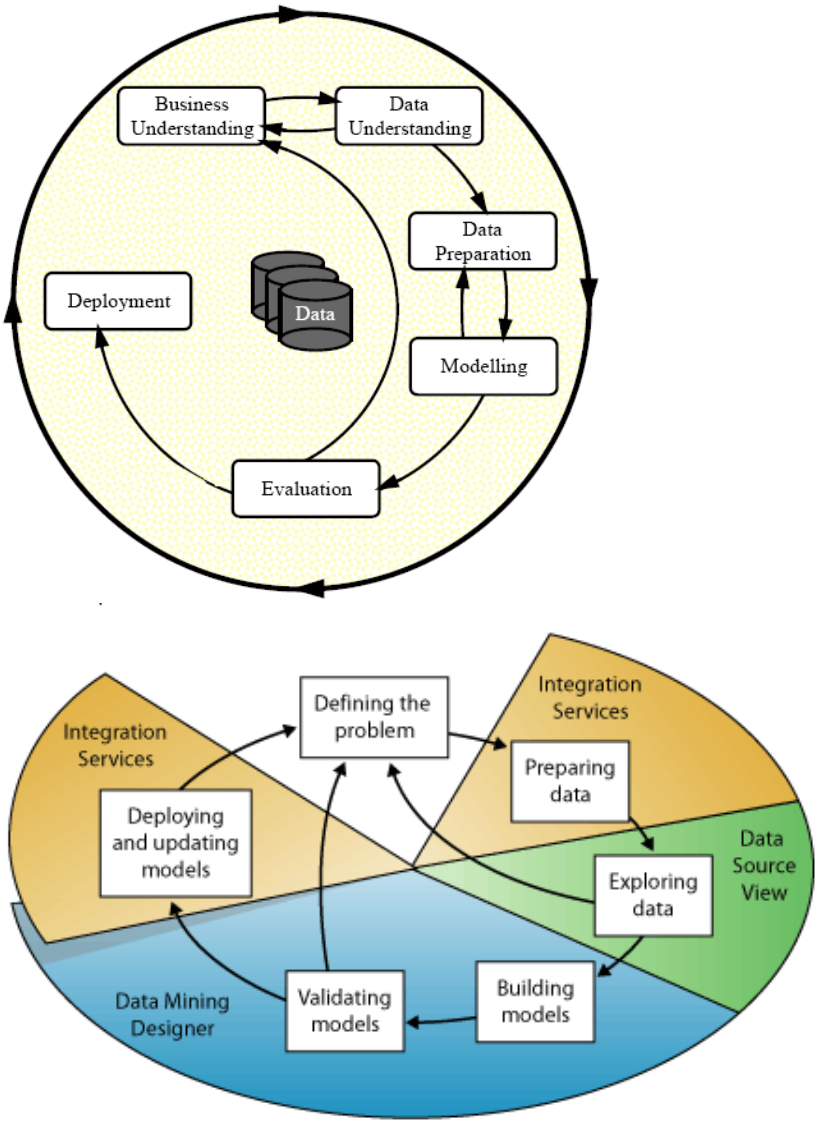


Figure 1. The CRISP-DM model (top) and the SQL Server Analysis Services model (bottom).

Our reason for including the SQL Server model here, alongside CRISP-DM, is that we used MS SQL Server Analysis Services 2005 for this study (see Section 5.1.4. for more information about the software we used). The SQL Server model was created

with the added goal of showing which of their products and services can help users at the various stages of the data mining process. While slight differences exist between the two models, they are very similar overall.

1.3.2 CRISP-DM Steps

This thesis is built around these process models. Notice that most of this thesis' chapter headings start with the data mining process model step that it pertains to. Here is a brief overview of each step.

Business Understanding (Defining the Problem). The first phase of a data mining project is to define the problem that a business wishes to address. In research studies, related work and research questions fit under this step. This is when a preliminary project plan is designed, with the project objectives and requirements in mind (see Chapters 2, 4, 5).

Data Understanding (Exploring the Data). The second phase includes first collecting the data and then exploring and becoming familiar with them. Exploration techniques include methods like calculating the mean, median, minimums, and maximums, and looking at the distribution of data (see Chapters 6, 7, 8, and Appendix B).

Data Preparation. Data preparation includes data consolidation and data cleaning. The data may be scattered among several studies or parts of a company (this is where consolidation comes in). Furthermore, there might be inconsistencies in the protocols by which data were recorded, compacted, or "scored" (which is why cleaning is needed). This step gets rid of the inconsistencies in the combined dataset (see Chapters 7, 8, and Appendix A).

Modeling (Building Models). After the data are combined and cleaned, models can be built using them. The data first need to be split into a training set and a

testing set. If the original dataset is big enough, it is better to split it into three sets: training, validation, and testing. This is not the case for either Study 1 or Study 2, since the datasets are too small. Competing models are then built on the training set, using different algorithms and parameter choices (see Chapters 3, 8, 9, Appendix C, and Appendix D).

Evaluation (Validating Models). Often, more than one model is created. This step evaluates which model performs best and determines how well that model performs. The testing set can be used to validate the models built on the training set. This step also determines whether the model properly answers the research questions posed in the first step (see Chapter 9).

Deployment (Deploying and Updating Models). Once the best model is picked, the results need to be deployed. The deployment step can range from writing a report to implementing a package that allows the data mining process to be directly repeated from within an application (all chapters).

Almost every step in this model is heavily tied to the others. The steps can be repeated multiple times and in various orders. For example, preparing data leads to a better understanding of them. A resulting mining model can produce a better problem definition, which helps build a better model. Building a model can also point out data that we missed in the preparation phase. During studies, it is recommended to explicitly allot time for three iterations of each step, with the second taking half the time of the first, and the third taking a quarter of the time of the first (Wirth and Hipp 2000). As with all data analysis, findings from one study will raise many other related research questions to be addressed in future studies. Thus, once a project is deployed, this often results in more specific research questions and the cycle repeats.

In addition to each chapter being titled with the step that it refers to, each chapter will also begin with a table similar to Figure 2. The part that is highlighted is the section that the chapter relates to. We have highlighted the Deployment step here as an example since the deployment phase consists of writing a report (this entire thesis).

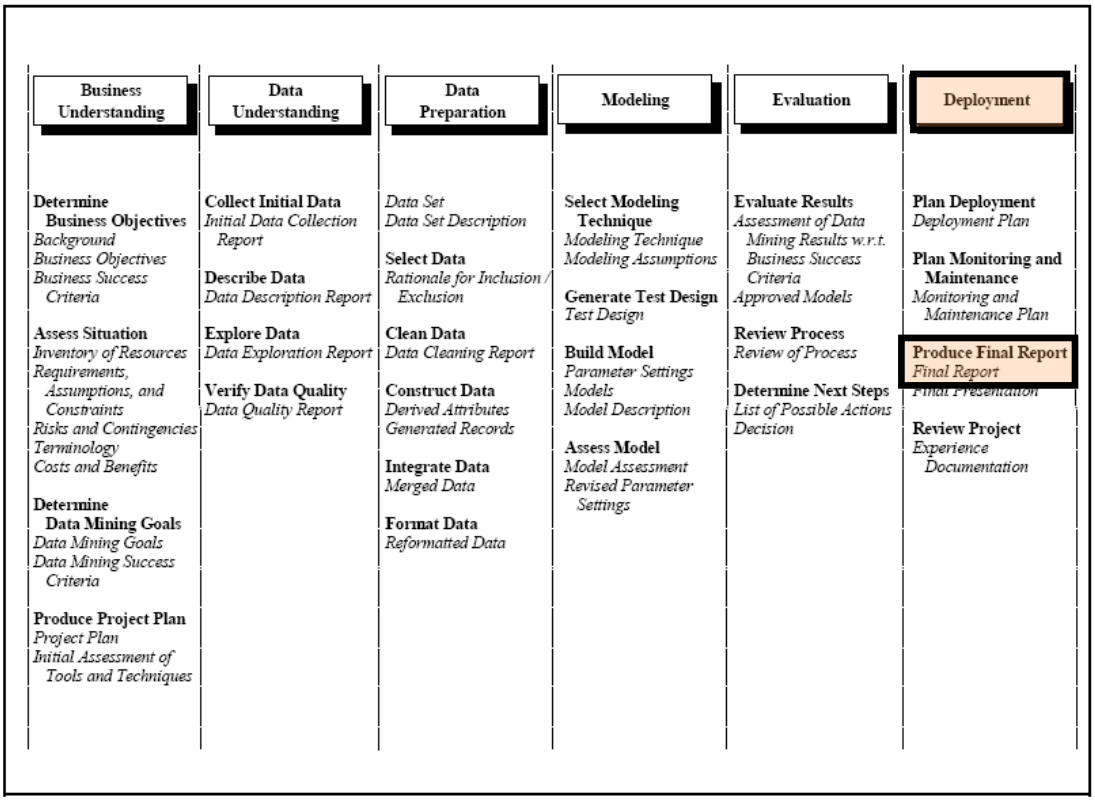


Figure 2: These are the six steps of the CRISP-DM data mining process model and the subparts and their outputs. This thesis is a final report of two Gender HCI data mining studies and the deployment phase is highlighted here as an example.

2. (BUSINESS UNDERSTANDING) GENDER HCI BACKGROUND AND RELATED WORK

Gender HCI research has been conducted in the following areas (among others): the effects of confidence and self-efficacy on both genders' interactions with software, the design of gender-specific software, such as video games created for girls, the design of display screen sizes and how they affect both genders, and the design of gender-neutral problem-solving software.

The subfield of Gender HCI is a highly interdisciplinary area. Findings from fields such as Psychology, Computer Science, Marketing, Neuroscience, Education, and Economics strongly suggest that males and females problem solve, communicate, and process information differently. Gender HCI investigates whether these differences need to be taken into account in the design of software and hardware. The term Gender HCI was first coined in 2004 by Beckwith and Burnett, but research relevant to that topic predates the term.

Some of the findings of Gender HCI research are related to male and female confidence in dealing with computer software (or computer self-efficacy). Self-efficacy is measured using a standard pre-study questionnaire. For example, for spreadsheet problem-solving tasks, (1) female end users had significantly lower self-efficacy (a task-specific form of confidence) than males and (2) females with low self-efficacy were significantly less likely to work effectively with problem-solving features available in the software. In contrast, males' self-efficacy did not impact their effectiveness with these features (Beckwith, Burnett and Wiedenbeck, et al. 2005).

In a study of the computer attitudes and self-efficacy of 147 college students, gender differences existed in self-efficacy for complex tasks (such as word processing and spreadsheet software), but not simpler tasks. Also, male students had more experience working with computers and reported more encouragement from parents and friends (Busch 1995).

Another category of findings relate to what kinds of features were used in software systems. For example, in spreadsheet problem-solving tasks, female end users took significantly longer before trying out unfamiliar features (Beckwith, Burnett and Wiedenbeck, et al. 2005). Also, females significantly more often agreed with the statement, "I was afraid I would take too long to learn the [untaught feature]." Even if they tried it once, females were significantly less likely to adopt new features for repeated use. For females, unlike for males, self-efficacy predicted the amount of effective feature usage. There was no significant difference in the task success of the two genders or in learning how the features worked, implying that females' low self-efficacy about their usage of new features was not an accurate assessment of their problem-solving potential, but rather became a self-fulfilling prophecy about their use of features (Beckwith, Burnett and Wiedenbeck, et al. 2005).

There is also a "how" part to software feature usage. In spreadsheet problem-solving tasks, tinkering (playfully experimenting) with features was done by males more often than females. Males were comfortable with this behavior; in fact, some did it to excess. For females, the amount of tinkering predicted success, but for males, excessive tinkering hurt them. Pauses after any action were predictive of better understanding for both genders (Beckwith, Kissinger, et al. 2006).

Another finding related to the participants' behavior is that males viewed machines as a challenge, something to be mastered, overcome, and be measured

against. They were risk-takers, and they demonstrated this by eagerly trying new techniques and approaches. Females rejected the image of the male hacker as alienating and depersonalizing. Their approach to computers was "soft;" tactile, artistic, and communicative (Turkle 1988).

Much of the research in Gender HCI has been about video games. Several findings were reported about girls' interests that relate to video games, with implications for the video game software industry (Gorriz and Medina 2000). Researchers explored what girls seek in video games, and implications for video game designers. Among the implications were collaboration vs. competition preferences, and use of non-violent rewards versus death and destruction as rewards. These works argue both sides of the question as to whether or not to design games specifically for girls (Cassell 1998) (Cassell and Jenkins 1998).

Not all of the findings have been about software; Gender HCI research has also been conducted in the hardware realm. Larger displays helped reduce the gender gap in navigating virtual environments. With smaller displays, males' performance was better than females'. With larger displays, females' performance improved and males' performance was not negatively affected (Czerwinski, Tan and Robertson 2002), (Tan, Czerwinski and Robertson 2003).

Other studies have been related to Internet behavior and perceptions. For example, in a study of the way people interacted with conversational software agents in relation to the sex of the agent, the female virtual agent received many more violent and sexual overtures than either the male one or the gender-free one (a robot) (De Angeli and Brahnam 2006). Other examples are that males and females had different perceptions for whether a webpage would be appropriate for his/her home country,

and that females more often than males preferred more information on all web pages viewed during a study (S. Simon 2001).

In the home, where many appliances are programmable to some extent, different categories of appliance were found to be more likely to be programmed by men (e.g. entertainment devices) than by women (e.g. kitchen appliances). There was often one member of a household who assumes responsibility for programming a particular device, with a "domestic economy" accounting for this task (Rode, Toye and Blackwell 2004).

3. (MODELING) STUDY 1: PRELIMINARY GENDER HCI DATA MINING STUDY

Study 1 was our first Gender HCI data mining study. In this study, we applied sequential pattern mining to our log files to search for potentially interesting patterns in data that had previously been collected. This data came from the Summer 2005 Gender Tinkering study's Treatment group. To get more information on the setup for that study and a detailed description of the events mentioned in this chapter, see Section 6.2.2.

Using a data mining approach, we focused on gender differences in *how* features are used, with the aim of gaining new insights into our previous reports of *when* and *how much*. Our aim was to derive new hypotheses about females' and males' strategies, adding to the growing foundation for understanding the gender differences in end-user programming situations—by "listening" to the participants, through their data, from the ground up.

3.1. Study 1: The Pattern Mining Process

In this study, we looked at *how* features were used by finding patterns in common sequences of events used by participants. We considered each user action as an *event*. This abstraction transformed the data into *sequences* of events. Participants in our empirical studies can perform several events to help test and debug their spreadsheets: Tooltips, Checkmarks, X-Marks, Arrow Operations, Value and Formula Edits, and Help Me Test. See Chapter 6 for details about the testing and debugging

features available in our Forms/3 research spreadsheet software. Thus, one example of a sequence of user events is: (Tooltip Showing, Checkmark, Checkmark).

3.1.1. Preprocessing into Debugging Sessions

Following the procedure of (Ruthruff, Burnett and Rothermel 2005), we used the notion of *debugging sessions* to break the sequence of events into subsequences. As with Ruthruff et al.'s definition, a debugging session ends with a formula edit (or at the end of the experiment), which presumably represents an attempt to fix a bug. However, unlike Ruthruff et al.'s definition, in which a debugging session began with the placement of an X-mark, our debugging sessions begin as soon as the previous one ends (or at the beginning of the experiment), so that all actions could be considered—not just the subset following an X-mark. In some cases participants edited the same formula multiple times consecutively. Since such edits were obviously a continuation of fixing the same bug, we included them in the preceding debugging session. Based on this definition, we broke each log file into debugging sessions.

3.1.2. Sequential Pattern Mining

We used the SLPMiner program (Seno and Karypis 2002) to search for patterns of the form (A, B, C), where A, B, and C are events that happened in the specified order. A debugging session was considered to contain the pattern (A, B, C) if it had at least one occurrence of events A, B, and C in that order, but the events did not need to be consecutive. For instance, one of the patterns that appeared in 68 debugging sessions is (checkmark, arrowon, arrowoff, postformula), meaning that the user placed a checkmark, worked with arrows and then opened a formula, with some other actions in between.We refer to the percentage of all debugging sessions that contained a pattern as the support of the pattern.

SLPMiner searches for all sequential patterns whose support exceeds a pre-specified threshold, and these patterns are referred to as frequent patterns. To avoid redundancy due to the fact that any subsequence of a frequent pattern will also be a frequent pattern, the software output the maximal patterns, i.e., patterns that are not subsequences of other frequent patterns. We chose the support threshold to be 10%, i.e., a pattern had to be contained in more than 10% of the 641 debugging sessions to be output by SLPMiner. This relatively low threshold was chosen because it allowed us to find patterns that were common to multiple users while still containing some of the interesting but less frequently used features such as X-marks and Arrow operations. The threshold did however get rid of some of the patterns that were "flukes". We focused our attention on patterns of limited size, in particular of length between one and four, because without limitations there would simply be too many patterns to process, and longer patterns often contained cyclic behavior and were difficult to interpret.

3.1.3. Output and Post-processing

From the 641 debugging sessions, SLPMiner found 107 patterns of length one (such as "HMT") through four (such as "Post Formula, Edit Value, Checkmark, Hide Formula"). Note that SLPMiner (and other sequential pattern mining algorithms) can only find "frequent" patterns, i.e., those satisfying the minimum support criterion, which was 10% in our case. It was up to us to determine which of the found patterns were interesting to our research goal.

Toward this aim, for each pattern, we computed its occurrence frequency for each user as the percentage of that user's debugging sessions that contained the pattern. For example, if user A had 20 debugging sessions and 10 of them contained pattern p, the occurrence frequency of pattern p for user A was 50%. As a result, we

obtained a pattern occurrence frequency table, which provided a comprehensive description of the distribution of the pattern occurrence among all users. We then analyzed these pattern occurrence frequencies in relation to the gender, task performance, and self-efficacy of the participants who used them.

To help analyze the pattern occurrence frequencies in an organized manner and gain a high-level understanding of the patterns, we categorized the found patterns such that each category contained patterns centered on a certain set of features. We then grouped them based on the features that they contained. When patterns contained more than one event of interest, we created a new category for it to make sure that the categories did not overlap. This process resulted in 9 categories. Figure 3 shows how the 107 patterns fell into these nine non-overlapping categories. For example, a pattern that contains arrow events, formula events, and tooltip events, would fall under the "Arrow, Formula & Tooltip" category, but not the "Arrow & Formula" category. See Table 1 for examples of patterns and their categories. Our analysis described in the following sections will be presented based on these categories.

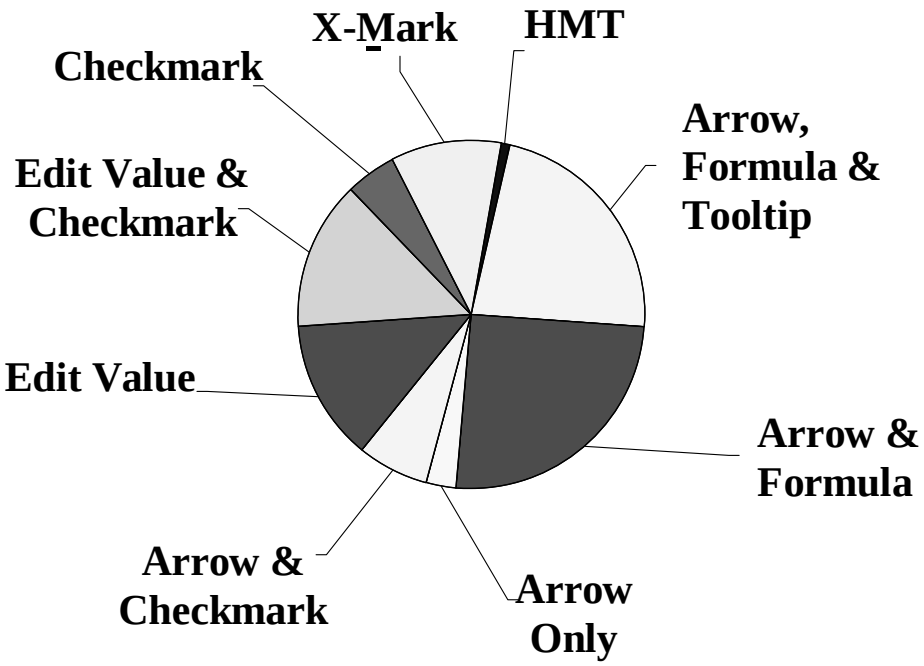


Figure 3: We grouped the 107 patterns into these 9 categories. The categories, based on the patterns' content, are focused on the debugging and other features available in the environment.

Table 1: Each of the nine categories contained patterns that only had the features mentioned in the category name as a part of them. For example, the Arrow & Formula category contained patterns such as "Arrow Off, Post Formula, Hide Formula, Post Formula." If one of the events in the pattern was a Tooltip event, however, then the pattern now fell into the "Arrow, Formula & Tooltip" category.

Category	**Example Pattern**
Help Me Test (HMT)	(HMT)
Arrow, Formula & Tooltip	(Tooltip Showing, Arrow On, Arrow Off, Edit Formula)
Arrow & Formula	(Arrow Off, Post Formula, Hide Formula, Post Formula)
Arrow Only	(Arrow On, Arrow On)
Arrow & Checkmark	(Hide Formula, Checkmark, Arrow On)
Edit Value	(Edit Value, Edit Value)
Edit Value & Checkmark	(Post Formula, Edit Value, Checkmark, Hide Formula)
Checkmark	(Checkmark, Tooltip Showing, Tooltip Showing, Checkmark)
X-Mark	(Hide Formula, X-Mark, Post Formula, Edit Formula)

3.2. Study 1: Results about How Each Gender Pursued Success

How did the successful versus unsuccessful females and males go about debugging? "How" in this question means the counts of each type of pattern, rather than the count of events performed (or the "what").

To investigate this question, we divided the 39 participants (16 males and 23 females) into four groups by gender and number of bugs fixed. We considered a participant "successful" if they fixed at least 7 of the 10 bugs, where 6 was the median number of bugs fixed. See Table 2 for the distribution of subjects by bugs fixed. The

groups and number of participants are displayed in Table 3. Arrow counts are also displayed in the table. Since earlier studies were about what features are used and how much, we did not look for that in this study, where we just looked at *how* features were used. Sometimes, feature counts helped us better understand pattern usage however. We will return to these later in this chapter.

Table 2: Distribution of participants by number of bugs fixed.

Nr. of Bugs Fixed	0	1	2	3	4	5	6	7	8	9	10
Nr. of Participants	3	1	1	3	4	4	5	5	7	4	2

Table 3: What (not How): The median number of arrows turned on and off during the experiment by gender and debugging success (count of feature usage). There is an especially big difference between the successful and unsuccessful males.

Group	Number of participants	Arrows
Successful Females	8	17.5
Unsuccessful Females	15	24
Successful Males	10	12
Unsuccessful Males	6	25.5

3.2.1. Just Like Males: A Female Success Strategy?

Strikingly, in Figure 4 the unsuccessful females and successful males showed the most similar frequency of pattern usage ("how") profiles for each of the five categories on the left half of the graph (from Edit Value to HMT)—all of which are testing-oriented activities. (We follow the software engineering definition of "testing" here: judging the correctness of the values produced by the program's execution.) The pattern usage was also similar between successful males and unsuccessful females for the other categories.

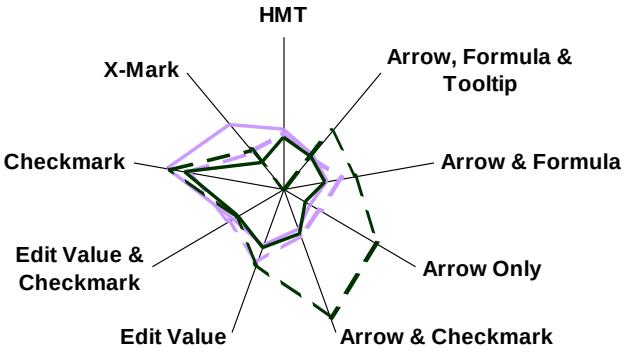


Figure 4: How (count of pattern usage) by success group. Successful: solid line, unsuccessful: dashed line, females: light, males: dark. (Each category is represented by an axis line radiating from the center. Where the polygon crosses an axis represents the frequency of that pattern.)

This suggests that the ways males successfully went about their debugging task are the very ways that did not work out well for the females, leading to the following hypothesis:

> **Hypothesis:** The debugging and testing strategies that help with males' success are not the right ones for females' success.

While there is a clear difference between successful and unsuccessful behaviors for males, this difference disappears for females.

3.2.2. Unsuccessful Males Like Arrows

Turning to the right half of Figure 4, which represents arrow-oriented patterns, the successful and unsuccessful females converge with the successful males. Interestingly, regarding this "how" aspect of arrows, there was a striking difference in the number of arrow patterns between successful and unsuccessful males. This difference is further illustrated by Figure 5, which shows that, unsuccessful males used "Arrow Only" patterns far more frequently (in more debugging sessions, on average) than the successful males.

The higher frequency of arrow patterns for unsuccessful males coincides with a higher raw count of arrows used. As Table 3 shows, successful males used a median of 12 arrows, whereas unsuccessful males used more than twice as many, 25.5.

Hypothesis: Unsuccessful males overdo use of arrows—unlike successful males, successful females, or unsuccessful females.

3.2.3. Unsuccessful Males: Tinkering Addicts?

We suspected that gender differences in tinkering behavior may be a factor in observed pattern differences. In particular, the unsuccessful males' more frequent use of arrows and their greater variety of arrow-related patterns is suggestive of a larger picture of unsuccessful males tinkering with arrows, to their detriment.

In fact, in previous work, we reported results in which males were found to do more unproductive tinkering, using a different environment (Beckwith, Kissinger, et al. 2006). However, the definition of tinkering used in that paper was necessarily simple—and its simplicity prevented it from capturing the excessive exploring/playing the unsuccessful males did. Based on patterns found via mining that data, we are now able to identify more complex tinkering behavior of unsuccessful males in this environment, which we failed to notice in our previous study.

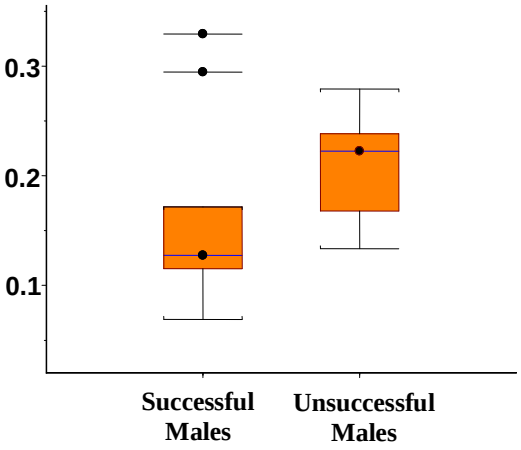


Figure 5: How: Percentage of debugging sessions that contained "Arrow Only" patterns in the by successful versus unsuccessful males.

For example, referring to Figure 5, notice the large differences in "Arrow Only" pattern usage for unsuccessful versus successful males. This category contains patterns that involve only arrow operations. Two representative patterns in this category were (Arrow Off, Arrow On) and (Arrow Off, Arrow Off). Unsuccessful males used these patterns often—in one out of every four debugging sessions for the unsuccessful males versus only one out of 20 for the successful males.

Hypothesis: Unsuccessful males have a tendency to tinker excessively with the features themselves rather than using the features to accomplish their task.

3.3. Study 1: Results about Self-Efficacy

Self-efficacy measures a person's belief in his or her ability to perform a particular task (Bandura 1986). Half of the 12 high self-efficacy females were successful but only two out of 11 low self-efficacy females were successful. However,

it was not true for males: seven out of 10 high self-efficacy males were successful and half of the low self-efficacy males were successful.

How do high and low self-efficacy females and males go about debugging? Since self-efficacy did not give the same groupings of the participants as given by task success, it is useful to consider how self-efficacy related to pattern choices.

To investigate the question of whether self-efficacy played a role in pattern usage, we divided the participants into four groups based on their self-efficacy scores. In particular, we considered a participant to have high (low) self-efficacy if her or his score was higher (lower) than the median of all participants. See Table 4 for the grouping of the participants.

We turned to median raw counts of the number of features used (the "what") to better understand the reasons behind the patterns that we were seeing (the "how"). Low self-efficacy female feature counts (Table 4) revealed that low self-efficacy females were the highest usage group for all of the features—except the checkmark.

Table 4: What by self-efficacy group. These are about the number of features used, rather than how they were used, to supplement our understanding of the sequential patterns. We divided the participants into four groups based upon their gender and pre-task self-efficacy. The rest of the table shows median raw counts of the number of testing features used during the experiment.

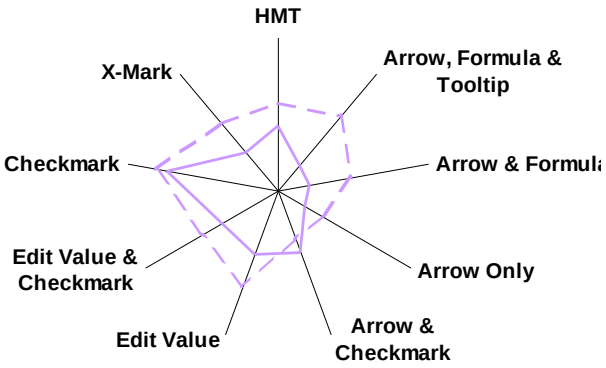
Group	**Number of participants**	**Arrow**	**X-mark**	**Checkmark**	**HMT**
High Females	12	10.5	3	65.5	5
Low Females	11	24	8	45	8
High Males	10	20	2	52	1.5
Low Males	6	20	5.5	39	3

High feature usage by low self-efficacy females may at first seem to contradict our previous results, which showed that for females, high self-efficacy predicted more effective use of features (as measured by the overall testedness of the spreadsheet), which in turn led to greater debugging success (Beckwith, Burnett and Wiedenbeck, et al. 2005). We proposed that offering greater support in the environment would encourage low self-efficacy females to use the features more. The current study used the High-Support Environment, which included features designed to fix that very problem. Our results show that they worked—the low self-efficacy females did indeed use the features in this version! But our current study suggests that quantity of feature adoption is misleading in isolation: feature adoption must be considered in conjunction with how the features are used.

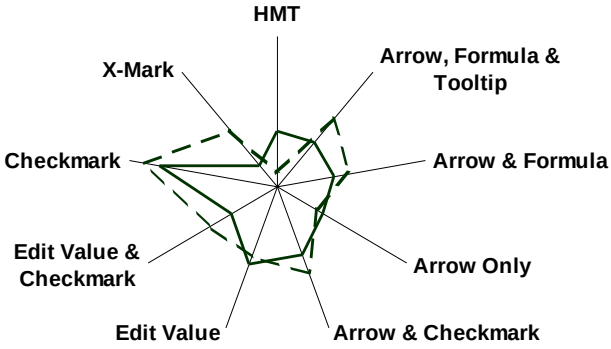
How were the checkmarks, so popular with the high self-efficacy females, used? Remarkably, the checkmark usage fell into the same number of patterns for the high and low self-efficacy females (and in fact for both groups of males as well). This

suggests that the checkmark only strategies used, which relate to systematic testing, were the same for both groups (Figure 6), but the amount they were used (Table 4) was different. There are many prior studies indicating that using this feature is directly tied to success (e.g., (Beckwith, Burnett and Wiedenbeck, et al. 2005), (Beckwith, Kissinger, et al. 2006)), and in this study, high self-efficacy females used it more and indeed succeeded more.

Other than the checkmark-related patterns, Figure 6 shows that high and low self-efficacy females had pattern frequency profiles that are very distinct from one another, suggesting that self-efficacy made a difference with females. However, the males' self-efficacy did not appear to matter much in their pattern choices.



(a) Female



(b) Male

Figure 6: How by self-efficacy group. High self-efficacy: solid line, low self-efficacy: dashed line.

High and low self-efficacy females diverged in both counts and patterns. Notice in Figure 6 how many different patterns the low self-efficacy females used compared to high self-efficacy females, except for the checkmark and arrow & checkmark. As suggested by self-efficacy theory, people with high self-efficacy are more likely to abandon faulty strategies faster than those with low self-efficacy (Bandura 1986). Our results were consistent with this. For patterns other than the checkmark patterns, the high self-efficacy females were willing to try out and quickly abandon many patterns in order to settle upon the ones they liked, whereas the low

self-efficacy females were more likely to try a pattern again and again before ultimately moving on. Figure 7 shows this tendency for unsuccessful females to use a bigger set of patterns more often, whereas successful females stick with a smaller set of frequent patterns. For example, 54 patterns were only used 5-10% of the time by high self-efficacy females, but only 16 were abandoned so quickly by the low self-efficacy females. This leads to the following hypothesis:

Hypothesis: Females with lower self-efficacy are likely to struggle longer to use a strategy that is not working well, before moving on to another strategy.

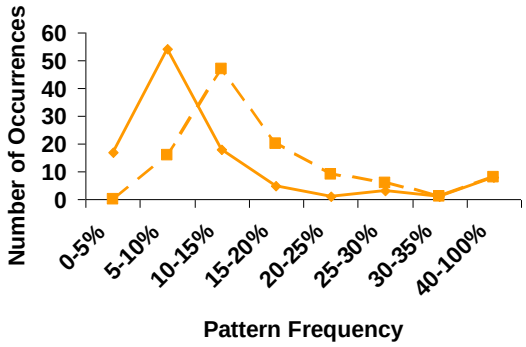


Figure 7: How: The high self-efficacy females (solid line) had more patterns fall in the frequency range of 5-10%, whereas the low self-efficacy females had more of their patterns fall in a higher number of debugging sessions (10-15%).

4. (BUSINESS UNDERSTANDING) STUDY 2: A NEW DATA MINING STUDY

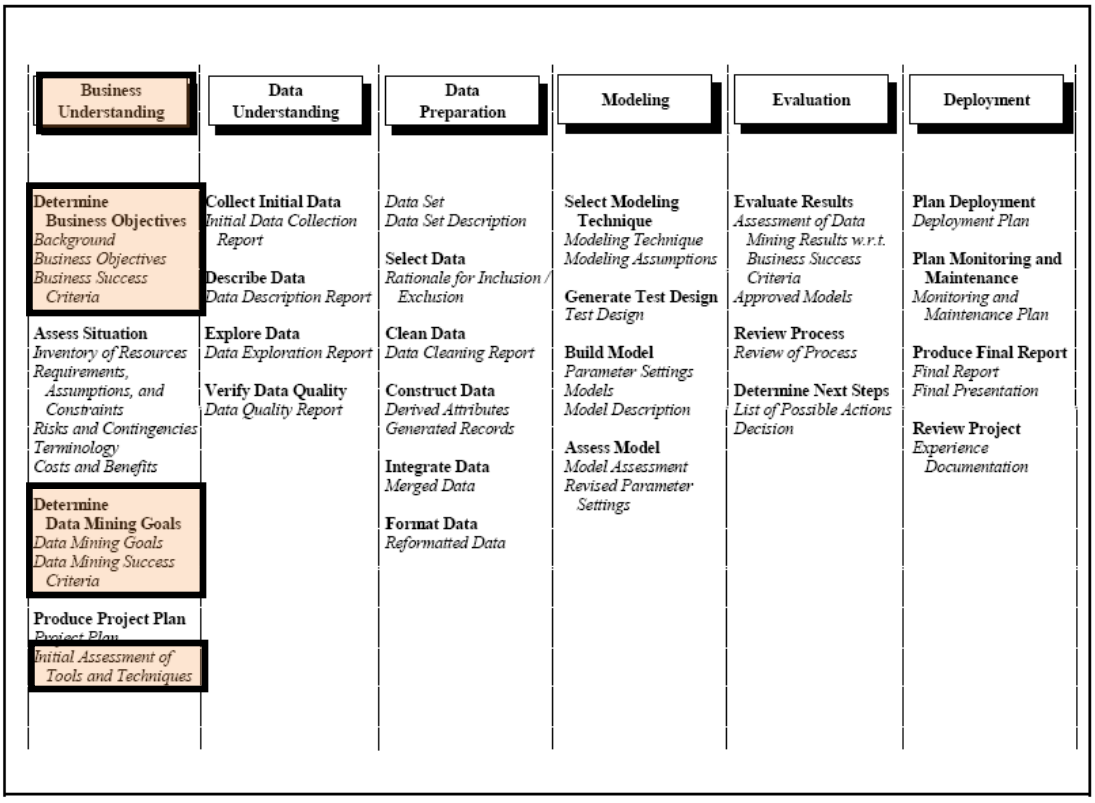


Figure 8: These are the six steps of the CRISP-DM data mining process model and the subparts and their outputs. Chapters 4 and 5 are about understanding the business understanding step.

In this second Gender HCI data mining study, we aimed to treat several aspects of the data mining process differently from Study 1. We did so in the hopes of further

advancing the subfield of Gender HCI, by getting one step closer to understanding successful male and female problem-solving behavior. The four aspects that we addressed differently in this study are: (1) arriving at statistically-significant results, (2) increasing the size and complexity of the dataset, (3) decreasing the amount of grouping, and (4) differentiating between "strategies" and "complex behavior."

4.1. Arriving at Statistically-Significant Results, Rather Than Hypotheses

Study 1 resulted in several hypotheses about male and female feature usage patterns. We did not run statistical tests, since we were reanalyzing old data and only using part of them. In Study 2, we followed the recommendation of Thomas Green, a psychology researcher, of going after statistically-significant results. After having discussed the possibility of running statistics on the results from Study 1 with statisticians, we were not able to come up with a good way of doing so. We were set on doing so in Study 2, however.

Many practitioners and researchers fail to create good data mining models because they do not place enough importance on statistical significance, and therefore overfit the model to the specific dataset that they have available (Elkan 2001). Existing methods guard against this by measuring how well the model created on part of the data (the training set) fits a second set (validation and/or testing sets). Splitting the data into multiple sets increases the probability that the models developed on the training dataset generalize to any new set of data. This is important both when data mining is used for data exploration and also when it is used for predictive purposes.

Another common data mining practice is to use several data mining algorithms and to compare the accuracy of the resulting models. We compared and evaluated the

competing models' performance through the use of lift charts and classification matrices.

4.2. Increasing the Size and Complexity of the Dataset

The more data we use, the richer the resulting models will be. In our original data mining study, we only used the Treatment Group participants of the Summer 2005 Gender HCI study. In order to get patterns at the right granularity (as well as because of certain software and algorithm limitations), we had to narrow our log file and questionnaire data down to the events in the order that they happened, the participant's gender, whether their pre self-efficacy was high or low, and whether they fixed more or fewer bugs than the median.

Using More Available Data per Study. For the Study 2, we used as much of the contextual data as possible: background questionnaire answers, information specific to the task, information specific to the spreadsheets, information about the cells that are touched, etc. With richer data, we can take more factors into account, thereby getting a deeper understanding of female and male factors that lead to success in debugging spreadsheets.

Using more information for each participant made it harder to organize the data in such a way that the algorithms output meaningful results; the data have to be organized into hierarchies so that the mining algorithms can understand how the pieces of data relate to each other. Data preprocessing steps were also more arduous, since we had a much larger and more varied dataset to clean and standardize.

The software tool that we selected to find patterns gave dependable models resulting from the analysis of several tables of varied and interconnected data. While this process was harder, it resulted in links between complex behavior and subject characteristics.

Gathering Data from Multiple Studies. In addition to using more of the data that we had available for each participant, we also decided to get more "data points." The data that we analyzed came from three earlier studies. Having more participants to examine increased the statistical power of the results. Furthermore, since these patterns span three slightly different environments, the resulting findings are that much stronger than if they would have only occurred in one environment. This combination of data also made the data preprocessing steps more involved. We had to find out about the similarities and differences between the three studies and how those affected both the data and the resulting patterns. While collecting, combining, and standardizing the data were time-consuming, the richness of the results was worth this extra effort.

4.3. Decreasing the Amount of Grouping

Input Data Grouping. Due to the data mining algorithms used and because of the low number of individuals in the target group, a certain amount of grouping of the data was required. During our sequential pattern analysis, we grouped participants into the dichotomous groups of "Successful" and "Unsuccessful" (depending on whether they fixed more bugs than the median, or not) and "Low Self-Efficacy" or "High Self-Efficacy" (depending on whether their pre-task self-efficacy score was higher than the median or not). Dividing into more groups or modeling the success variable as continuous data (between 0 and 100% of bugs fixed) did not result in dependable patterns since we had too little participant data to actually look for these types of patterns.

The statisticians that we consulted about how to get statistical significance recommended keeping the individual the center of attention, rather than immediately splitting participants up into dichotomous groups. They recommended not losing the

person effect, unless we first show that there are roughly no differences between participants who would get grouped together. For example, instead of using a box plot, they recommended using a scatter plot, which also shows the sample size.

In Study 2, increasing the number of participant data allowed us to keep more of a continuum of information for certain fields (like self-efficacy). Though some grouping was still necessary, we varied its amount to see which helped us best understand participants' behavior.

4.4. Differentiating Between "Strategies" and "Complex Behaviors"

Marian Petre, a psychology researcher, pointed out that patterns only give us information about the participants' behavior, not their strategies. We can only know what a participant's strategies are by acquiring statements about their intentions (such as through open-ended survey questions or a think-aloud study). However, patterns allow us to notice more obscure links between a participant's background, their behavior, and their success at the problem-solving task.

With this in mind, and knowing that only one of my datasets has some information about participants' strategy choices, we will differentiate between "strategies" and "complex behavior" in this paper. Examining how those mental strategy choices relate to the complex behavioral patterns that we find is beyond the scope of this paper. Future studies should be conducted to explore links between participants' self-proclaimed strategies (or explanations for why they take certain actions) and their observed behavioral patterns.

5. (BUSINESS UNDERSTANDING) CONTEXT, OBJECTIVES, AND SUCCESS CRITERIA

5.1. Data Mining Study Context

A *data mining context* is the first step in mapping a generic data mining process model to a specialized study. The context of a data mining project can be thought of as being made up of four parts: (1) the application domain, (2) the data mining problem type, (3) the technical aspects, and (4) tool and technique (Chapman, et al. 2000). Together, these parts also help create a good general overview of this study.

5.1.1. Application Domain

We applied data mining techniques to the domain of Gender HCI. Gender HCI deals with differences in how males and females interact with software. In particular, our data come from studies conducted with the research spreadsheet environment Forms/3 (Burnett, et al. 2001), which employs the What You See Is What You Test (WYSIWYT) testing methodology (Rothermel, et al. 2001).

5.1.2. Data Mining Problem Types

Various problem types exist in data mining. A data mining study can aim to solve more than one type of problem. The problem types of this study are both ones for finding interesting patterns and associations in the data and for seeing what factors combine to predict debugging success: data description and summarization, segmentation, concept description, and classification.

Data description and summarization can be a self-standing data mining project. However, it is often used in combination with other problems, as a part of the data understanding step of data mining. Data description and summarization consist of a thorough exploration of the data, combined with simple descriptive statistical and visualization techniques, to provide insights into hidden information in the data early on. The difference among calculating maximums and minimums, finding statistically significant connections between background factors and success, and creating models that predict success based on background and behavioral characteristics, is simply the level at which the data is described and summarized, ranging from less to more complex interactions.

Segmentation types of studies divide data into subgroups with interesting ties and similar characteristics. In some studies, the detection of groups can be the main goal of the study. Segmentation is also often a part of a more involved (higher end) data mining study. It is often used to make data set sizes more manageable. Another use of segmentation in data mining studies is to come up with more meaningful models, based on more homogeneous groups. As with all statistical analysis techniques, when datasets are big, various factors from different groups can overlap and counteract each other. It is often both more meaningful and easier to look for patterns in interesting segments of the population. In Study 2, we used segmentation to find interesting homogeneous groupings of our participants.

Concept description is highly tied to the two aforementioned problem types. While segmentation problems provide classes, concept description provides an understandable description of each class. Its goal is also to provide a deeper understanding of relationships within the data, rather than creating dependable prediction models. In this case, the concepts (or classes) that were especially interesting were successful males and successful females. From the description of

what factors relate to male success and what factors relate to female success, we can already make some inferences about what to focus on in the software. The result will therefore be a set of guidelines for spreadsheet testing tool design: some coming from only females, some only from males, and others from both.

Classification type studies are predictive studies. As with concept description, a set of classes of individuals is created. Classification maps every participant to a grouping of individuals through prediction. Unlike with concept description, those classes do not have to be understood. The group names can be arrived at through segmentation. They can also be arrived at by discretizing continuous values from predictive models into class labels. Techniques for solving Classification problems include discriminant analysis, rule induction methods, decision trees, neural networks, k-nearest neighbors, case-based reasoning, and genetic algorithms.

5.1.3. Technical Aspects

Technical aspects are details that will be encountered during the development of the data mining model and are sprinkled about the sections they pertain to. Such details include what to do with missing values, deciding whether or not to keep outliers in, and other such choices. Details about these technical aspects are included in the sections that they pertain to, throughout the thesis.

5.1.4. Tools and Techniques

In addition to the usual analysis tools (Excel, Access, and S-Plus), we used Microsoft's SQL Server 2005 and its Analysis Services (SSAS) data mining algorithms in the Business Intelligence Development Studio. The data that we collected from the three studies came in the form of Excel files and text files which were very close to CSV form. It was advantageous to migrate from Excel to a database

(Access and SQL Server) since database tools facilitate cleaning large amounts of data and, in the process, also allow for a better understanding of the data.

We used all six of the SSAS data mining algorithms (techniques) that applied to our type of data to build competing models: association rules, cluster analysis, neural networks, decision trees, naïve bayes, and logistic regression.

5.2. Objectives and Success Criteria

5.2.1. Business Objective

In order to avoid the situation of finding right answers to the wrong questions, a researcher first needs to come up with a clear objective for the data mining study.

> *My business objective was to find relationships between static (background and self-efficacy scores), behavioral, study-specific, and success Gender HCI data that, if taken into consideration, will ultimately make spreadsheet software more gender-neutral.*

Research questions related to my objective include:

- *What combinations of static, study specifics, and action characteristics lead to female success at fixing bugs?*
- *What combinations of static, study specifics, and action characteristics lead to male success at fixing bugs?*
- *Do unsuccessful females exhibit any common characteristics?*
- *Do unsuccessful males exhibit any common characteristics?*

5.2.2. Business Success Criteria

The business objective has been properly met if the study results in any findings that help us better understand the differences in how males and females successfully problem-solve and what differences exist between the two genders. Finding useful relationships between various behavioral, background, study, and success variables would mean that the goal has been achieved. These results will not have to be new, since the data we analyzed has been previously analyzed in multiple ways for Gender HCI with the same goal in mind. Therefore, it would be just as useful to triangulate by finding patterns that simply verify previous Gender HCI findings through this different analysis method.

In order to be useful, these findings need to be actionable. This means that, as Gender HCI researchers, we can generate hypothetical solutions to help make problem-solving software more gender-neutral based on this study's results. The usefulness of these patterns will be verified by Gender HCI researchers.

5.2.3. Data Mining Goal

A data mining goal differs from its business equivalent only in the terms used to describe it.

The data mining goal in this study is to find statistical patterns specific to homogeneous groups that the data will be divided into through clustering and other methods. In other words, the goal is to find out which combinations of static, behavioral, and task characteristics relate to debugging success for females and males.

The outputs for this study are a set of understandable homogeneous groupings based on several characteristics. Association rules will also be derived to find out what factors are tied to male and female success.

5.2.4. Data Mining Success Criteria

In order to choose between models, we created lift charts to determine how well each model fits a new set of data (see Chapter 9). While there are no clear cut-offs for what makes for a "good model" vs. a "bad model", the closer to 100% fit, the better. Since we had two possible outcomes (successful or unsuccessful), a 50% fit would be achieved by random chance. A 60% fit already means that the model is onto something. For human data, a 70% fit is considered a very good performance. We considered models who predicted more than 70% of the participants' success correctly to be satisfactory.

5.3. Rejected Objectives

While this thesis did not address them, the domain of Gender HCI would also benefit from answers to the two data mining problem types below.

Dependency analysis is often followed by predictive types of problems and acts as a set up for them. In Gender HCI, the dependency analysis problem would be to create a model that ties behavioral events and background information to the success of both genders in problem-solving tasks. Unlike for Concept Description, models imply having a comprehensible class for every data point (for example, "successful females", "successful males", "unsuccessful females", and "unsuccessful males").

Prediction type algorithms only differ from classification algorithms in that the target class is continuous. For example, instead of seeing whether a user will be successful or unsuccessful, prediction algorithms would classify a user as fixing 70%

of bugs. Time series algorithms are for forecasting prediction problems. The other types are usually called regression algorithms and employ algorithms such as linear regression and logistic regression. They are often used for tasks such as predicting the expected revenue of a company.

The Gender HCI domain can be translated into any of these three types of problems as well. While these objectives had to be rejected for this study due to time constraints, using techniques particular to these problems in the future could further help us understand just what males and females are doing when they problem-solve and what kind of support would help their efficiency and effectiveness at debugging spreadsheets.

6. (DATA UNDERSTANDING) WHERE WE GOT OUR DATA

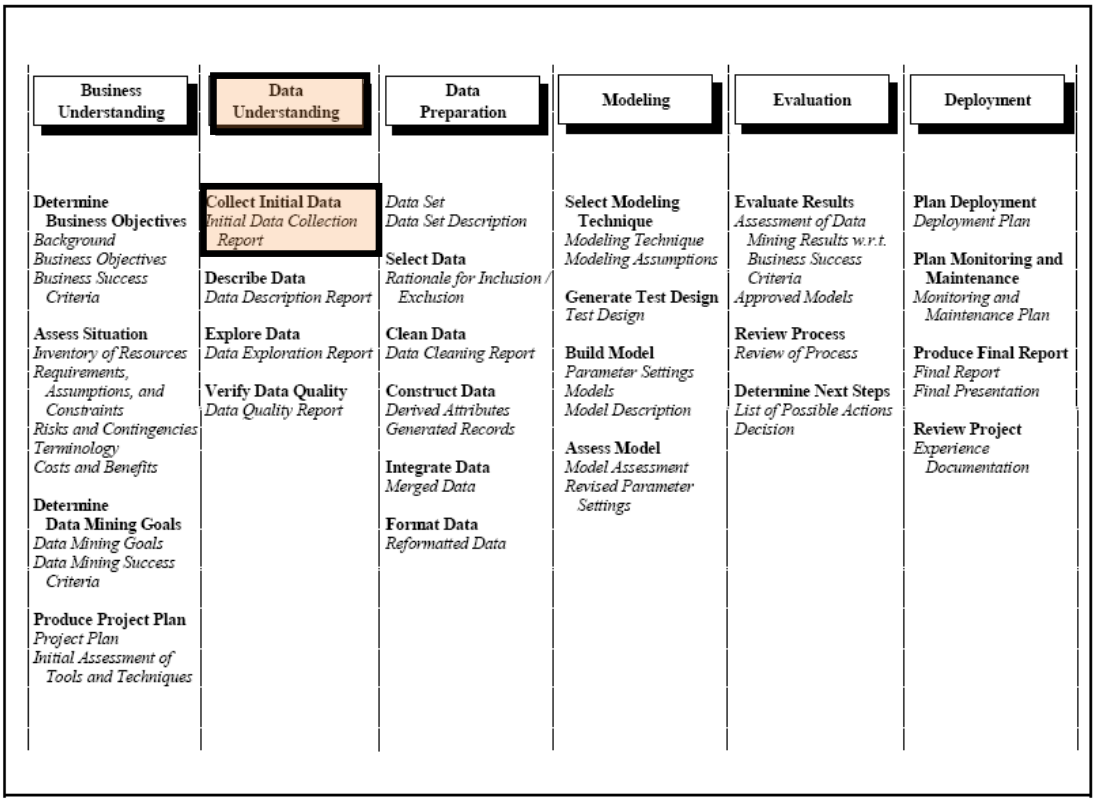


Figure 9: These are the six steps of the CRISP-DM data mining process model and the subparts and their outputs. Chapter 6 is about understanding the data.

6.1. Criteria for Selecting Studies

Since it is always advantageous to train models on as big a dataset as possible, we selected several studies that had previously been conducted within the Forms/3 research group and combined their data. My criteria in selecting those studies were:

1. They had at least 25 participants (so that the effort needed to combine datasets did not exceed the reward from having a few additional participants) and
2. Their setups and recorded data were similar enough to be combined to give Gender HCI related patterns.

Four quantitative empirical studies had been conducted that fit the first criterion (Spring2002Assertions, Winter2004FaultLoc, Summer2005GenderTinkering, and Summer2006GenderStrategies). Spring2002Assertions, unfortunately, did not meet the second criterion: it lacked any data on the gender of the participants. Fall2006Explanations and Fall2003HighIntensityInterruptions were studies that met the second criterion, but not the first. In addition, there were several studies that did not meet the second criterion and therefore were eliminated right away.

We built the models using the following types of data that the three studies had in common: questionnaire data on the participants (including gender, confidence, and academic data), data about the study (what cells were formula cells, which were value cells, what events the participants could perform, what types of events those were, etc.), data about their behavior during the study (log files), and data about their success (how many bugs they found, how many bugs they fixed, and how well they understood the features).

6.2. Chosen Studies Background

The three chosen studies were:

1. Winter 2004 Fault Localization (Ruthruff, Phalgune, et al. 2004),
2. Summer 2005 Gender Tinkering (Beckwith, Kissinger, et al. 2006), and
3. Summer 2006 Gender Strategies (Beckwith, Grigoreanu, et al. 2007)

Since these studies were experimentally set up to answer different research questions, this section provides an overview of the goals of the three studies, their setup, and their results.

6.2.1. Winter 2004 Fault Localization

The dataset from the Winter 2004 Fault Localization dataset had 54 participants. 24 of those participants were from the Control Group, which contained 9 females and 15 males, and 30 were from the Treatment Group, which contained 14 females and 16 males.

Goals. The Winter 2004 Fault Localization study (Ruthruff, Phalgune, et al. 2004) was conducted to get a better understanding of the impact of rewards in the Surprise-Explain-Reward methodology. The participants were divided into two groups, both of which provided them with the same amount of feature functionality. One group, however, got more feedback that could be perceived as rewards for using fault localization techniques than the other group.

Experimental Setup. The researchers first looked for rewards and punishments relating to the fault localization device in the spreadsheet environment used. The fault localization device is a part of the "What You See Is What You Test" (WYSIWYT) testing methodology (Rothermel, Burnett, et al., A Methodology for Testing Spreadsheets 2001). They then implemented two versions of the environment with varying amounts of perceivable rewards and punishments.

The features used in this study were also used in the following studies. It is important to understand what some of these features do, since we counted the number of times that they showed up in log files and built mining models using those counts. When a checkmark is placed, cell borders change colors to show how tested a cell is. A cell with red borders means that it has not been tested yet. A cell with blue borders

means that all of its situations have been tested. A purple border means that the cell is somewhere between 0% and 100% tested. At the cost of placing a checkmark on a cell, a user is thereby provided the visual feedback about the progress made in testing the spreadsheet, which can be considered a perceived reward.

Forms/3 gives this testedness progress feedback in three ways. The first is at the cell level, as already mentioned, by coloring the cell border. This feedback is also provided at the sub-expression level through arrows. When arrows are brought up to see the relationship between cells in the spreadsheet, the arrows are colored with the same red-purple-blue color scheme to depict how tested the relationship between those two cells is. Furthermore, posting a formula breaks the arrow into the number of sub-expressions that the formula has, to show how tested each sub-expression relationship is. A third testedness progress feedback mechanism is an overall testedness bar that shows the cumulative percentage of the spreadsheet that has been tested.

Other than allowing the user to keep track of how much they have tested each cell, more tangible rewards can also follow: in wanting to make testing progress, a user might notice that a cell does not return an expected value and might therefore be buggy.

When a user notices an incorrect value, they can place an x-mark in the cell's decision box. Similarly to checkmarks, placing x-marks also provide visual feedback that can be perceived as a reward. When an x-mark is placed, the interior of all of the cells that contribute to that cell's value get colored in different shades of orange (ranging from light to dark). The darker the interior coloring of the cell, the more likely it is that it contains a bug. The reward here is that the user gets a clearer picture of where errors are likely to hide in the spreadsheet.

There were three differences between the low-reward environment used by one group and the high-reward environment used by the other. (1) The first was to remove the testedness progress of a cell when it was found to be potentially buggy. This is because end-user programmers often get confused about the difference between testing a cell based on a set of input values and deciding that a cell's formula is "correct." Thus, a cell that provides conflicting feedback might confuse some users. For the low-reward environment, the testedness feedback was removed, but both the testedness and fault likelihood feedback were left in the high-reward environment. (2) The first change affected a second change: the explanations that are given for cell borders and arrow testedness. Since how tested the cell was did not actually change (only the visual feedback did), the explanations that pop up when users hover over the borders and arrows would still give conflicting feedback, when compared to the colors. Thus, those explanations were removed for the low-reward group. (3) The third change was that a fault localization bar was added to the high-reward group's environment, similar to the overall spreadsheet testedness bar. This was so that the rewards from using checkmarks are not out of balance with the rewards from using x-marks.

Results. The study's results were that the group that had a higher reward structure fixed significantly more bugs in the harder of the two given tasks (Payroll) and also had a better comprehension of how the fault localization features worked. The conclusion of the paper was therefore that it is not sufficient to make features that work well and to explain how they should be used, but that it is also important to increase the perceivable rewards of using those features, in order for participants to be more successful at using the features.

6.2.2. Summer 2005 Gender Tinkering

The Summer 2005 Gender Tinkering study provided 76 participants. 37 of those participants were from the Control Group, which contained 17 females and 20 males, and 39 were from the Treatment Group, which contained 23 females and 16 males.

Goals. The Summer 2005 Gender Tinkering study (Beckwith, Kissinger, et al. 2006) was conducted based on design changes proposed in an earlier Gender HCI study. Two environments were compared to look for the effect of the changes on tinkering behavior and on self-efficacy.

Experimental Setup. As in the Winter 2004 Fault Localization study, the debugging features present were part of WYSIWYT ("What You See Is What You Test").

For this study, a high-support environment (for the treatment group) was designed based on previous findings and was compared to the earlier version of the environment, which was low-cost in terms of tinkering (control group). The treatment high-support environment had several additions. It included 4-tuple confidence marks, expandable tooltips, and Help Me Test.

The control group users had two choices of marks to place in a cell's decision box: a checkmark or an x-mark. To place a checkmark, a user had to left-click and, to place an x-mark, right-click. In the high-support environment, a user had four choices of marks: a high-confidence checkmark, a low-confidence checkmark, a high-confidence x-mark, or a low-confidence x-mark. The only difference between a high- and a low-confidence mark is the transparency of the colored feedback (high-confidence feedback has a much darker shade than low-confidence). In this study, we

grouped low- and high-confidence marks together, since high-confidence marks were not available for one third of the data.

The expandable tooltips are variations on the explanations that come up when a user hovers over any feature in the environment. In addition to the regular tooltips, users in the high-support environment also had the option of expanding the tooltip to read additional information about that feature. Also present in the environment was the "Help Me Test" (HMT) feature. Sometimes it can be difficult to find test values that will cover the untested logic in a collection of related formulas, and HMT tries to find inputs that will lead to coverage of untested logic in the spreadsheet, upon which users can then make testing decisions. While the users had these features available, we did not use either tooltip or HMT data in this study. Tooltip data was ignored because we do not trust those data: tooltips pop up all the time and it is impossible to differentiate between the tooltips that participants wanted to bring up and those that came up just because of where they left their mouse. We also ignored HMT data because only half of the overall dataset had that feature available to them.

Results. Males in the low-cost environment tinkered significantly more than everyone else. That environment made it easier to tinker by requiring only one click to place a mark in the decision box. Tinkering behavior, in general, was positive, except for the low-cost males. The low-cost males did more mindless tinkering, since they did not take the time to pause and think about the feedback.

6.2.3. Summer 2006 Gender Strategies

The Summer 2006 Gender Strategies study (Beckwith, Grigoreanu, et al. 2007) is a study in progress, though all of the data has been collected for it. It provided 61 participants: 37 females and 24 males.

Goals. The goals of this study were to quantitatively and qualitatively go after the differences in strategies that males and females employ while problem solving (debugging spreadsheets, in particular).

Experimental Setup. The main difference in the set up between this study and the Treatment group from the Summer 2005 Gender Tinkering study is that, in this study, the cells were laid out in a grid-like pattern to give them an Excel-like appearance, since participants were familiar with Excel. Another difference was that cells were rearranged to remove confounds in cell orders. We looked for several orders in which users traversed the spreadsheet: dataflow, western reading order, column order, description order, and example order. Dataflow order could not overlap with western reading order, for example, because we then would not know exactly which of the two they were following.

As in the Summer 2005 study treatment group, users had the choice of placing either low-confidence or high-confidence marks. They also had the opportunity of using Help Me Test, if they wanted. Arrows, border colors, and cell interior coloring were as in previous studies. Tooltips were available to explain all of the features. Unlike in Summer 2005 treatment group, they were no longer expandable, but only provided the shortened explanations provided to the Summer 2005 control group.

Another change major change made in the setup of this study was to the study's tutorial. During each session with the participants, we give them a tutorial that typically lasts about 25 minutes. This tutorial gives participants a chance to learn about the features and to explore them before having to use them for an actual task. Usually, we told participants both what the tools are and also how to use them. In this study, however, we did not want to bias participants' behavior, since we were looking

for the strategies that males and females employ to problem solve. This, this time around, the tutorial was simply a "tour of features."

Results. There were significant gender differences in the strategies that successful males and successful females used to debug spreadsheets. While spreadsheet debugging tools best support users that approach the problem from a depth-first dataflow perspective, females rarely used depth-first dataflow strategies. Both genders used testing as a strategy, but mainly females used code inspection, either by itself or in conjunction with testing. This study also revealed strategies used by the participants that are virtually unsupported in spreadsheet environments.

7. (DATA UNDERSTANDING AND PREPARATION) DATA COLLECTION, INTEGRATION AND STRUCTURING

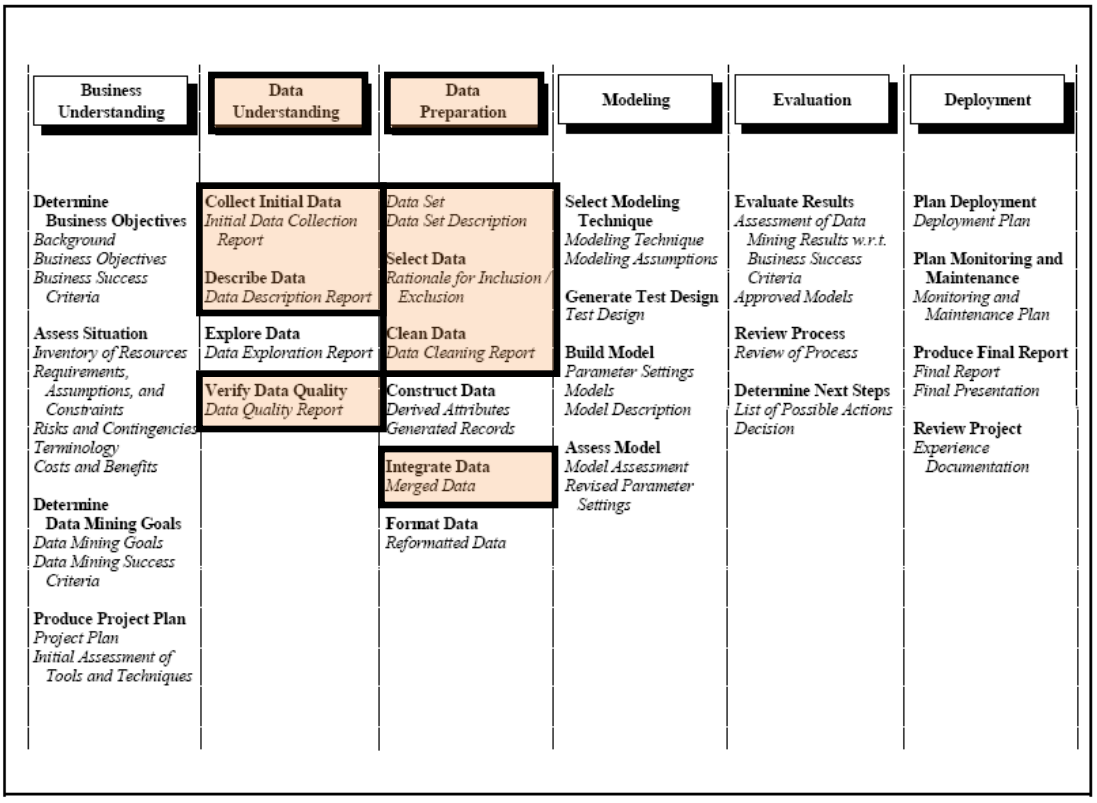


Figure 10: These are the six steps of the CRISP-DM data mining process model and the subparts and their outputs. Chapter 7 is about understanding and preparing the data.

The data used in this study came from the three studies mentioned in Chapter 6. This chapter is about the low-level similarities and differences between those three sets. This knowledge helped with accurately integrating them, making decisions about how to best apply the mining algorithms, and interpreting the resulting models.

Each study provided four types of data: (1) log files of the actions that participants took while working with the software, (2) a set of questionnaire data plus various calculations resulting from them, (3) spreadsheet characteristics, and (4) study-specific decisions about what events got logged. The first dataset is addressed in Section 1, the second in Section 2, and the third and fourth are addressed in Section 3 of this chapter. For the complete tables that resulted from the collection-integration-structuring process, as well as descriptions and examples for each of the variables, see Appendix A.

Some of the data available from the previous studies were calculations derived from the raw data (such as totals of bugs fixed and results from scripts counting the amount of tinkering or how much a certain cell order was followed). We collected and integrated the raw data only, and redid some of the calculations that we needed (like bugs fixed) later in the process.

The data preparation consisted of three steps: (1) collecting them, (2) combining/integrating data from the three studies, and (3) standardizing them. The data collection step resulted in 14 tables: see Figure 11. The combination step consisted of reducing the 14 tables to four (one for each type of data). The standardization step involved making sure that the values in any field for one study were comparable to the values in the same field for a different study.

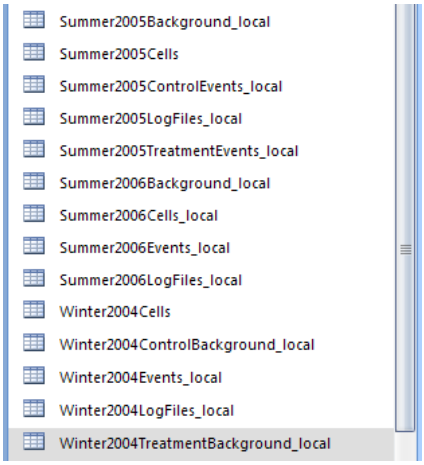


Figure 11: All of the data from the different studies, in a format that Access likes on the table scale. These were later combined into only four tables.

7.1. Log Files Data

7.1.1. Collection

Each subject included in this study had two log files in the original studies' directories: one for the Gradebook task and one for the Payroll task. The Winter2004 and the Summer2005 studies had a Control and a Treatment group. There were times when log files were available for subjects who were not listed in the questionnaire and background data tables. Those subjects' log files therefore had to be dropped from the dataset.

7.1.2. Combination

The log file combination consisted of seeing what differences there were in the structure of the log files between the three studies. We then created an overall structure that would fit all three studies. Examples of differences among the files included: the addition of low and high confidence checkmarks and x-marks in the Summer 2005 and Summer 2006 studies, the addition of keyboard shortcuts in the

Summer 2005 study, and adding an "undo" field to the "Checkbox Clicked" event that was changed to "T" for the Summer2006 study.

7.1.3. Standardization

Our log files are semi-structured since different fields mean different things, depending on the user or system event. Data mining algorithms need structured data to come up with sensible models (i.e. each column heading is the same for all of the rows in the log files). At least two solutions exist when going from a semi-structured to a structured dataset: (1) to throw away most of the data, keeping only structured bits for analysis, or (2) spend time structuring the data to take advantage of having more data available to mine.

In Study 1, we stuck with the first choice, keeping only the event name for analysis (as well as a few extra details from one of the fields, which we concatenated to the event name). For example, we have an "Edit" event. When a user edits a cell, they can either be editing a formula (which got changed to "Edit Formula") or simply editing a value (which got changed to "Edit Value"). Similarly, "Checkbox Clicked" was changed to either "Checkmark" or "X-Mark". The rest of the log file information was ignored.

In this study, we chose to take the second approach, by structuring the log files in such a way that the fields were the same across all events and across the three studies. For the purposes of data mining, we also had to add some additional information that was not previously in the log files which included the log file that the data came from, the subject identification number, the task that the user was on, a "Seconds" field, and the line number in the log file.

Some other fields that we added to the records were the status of system events. Our log files record two types of events: System Events and User Events. One

example of a system event is displaying how tested the total spreadsheet is. The system events are most interesting in terms of their status when a user event is performed. We therefore moved the system events from their own records into fields for the user events. We added a field for each user event that says how tested the spreadsheet was when they performed any one event (for example, placing a checkmark). Similarly, we added fields for how the cell's testedness, the cell's fault-likelihood, whether the formula is open for that cell, and what time the formula was opened at.

This process resulted in a log file table that contained the same information as the original log files, but was now structured. For the analysis of this data, we linked it to static data about the users (questionnaire answers and performance data per subject) and task-specific data (event data and spreadsheet data), both of which we cover in the next two sections.

7.2. Questionnaire and Performance Data

7.2.1. Collection

The per-subject questionnaire and performance data came from multiple files. The questionnaire data included: actual data from the questionnaires (e.g. individual SE scores), calculations based on those answers (e.g. total pre SE), data coming from other files (e.g. bugs fixed), calculations made based on data from other files (e.g. total bugs fixed), and scores resulting from scripts being run (e.g. western reading order score). In our study directories, these data are in three types of Excel files: Questionnaire files, BugsFixedFound files, and AllData files.

7.2.2. Combination

For each of the types of data, we did two types of combinations: one within each study and one between studies. This questionnaire and performance data were the

most complex to combine because, even within one study, they came from different files.

In combining the data from the three separate studies, the first decision was how to deal with data that was not available for all studies. For example, one study had a questionnaire question specific to that study that the others did not have or some studies recorded all of the individual self-efficacy scores, while others only recorded the total. In data mining, the more data is available, the better. Combining the columns this way made us realize that we could get some of the missing data from other files saved in the original studies' directories.

Unfortunately, the studies had very little data recorded in common. Of my gigantic table of 291 columns, only a mere five fields contained data for all three studies! The big combined set included everything from answers to the questionnaires, to script outputs about tinkering, number of bugs fixed, counts of dumb vs. smart mistakes, comprehension scoring, and feature usage statistics. The ones that all three studies had in common were: Participant ID, Study, Gender, GPA, and Total Pre Self-Efficacy.

7.2.3. Standardization

There was little in common among the three studies' "All Data" spreadsheet files (files that are usually created from the raw data during the analysis of the data, which include a diverse set of relevant data). Some of the information that was missing from some studies' All Data file was lying around in other files in the directory. We looked up both data that we could not do without and data that were low-hanging fruit. This brought the total number of common fields up to 46. In order to make sure that the values within a column were comparable for all three studies, we standardized success measure scores, graded the spreadsheet experience and

programming experience of the users, and made sure that the format of the entries was the same throughout for all columns.

7.3. Forms/3 Details: Cell Data and Events

7.3.1. Collection

The cell data and the events data are particular to each study's setup. These data are important for eliminating confounds so that they might give an even deeper understanding of what participants are doing. The data collected included information about which cells contained bugs in the beginning of the task for cell data, where cells were located, which were formulas and which were values, and all of the possible events that participants could have used.

7.3.2. Combination

We created a table with all of the events that occurred in any one of the three studies. The goal of this events table was to list each possible event as a system, user, or HMT event. The Summer 2005 "All Data" spreadsheet file contained all of the events that were allowed in the other studies and more. The combined table was therefore the same as Summer 2005's. We used these data to write scripts that reorganized the log files data into a more structured dataset. We first moved system events like spreadsheet testedness, cell testedness, and cell fault-likelihood from log file rows into columns. Though system event analysis would be interesting to look after in future studies, we did not pursue this beyond the data preparation phase in this study. In this study, we analyzed only the lines of the log files that were user events (taking out system events and events like "Edit Formula" that were generated by Help Me Test).

The cell data table contained information about the layout and formula content of the cells, as well as data about what order those cells were in on the handouts that

we gave out (spreadsheet description and sample values). Both the layout of the spreadsheet and the order in which cells were listed on the various handouts differed among the studies. These data are also now preprocessed and available for future studies. In this study's analyses, the main cell data information that we used included whether a cell was a value or a formula, what the cell's name was, and what its ID number was.

7.3.3. Standardization

No standardization was needed for either the event or the cell data.

8. (DATA UNDERSTANDING AND PREPARATION, AND MODELING) BUILDING THE COMPETING MODELS

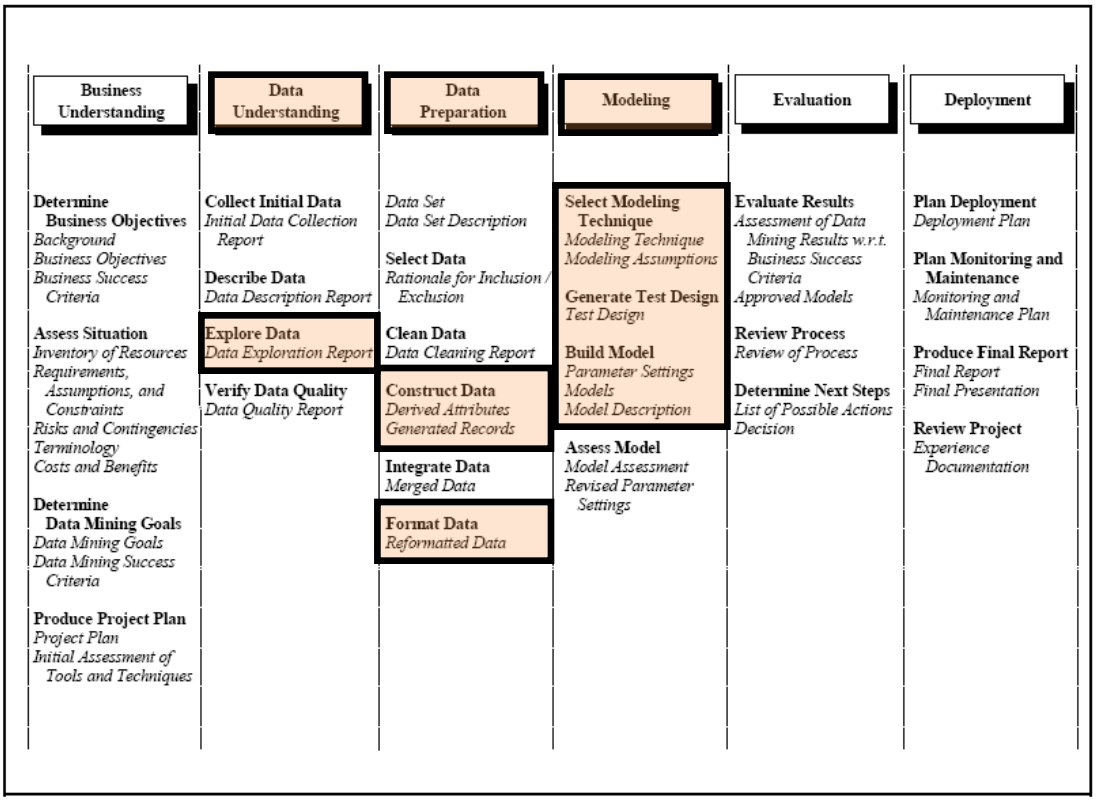


Figure 12: These are the six steps of the CRISP-DM data mining process model and the subparts and their outputs. This chapter is about exploring and preparing the data and then using them to create competing data mining models.

To reiterate our goal in this Gender HCI study:

> *Our business objective was to find trustworthy relationships between static (gender, GPA, self-efficacy, etc.), behavioral (event counts), study-specific (cell characteristics, number of bugs per task, user events permitted, etc.), and success (percentage of bugs fixed) Gender HCI data that, if taken into consideration, will ultimately make spreadsheet software more gender-neutral.*

Research questions related to our objective were:

- *What combinations of background, study specifics, and action characteristics lead to female success at fixing bugs?*
- *What combinations of background, study specifics, and action characteristics lead to male success at fixing bugs?*
- *Do unsuccessful females exhibit any common characteristics?*
- *Do unsuccessful males exhibit any common characteristics?*

In order for the algorithms to be able to answer our research questions, they needed to (1) be able to be used as predictive algorithms and (2) give some kind of insight about their rationale in predicting success. The goal of our final model is to see what combinations of questionnaire data and event counts could predict the number of bugs fixed.

8.1. OLAP Cubes

We used OLAP cubes to organize the data. Like data mining, OLAP cubes are a business intelligence approach for data reporting and forecasting. Thus, in the

process of organizing the data this way, we also explored them. The descriptive statistics resulting from that phase can be found in Appendix B.

An *Online Analytical Processing (OLAP) cube* is similar to a two-dimensional spreadsheet that has been extended to three or more dimensions. OLAP cubes have dimensions and measures. Each *dimension* of an OLAP cube is a category by which all of the data can be viewed (similar to a heading that might categorize a set of column or row headings). Each dimension is also broken up into its members. For example, "Gender" is one dimension of our cube and "Male" and "Female" are that dimension's members. Other dimensions in the Gender HCI data included Study, Subject, Workbook (the task, either Gradebook or Payroll), Line (line number in the log files), Time (time that the event was performed at), CellName, CellID, Event, and EventType. A dimension is the descriptive attribute of a measure. Thus, a *measure* (usually numeric) is a set of values, based on an attribute. They are the values that are being aggregated and analyzed. An example of a measure is "Event Counts". Events can be counted by line, by gender, by workbook, etc. See Figure 13 for an example of a three-dimensional OLAP cube.

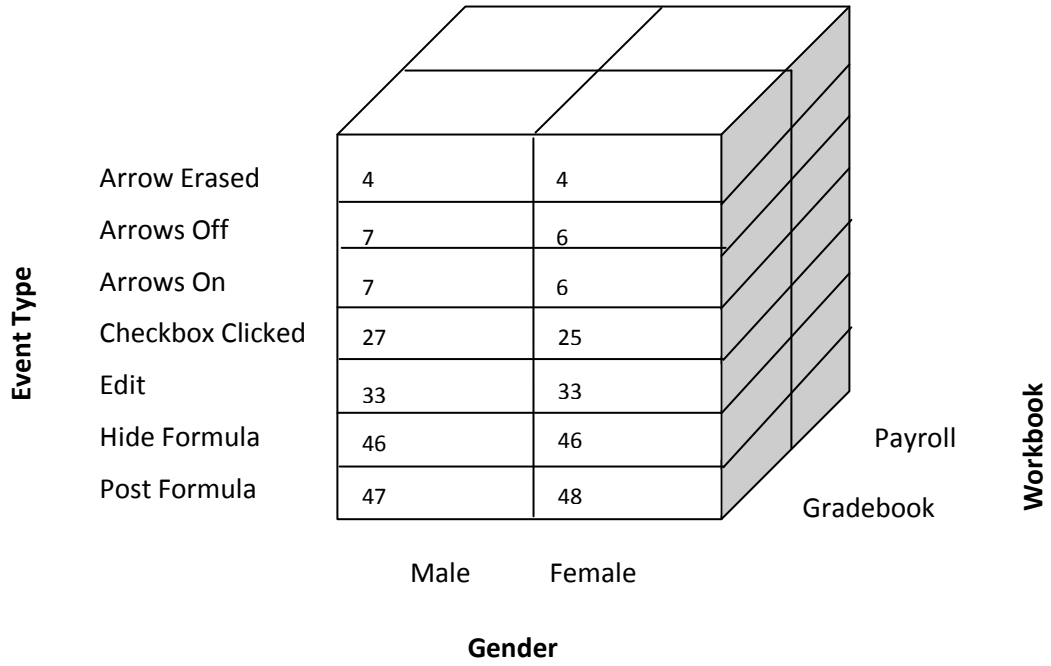


Figure 13: This is a sample three-dimensional OLAP cube for Gender HCI data. The three dimensions are Gender (members: Male and Female), Event Type (members are events that the participants can perform: Arrow Erased, Arrows Off, etc.), and Workbook (members are the two spreadsheet tasks: Gradebook and Payroll). The measures are event counts, which populate the cells. For example, in this OLAP cube, males had an average of four Arrow Erased events in the Gradebook workbook.

8.2. Modeling Technique

Since different modeling algorithms are better at predicting some measures than others, we ran all of the relevant algorithms on the input data described in the previous chapter (all non-time series algorithms that worked on discretized data). Those six algorithms were: Association Rules, Clustering, Neural Networks, Decision Trees, Naïve Bayes Networks, and Logistic Regression. See Appendix D for the parameter settings we used for each model.

The way of viewing slices of the OLAP cube is by creating pivot table views - these are a variety of summary tables. My source data for the mining models was a

combination of the resulting pivot table view of the OLAP cube that showed event count per cell type (value or formula) per task (Gradebook or Payroll) per participant, data from the questionnaires, and success data (see Figure 14).

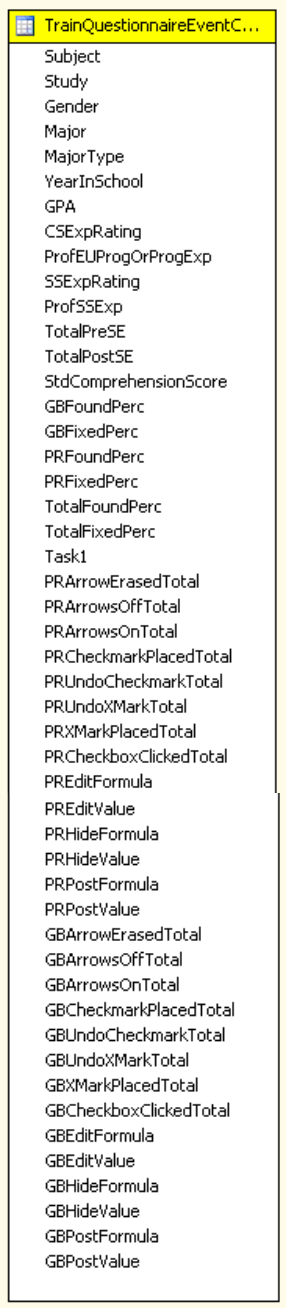


Figure 14: Fields from which competing models for predicting success at fixing bugs were built. The variable names should be pretty self-descriptive. They include static data, counts of events in Gradebook (GB) and Payroll (PR) tasks, and success data.

In order for all of the algorithms to be applicable, we discretized the continuous variables (such as GPA) into the maximum number of groupings possible. For the percentage of bugs fixed, after multiple unsuccessful attempts at creating models that accurately predicted participant success at a low level of granularity (10, 5, and 3 equally-sized buckets), we had to settle for the categorizations of "successful" and "unsuccessful", split at the median (two buckets of equal sizes). With more buckets, too few cases fell into the different success groupings, which did not give the model enough examples to dependably train the models.

8.3. Test Design

In order to test the validity of my models, we separated all of the data we had into two groups: a training set (randomly selected 70% of entries) and a testing set (randomly selected 30% of entries). We then verified the accuracy of the models using lift charts and classification matrices to see how well each model predicted bugs fixed.

When deciding on the percentage of entries to allocate to each set, we had to consider the tradeoff between building a better defined model, based on a bigger training set, and getting a better evaluation of its performance with a bigger testing set. Because 191 participants is still a relatively small group for data mining purposes, we had to take the safe route of building the models on a bigger set of data and hoping that they would be able to accurately predict the smaller sets of predictable values. Training and testing set size range anywhere from 50/50 (not too common) to 90/10 (fairly common). Since the dataset is small, we could not have a testing set with just 20 total participants, yet we also did not want to build the models on only 100 of our participants. This 70/30 split was therefore a reasonable one, falling halfway between the two extremes.

In terms of the predictive power of our models, we were aiming for the highest fit possible. Since participants could either be successful or unsuccessful, random guessing would lead to a 50% fit. With a 60% fit, the model is likely onto something. Especially with human data, even a 60% predictive power is good. It is common practice, however, to consider 70% as a good fit. We aimed for this percentage to reduce the possibility that the results were caused by random chance.

8.4. Building the Models

For each of the six applicable algorithms mentioned earlier, we set the key to be the "Subject" (or participant ID). We set Percentage of Bugs Fixed to be "PredictOnly." We ignored the percentage of bugs fixed in Payroll, bugs found in Payroll, bugs fixed in Gradebook, and bugs found in Gradebook, since those could not be used as input when predicting the overall bugs fixed and bugs found (they would have made the prediction trivial).

In addition to those six models, we also created an additional three models using the clustering algorithm, based on some hypotheses that resulted from earlier data exploration (see Figure 15 for all nine of these models). It seemed that female success in the number of bugs fixed was highly dependent on static factors alone (data from the background and self-efficacy questionnaires). We also hypothesized that male success could be derived fairly reasonably from dynamic behavioral data alone. These hypotheses resulted from an early attempt at mining only the static data (see Appendix C). We therefore created one of each of those models: one with only static data (CABackground) and one with mostly behavioral data (CA2). In addition to those, we also created a model using only those variables that linear regression tests found to be related to bugs fixed in a statistically significant manner (see Appendix B); significant static ones for females and significant dynamic ones for males

(CAEvents). The reason why we decided to use the clustering algorithm for these additional three models is that this algorithm type gives a lot of information about how it came up with its patterns (unlike algorithms like neural networks and Bayesian networks) and it also often does a good job of predicting output variables.

The “Key”, “Input”, and “PredictOnly” variables used to build the different models can be seen in Figure 15. These nine models are evaluated in Chapter 9 to see which best predicts participant success at fixing bugs.

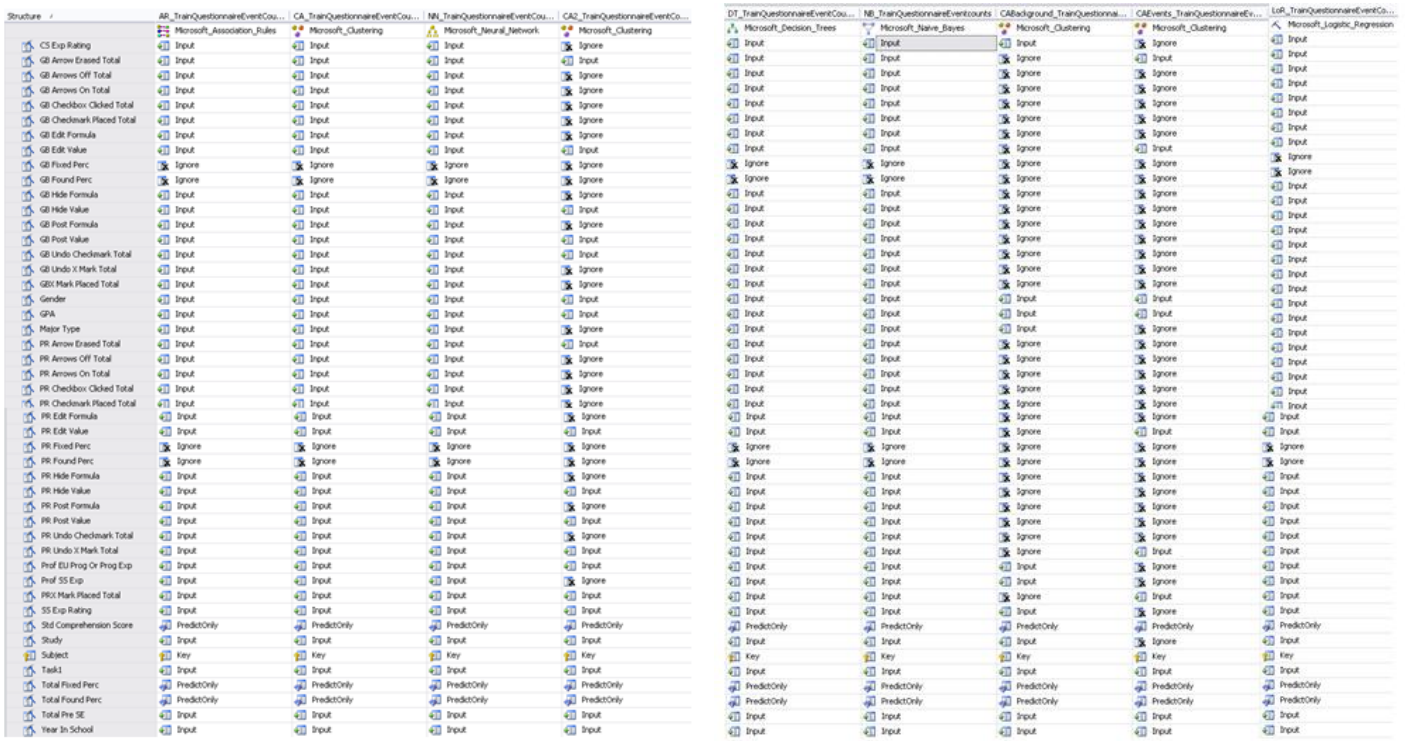


Figure 15: Each column is a model contending to be our “best” model for predicting success at fixing bugs. Each row is a variable and the role that it played in each competing model.

9. (MODELING AND EVALUATION) RESULTS AND VALIDATION: THE BEST MODEL FOR PREDICTING BUGS FIXED

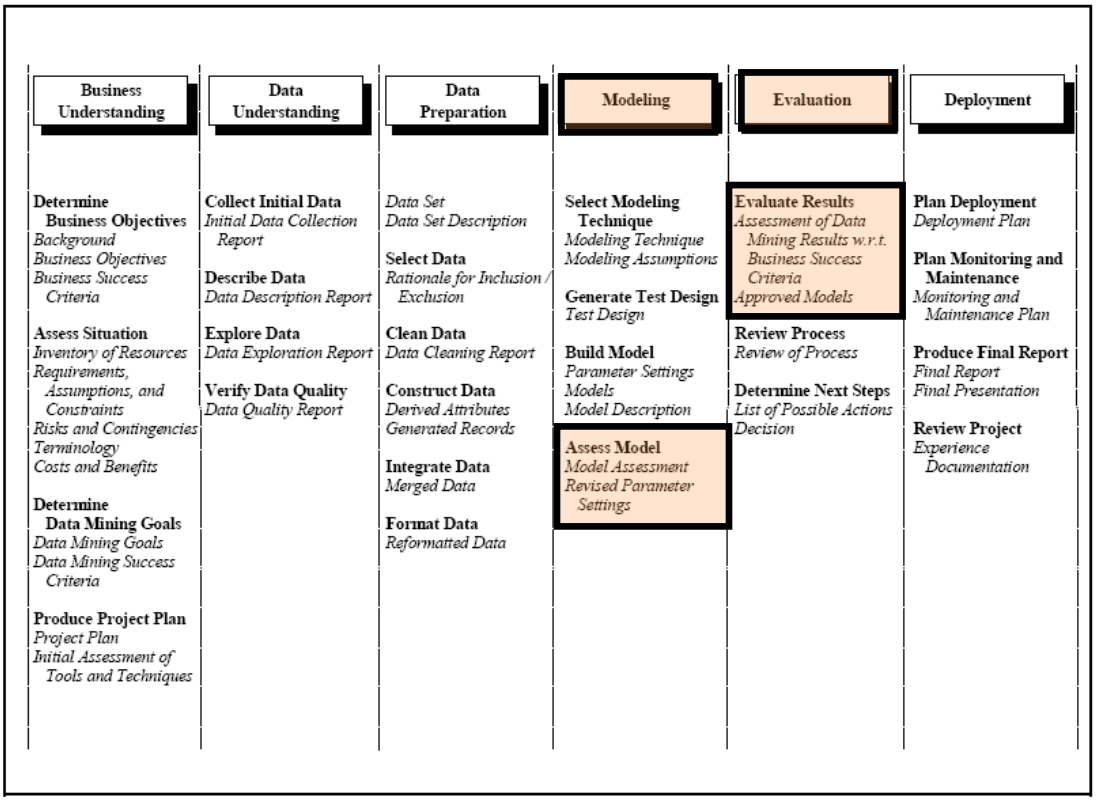


Figure 16: These are the six steps of the CRISP-DM data mining process model and the subparts and their outputs. This chapter evaluates the performance of the competing models and further creating an even better model.

One model often outperforms the other models in predicting a success variable. Occasionally, different models perform better under differing circumstances (such as

power in predicting female success vs. power in predicting male success). This chapter evaluates the models to see which best predicts success at fixing bugs and under what circumstances this happens.

As mentioned in the previous chapter, the median percentage of bugs fixed for the overall population was 60%. Any participants who fixed 60% or more of the bugs were considered "successful" at bug fixing. Otherwise, they were "unsuccessful". Thus, wherever "success" or "success at fixing bugs" is mentioned in this chapter, it means the binary value of "successful" or "unsuccessful" (as opposed to the actual percentage of bugs fixed).

9.1. The Best Models for Predicting Success at Fixing Bugs

Table 5 shows how well each of the models fared for the different population groups (listed in the columns). The NB Naïve Bayes model, the CAEvents cluster analysis model (created with the statistically significant attributes – static ones that were significant for the females and event count ones that were significant for the males), and the LoR logistic regression model performed best in predicting success by the overall population. All three of these correctly predicted whether the participant was successful or not at fixing bugs for 73% of the population, which is a very good prediction rate. The best models for predicting success by males were the NN Neural Network model and the CAEvents cluster analysis model. For females, the best model was the DT Decision Tree model.

Table 5: Percentage of the target population (column headings) for which each of the mining models (row headings) correctly predicted success at fixing bugs.

	Overall	Males	Females
Association Rules Model	46%	36%	53%
Clustering Model (CA)	50%	64%	40%
Neural Networks Model	69%	**82%**	60%
Clustering Model (CA2)	46%	54%	33%
Decision Tree Model	57%	54%	**73%**
Naïve Bayes Model	**73%**	64%	60%
Clustering (CABackground)	50%	64%	40%
Clustering (CAEvents)	**73%**	**82%**	66%
Logistic Regression Model	**73%**	73%	60%

Thus, there are five models (all built on both male and female data) performing at the top for some population: CAEvents, DT, NB, LoR, and NN (see Table 5). While Naïve Bayes, Logistic Regression, and Neural Network models performed very well in predicting success at fixing bugs, it is hard to understand the logic that they used to arrive at their classification rules. Cluster Analysis and Decision Trees, on the other hand, have very clear ways of showing why they classified a particular participant as either successful or unsuccessful, based on the number of bugs fixed. Since my goal for this thesis is not only to predict success at fixing bugs, but also to get a better understanding of the factors that lead to it for males and females, these two models warrant further scrutiny, which we do in the reminder of this chapter.

9.2. Predicting Bugs Fixed By Females

As was shown in Table 5, the Decision Tree model (DT) did the best job of predicting female success at fixing bugs (73%), although it did only slightly better than random for male success (54%). Thus, if we want to know what leads to female success and the lack thereof, the decision tree model is interesting to look at.

Decision trees pick the attribute that best explains the bugs fixing success for the overall population, which in this case, is "PR Edit Formula" (see Figure 17). This model was very simple, with only one level. If participants did between 9 and 13 formula edits in Payroll (inclusive), they were very likely to be successful at fixing more bugs than the median (see Figure 18).

Some might think that this model has no explanatory power. It is a simple model and the story it tells is easy to believe: if participants actually worked on fixing the bugs without getting lost, then they were successful. We would not want to discount this model as not being useful, however. This model raises all kinds of interesting questions that will further help us design better tools. Why is this simple model about edit formulas the model that best predicted bugs fixed for females? Could software somehow notice unproductive behavior with formula edits in problem-solving environments? Why did this model not work for males? Is it because males' ranges are higher or lower? Or are other factors more important for males?

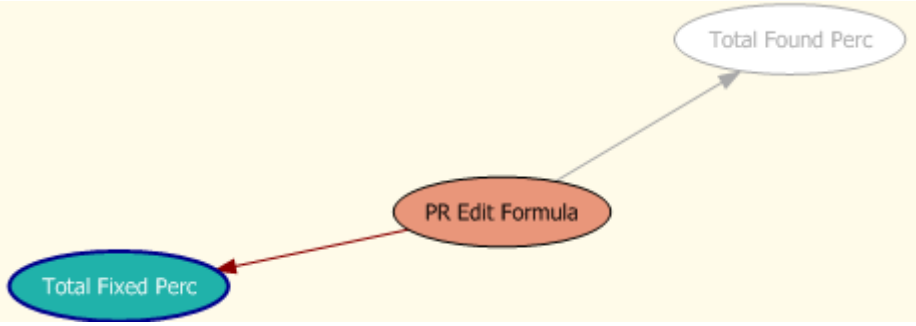


Figure 17: The number of formula edits performed in Payroll positively predicts the success at fixing bugs. A "reasonable" number of formula edits (9-13) in Payroll positively predicted a success at fixing bugs. Though the model was created using all of the participants, it only performs well in predicting female success at fixing bugs.

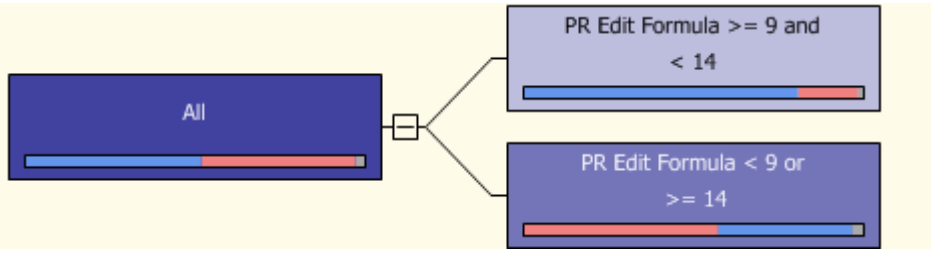


Figure 18: The darker sections of the bars depict the number of participants who fixed at least 60% of the bugs ("successful") and the lighter sections of the bars are the number of participants who fixed fewer than 60% of the bugs ("unsuccessful") per tree node. This model was run on the training data. It says that 34 out of the 41 participants who edited between 9 and 13 formulas (inclusive) were successful at fixing bugs. Also, 58 out of the 98 participants who did either fewer than 9 edits or more than 13 edits were unsuccessful at fixing bugs.

This information alone correctly predicted 73% of the female population's success at fixing bugs. Before data mining, we had not thought of seeing if there was anything other than a linear relationship between the number of formulas edited and the success at fixing bugs. Data mining helped us find this pattern, which makes sense. When a participant is faced with a complex task, such as debugging the Payroll

spreadsheet, editing either too few or too many formulas can be a sign that the user is lost. When a user is not editing many, they may be having a hard time deciding on what formulas to edit or how to edit them. While the minimum number of edits for fixing all of the bugs in Payroll was four, participants are not often able to fix all of the formulas correctly their first try. When a user is instead making many formula edits, it may be a sign that they may be having a hard time with the syntax or that they are making many changes without pausing enough to think beforehand.

Result 1: Performing either too many or too few edit formulas in complex tasks negatively predicted success at fixing bugs for females. It, not feature usage or background, was the only factor that differentiated the unsuccessful participants from the successful ones. This is shown by model "DT", which correctly predicts female success at fixing bugs 73% of the time.

9.3. Predicting Bugs Fixed By Males

The CAEvents cluster analysis model, which had an overall prediction rate of 73%, predicted male success at fixing bugs even more correctly (at a rate of 82%). Clustering algorithms divide the population into several groups that have similar attributes. Figure 19 shows what CAEvents looks like. Because of the CAEvents model's transparency and because it performed so well, this is a good model to look into in order to see which attributes led to male success and failure at fixing bugs.

As mentioned in the previous chapter, we built the CAEvents model as a competing model to the model that uses all of the inputs (model CA). Since unrelated input variables weaken the models, we decided to use only variables that we thought were highly related to success at fixing bugs. The criterion for "highly related" was whether regression analysis resulted in an input variable predicting bugs fixed in a significant manner. From an earlier modeling exploration (see Appendix C), we also

hypothesized that static data (like background factors and self-efficacy scores) would matter for females, while dynamic data (such as event counts) would matter for males. We used the same reasoning for picking the "important" variables to include in this model, only ignoring those statistically significant input values that were highly related to other input values. Table 5 shows that this model greatly outperformed model CA, for the overall population, for males only, and for females only. Figure 20 gives details about the values of each attribute for each of these four clusters.

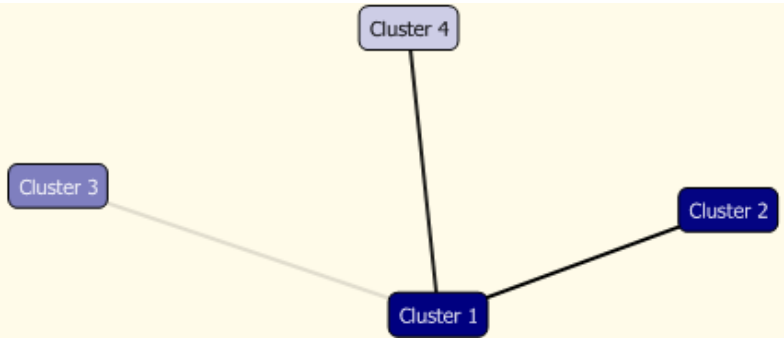


Figure 19. The darker the cluster, the more highly populated it is. The darker the line between the clusters, the more highly related those two clusters are.

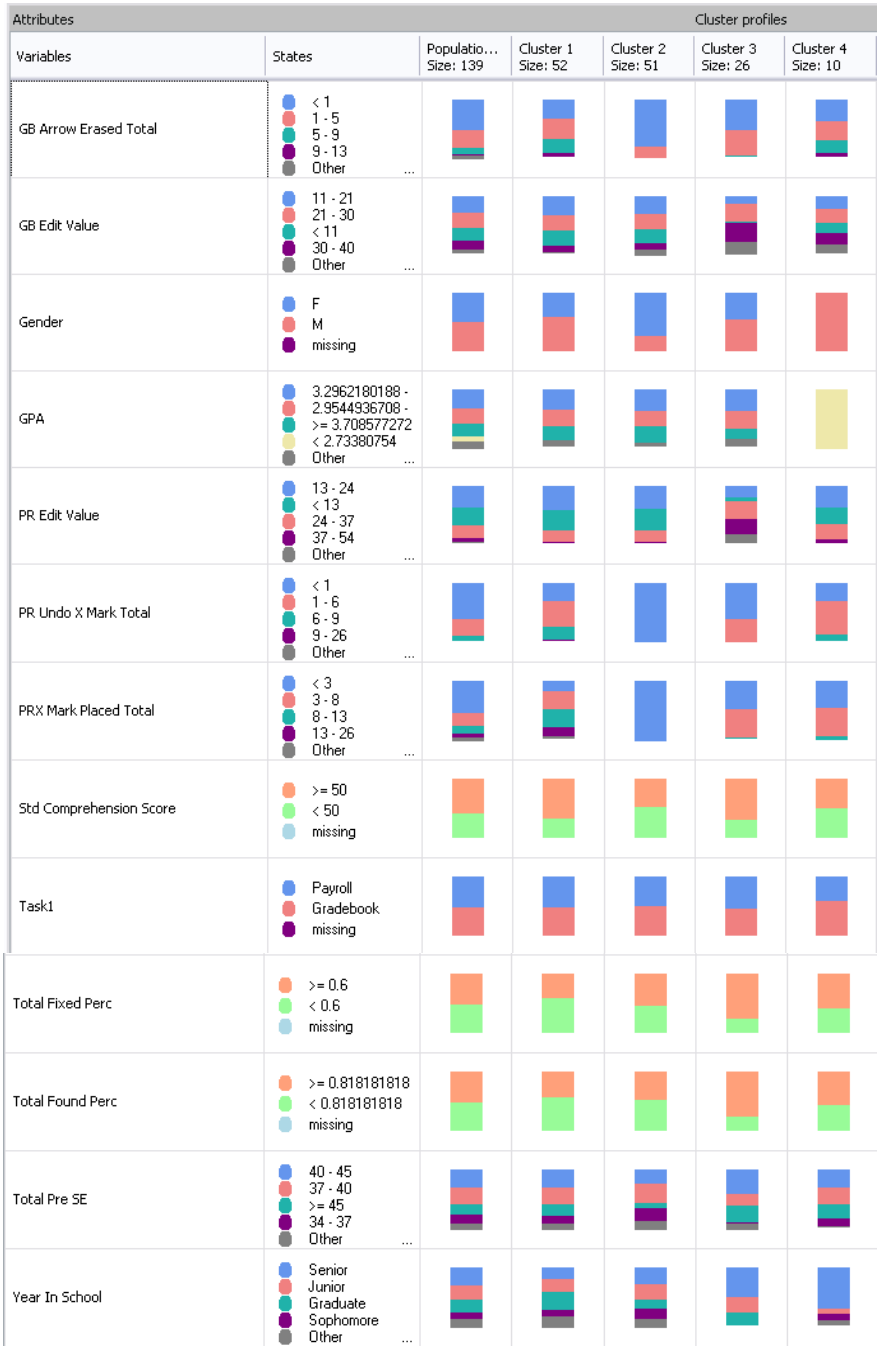


Figure 20: This is the distribution of values for each of the variables (in the rows) per cluster in the CAEvents model (in the columns).

To determine whether an attribute is of interest in a particular cluster, we compared the cluster distribution of the values for that variable to the overall distribution. Since this model worked best for the males, we analyze it only in terms of how it predicts male success at fixing bugs. Cluster 1 had a few more males than females (59% males), Cluster 2 was mostly female (73% females), Cluster 3 had more males than females (55% male), and Cluster 4 was almost completely male (98% male).

We will use the concept of "advanced features" in interpreting these clusters. Placing and taking away X-Marks and performing Arrow Erased events are "advanced" features in our studies. As mentioned in Chapter 6, X-Marks are available for users to place in cells for which they believe the value is wrong. While we told the participants that this feature existed, we did not tell them anything about the meaning of the feedback they received when performing such an action or how to act on that feedback.

After bringing up all of the arrows for a cell, a participant can pick arrows off one by one by performing an Arrow Erased event. This allows them to hide relationships between cells that they find unimportant. This feature is not conducive to playful tinkering, since it is physically a little tricky to pick off the arrow and, once the arrow is gone, it cannot be easily brought back. Arrow Erased also does not provide additional feedback or color changes, as the other features do. This feature is especially useful when looking at a cell with its formula open. Whether the formula is open or not, using it commits the participant to a choice of which relationships to pursue (an advanced concept that was not addressed in our tutorials). So while taking off all of the arrows for a cell (which is a different action from Arrow Erased) is easy to do, Arrow Erased can be considered an advanced feature in the environment.

Cluster 1 had 59% males. They were fairly average in terms of GPA, Year in School, Pre Self-Efficacy, and Edit Value. They did, however, have a high usage of the advanced features. While this group learned a lot about the features, as shown by their high Comprehension scores, they had the most unsuccessful participants out of the four clusters. One possible reason for this was that their goal may have been to *learn* about the features, rather than to *use* them to fix bugs in this new environment. Learning how to best use new features does mean that they will have less time to fix bugs. With tasks only about half an hour, this behavior could have a negative impact on their performance. However, this group of people might have turned out to be successful debuggers in the real world, where more time would be allowed for fixing bugs.

Cluster 2 is 73% female. These females had the lowest self-efficacy out of all of the groups. There is only a small group of males that this cluster used to predict the success of. The cluster is also not great at predicting success, with about half the group being successful and the other half unsuccessful. What this cluster does say is that there was a low self-efficacy group of people (mostly female) who were average in terms of all other static factors and in terms of Edit Values, but that stayed away from advanced features. (Editing values is a necessary part of testing.) While this group had slightly fewer successful participants than clusters 3 and 4, the difference was slight, which means that lack of self-efficacy in combination with low advanced feature usage did not necessarily lead to fewer bugs fixed, provided that they were still testing.

Cluster 3 was the most successful at fixing bugs out of the three groups. The population is made up mostly of juniors and seniors of both genders with high self-efficacy. While they used the advanced features only slightly, this group had high

feature comprehension scores. Another distinctive characteristic of this group is that they had very high Edit Value counts.

Finally, there is *Cluster 4*. This cluster, like Cluster 2, jumps out in terms of gender differences. It had 98% male seniors with low GPA and average self-efficacy. 100% had less than a 2.73 GPA, yet most were successful at fixing bugs. Like Cluster 1, this group had high advanced feature usage. What differentiates them from Cluster 1 (which was unsuccessful at fixing bugs) is that they also had a high number of Edit Values (testing).

Result 2: For males, an emphasis on testing was important, with both clusters that were best at fixing bugs doing the most value edits. High advanced feature usage led to fewer bugs fixed when value edits were few. Unlike for females, some males with very low GPA were successful at fixing bugs.

9.4. Improving Prediction of Bugs Fixed By Males and Females

The CAEvents model was also the one that performed best for predicting the success at fixing bugs for all participants. However, it was much better at predicting males' success (82%) than females' (66%) success. We therefore tried to improve the model's efficiency by making a change: switching the unsupervised CAEvents model to a supervised model. We named this model CAPredictBugsFixed (see Figure 21 and Figure 22). In order to switch from unsupervised to supervised, we changed the Percentage of Bugs Fixed variable (which had values of ">=60%" and "<60%") from "PredictOnly" to "Predict". This means that the bugs fixed variable was now also going to be used in training the model, not just for evaluating the model's predictive power, as before.

This modification improved the power in predicting success at fixing bugs from 73% to 77%. Note that prediction of overall male success was the same with

both models (82%). What increased was the correct prediction of females' success at fixing bugs. Overall, it rose from 66% to 73% (see Table 6).

	Overall	Males	Females
CAEvents	73%	82%	66%
CAPredictBugsFixed	**77%**	82%	**73%**

Table 6: This shows the percentage of each population grouping (column headings) by model (row headings). CAEvents was the unsupervised model and CAPredictBugsFixed was the supervised model.

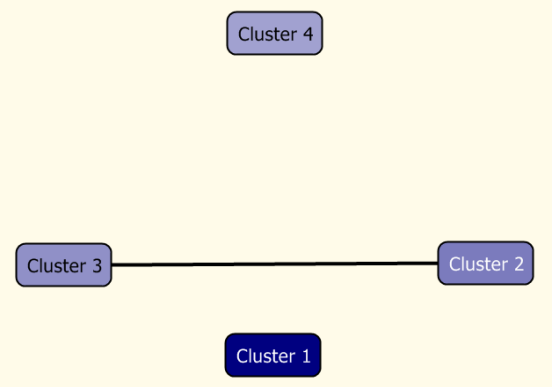


Figure 21. This is the CAPredictBugsFixed model. Darker clusters have more members and the link between Clusters 2 and 3 was the strongest.

Figure 22. These are the distribution of values for each of the variables (in the rows) per cluster in the CAPredictBugsFixed model (in the columns).

Once again, the clustering model has four groups. This time, two were mostly female (Clusters 1 and 4) and two were mostly male (Clusters 2 and 3). This tells us that gender is an important factor in finding patterns that led to success at fixing bugs.

Result 3: The combination of factors that led to success at fixing bugs for males is different from the combination of factors that led to female success at fixing bugs.

Cluster 1 was 61% female. They were average in almost every way, except for one main difference: they used no X-Marks in Payroll and very few Arrow Erased in Gradebook (i.e. low usage of advanced features). Yet, the cluster was above average in terms of success at fixing bugs.

Cluster 4 was the other female cluster (100% female), which was also above average in terms of success at fixing bugs. These were graduate students with high GPA and slightly higher self-efficacy than average. Another interesting fact about these females is that most were in sessions that started with the easier Gradebook task. They also used few advanced features, though not as few as Cluster 1. They had above average value edits in Payroll.

Result 4: Both female clusters had very low advanced feature usage and above average success. The cluster that used a few more advanced features did not perform any better than the group who did not. This suggests that the available advanced features are fairly inconsequential in terms of helping females with their debugging. Other factors matter more.

Cluster 2 was a very high advanced feature usage cluster and was mostly male (81%). This group also performed a lot of edit values. They were all over the place in terms of GPA, but had very high self-efficacy and they had more graduate students

than Cluster 1 and Cluster 3. This group had the greatest percentage of successful participants at fixing bugs.

Cluster 3 was another male group (66% male). This group did many fewer edit values than any other group. They also used fewer X-Marks than the other male group (though more than the female clusters), but more Arrow Erased events than any other cluster. Most of these participants were in sessions that started with the Payroll task. They had low self-efficacy going into the first task. All were unsuccessful at fixing bugs.

Result 5: For the two male groups, background factors like GPA did not seem to affect either the advanced features they used or their success at fixing bugs. What really seemed to help them was doing a lot of edit values (testing) and X-Marks (advanced testing).

Note that Cluster 3 was 1/3 female and 2/3 males. What characterized this cluster was a combination of low self-efficacy, starting with the Payroll task, and doing few value edits (despite using advanced testing features) led to all of these participants being unsuccessful. Because 1/3 of the group was female, we do not have evidence that SE impacted the males' success; it could be that the low amount of testing (value edits) was the determining factor for the males' lack of success.

9.5. Discussion of the Results

What we learned about factors leading to female success at fixing bugs:

- Performing either too many or too few edit formulas in complex tasks positively predicted that few bugs would be fixed.
- Different factors led to male success at fixing bugs than female success at fixing bugs (shown both by the clusters being characterized by a

specific gender and by different models performing better for the different genders).

- Females used fewer advanced features than males in general. Also, these advanced features did not seem to help the female cluster who used it more, since both clusters were above average in terms of bugs fixed.

What we learned about factors leading to male success at fixing bugs:

- An emphasis on testing through value edits was an important factor that led to male success at fixing bugs.
- Unlike for females, some very low-GPA (<2.73) males were successful at fixing bugs. (All of the participants with GPA that low were males. GPA predicted bugs fixed for females, however.)
- X-Marks were positively related to success at bugs fixing, while Arrow Erased was negatively related to success at bug fixing.

Data mining was useful, since we were able to find nonlinear relationships that we had not thought of checking before and we were also able to find complex relationships between several variables. All three final models also meet both the business and the data mining objectives. Getting a 73%, 82%, and 77% fit is quite good for human data.

10. CONCLUSION

This thesis has contributed several new results to the emerging area of Gender HCI. In this chapter, we situate our new results in the context of other findings in this area.

First we consider findings relating background factors to debugging success. So far, we have found the most important background factor to be self-efficacy. Findings related to self-efficacy include:

- Female end users had significantly lower self-efficacy (a task-specific form of confidence) than males (Beckwith, Burnett and Wiedenbeck, et al. 2005). (Other researchers have reported similar results.)
- For females, low self-efficacy was predictive of how effectively they worked with problem-solving features available in the software, unlike males (Beckwith, Burnett and Wiedenbeck, et al. 2005). This finding generalized to the MS Excel environment (Beckwith, Inman, et al. 2007).
- There was no significant difference in the success of the two genders or in learning how the features worked, implying that females' low self-efficacy about their usage of new features was not an accurate assessment of their problem-solving potential, but rather became a self-fulfilling prophecy (Beckwith, Burnett and Wiedenbeck, et al. 2005). This finding also generalized to the MS Excel environment (Beckwith, Inman, et al. 2007).

- New evidence-based hypothesis: Females with lower self-efficacy are likely to struggle longer to use a strategy that is not working well before moving on to another strategy (Study 1).

Other findings relate to background factors other than self-efficacy:

- Unlike for females, some males with very low GPA were successful at fixing bugs (Study 2).
- For the male groups, background factors like GPA did not seem to affect either the advanced features they used or their success at fixing bugs. What really seemed to help them was doing a lot of edit values (testing) and X-Marks (advanced testing) (Study 2).

Factors that relate to the features available in the environment and how those features are used by participants were also found to relate to participant success. One set of these results focuses on whether the particular feature used was of type familiar (formula edits) or type unfamiliar:

- Females significantly more often agreed with the statement, "I was afraid I would take too long to learn the [untaught feature]" (Beckwith, Burnett and Wiedenbeck, et al. 2005).
- Female end users were significantly slower to try out unfamiliar features, instead used the Type Familiar feature of formula edits (Beckwith, Burnett and Wiedenbeck, et al. 2005). This finding generalized to the MS Excel environment in an analogous form (Beckwith, Inman, et al. 2007).
- Females were significantly less likely to initially approach new features (Beckwith, Burnett and Wiedenbeck, et al. 2005).

- Even if they tried it once, females were less likely to engage in regular use of new features (Beckwith, Burnett and Wiedenbeck, et al. 2005).
- Performing either too many or too few edit formulas in complex tasks negatively predicted success at fixing bugs for females (Study 2).

While participants came in with knowledge about editing formulas, they were taught several features in a tutorial and were also encouraged to explore using some untaught features.

- New evidence-based hypothesis: Unsuccessful males overdo use of arrows – unlike successful males, successful females, and unsuccessful females (Study 1).
- The female clusters had very low advanced feature usage. The cluster that used a few more advanced features did not perform any better than the group who did not. This suggests that the available advanced features are fairly inconsequential in terms of helping females with their debugging. Other factors matter more (Study 2).
- Partial evidence: Providing ways to express uncertain or tentative judgments seems to equalize female and male usage of testing features (Beckwith, Gender HCI Issues in End-User Programming 2007).
- Partial evidence: Providing female/male video explanations of strategy hints seems promising in guiding females' strategies and encouraging more effective use of the environment (Subrahmaniyan, et al. 2007).

While some participants stick to the features they know how to use, others playfully experiment with the new features throughout the task:

- Tinkering (playfully experimenting) with features was done by males more often than females when the environment was conducive to tinkering. (Beckwith, Kissinger, et al. 2006).
- Males were comfortable with tinkering; in fact, some did it to excess (Beckwith, Kissinger, et al. 2006).
- New evidence-based hypothesis: Unsuccessful males have a tendency to tinker excessively with the features rather than using the features to accomplish their task (Study 1).
- For females, the amount of tinkering predicted success, but for males, excessive tinkering hurt them (Beckwith, Kissinger, et al. 2006).
- Pauses after any action were predictive of better understanding for both genders (Beckwith, Kissinger, et al. 2006).
- Females paused significantly more than the males did (Beckwith, Kissinger, et al. 2006).

The use of testing also mattered, at least to males. The most direct measure of testing in our environment is value edits.

- For males, an emphasis on testing was important, with clusters that were best at fixing bugs doing the most value edits (Study 2).
- For males, value edits mattered more than advanced testing feature usage. In fact, high advanced feature usage led to fewer bugs fixed if value edits were few (Study 2).

One last set of results is the overall picture that these findings paint: they have shown that different factors lead to male and female success at fixing bugs. A new

evidence-based hypothesis from Study 1 suggests that the debugging and testing strategies that help with males' success are not the right ones for female success. Then Study 2 showed that the combination of factors that led to success at fixing bugs for males was indeed different from the combination of factors that led to female success at fixing bugs. Thus, these strategies' data mining results add to the growing body of evidence that it is important to take gender differences into account when designing problem-solving software for end users.

BIBLIOGRAPHY

Bandura, A. *Social Foundations of Thought and Action.* Englewood Cliffs, NJ: Prentice Hall, 1986.

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

Beckwith, L., D. Inman, K. Rector, and M. Burnett. "On to the Real World: Gender and Self-Efficacy in Excel." *IEEE Conference on Visual Languages and Human-Centric Computing.* 2007. (to appear).

Beckwith, L. "Gender Differences in End-User Debugging Strategies." Technical Report CS07-60-01, 2007.

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

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

Burnett, M., C. Cook, and G. Rothermel. "End-User Software Engineering." *Communications of the ACM 47(9),* 2004: 53-58.

Burnett, M., 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." *Journal of Functional Programming 11(2),* 2001: 155-206.

Busch, T. "Gender differences in self-efficacyand attitudes toward computers." *Journal of Educational Computing Research,* 1995: 147-158.

Cassell, J. *Genderizing HCI.* MIT Media Lab, 1998.

Cassell, J., and H. Jenkins. *From Barbie to Mortal Kombat: Gender and Computer Games.* Cambridge, MA: MIT Press, 1998.

Chapman, P., et al. *CRISP-DM 1.0: Step-by-step data mining guide.* SPSS Inc., 2000.

Czerwinski, M., D. Tan, and G. Robertson. "Women take a wider view." *ACM Conference on Human Factors in Computing Systems.* ACM Press , 2002. 195-202.

De Angeli, A., and N. Bianchi-Berthouze. *Proceedings of Gender and Interaction, Real and Virtual Women in a Male World Workshop.* Venice, 2006.

De Angeli, A., and S. Brahnam. "Sex stereotypes and conversational agents." *Proc. of Gender and Interaction, Real and Virtual Women in a Male World Workshop.* 2006.

Elkan, C. "Magical Thinking in Data Mining: Lessons from CoILChallenge 2000." *ACM International Conference on Knowledge Discovery in Data Mining.* 2001. 426-431.

Gorriz, C., and C. Medina. "Engaging girls with computers through software games." *Communications of the ACM*, 2000: 42-49.

Grigoreanu, V., Beckwith, L., Fern, X., Yang, S., Komireddy, C., Narayanan, V., Cook, C., Burnett, M. "Gender differences in end-user debugging, revisited: What the miners found." *IEEE Symposium on Visual Languages and Human-Centric Computing.* Brighton, UK, 2006. 19-26.

Huff, C. "Gender software design and occupational equity." *ACM SIGCSE Bulletin*, 34 (2), 2002: 112-115.

KDnuggets. *Poll: What main methodology are you using for data mining?* July 2002. http://www.kdnuggets.com/polls/2002/methodology.htm (accessed March 2007).

Margolis, J., and A. Fisher. *Unlocking the Clubhouse: Women and Computing.* Cambridge, MA: MIT Press, 2001.

Rode, J., E. Toye, and A. Blackwell. "The Fuzzy Felt Ethnography - understanding the programming patterns of domestic appliances." *Personal and Ubiquitous Computing 8*, 2004: 161-176.

Rothermel, G., M. Burnett, L. Li, C. DuPuis, and A. Sheretov. "A Methodology for Testing Spreadsheets." *ACM Trans. Software Engineering and Methodology 10(1)*, 2001: 110-147.

Rothermel, G., M. Burnett, L. Li, C. DuPuis, and A. Sheretov. "A Methodology for Testing Spreadsheets." *ACM Transactions on Software Engineering and Methodology 10(1)*, 2001: 110-147.

Ruthruff, J., A. Phalgune, L. Beckwith, M. Burnett, and C. Cook. "Rewarding Good Behavior: End-User Debugging and Rewards." *IEEE Symposium on Visual Languages and Human-Centric Computing.* Rome, Italy, 2004. 115- 122.

Ruthruff, J., M. Burnett, and G. Rothermel. "An empirical study of fault localization for end-user programmers." *Int. Conf. Software Engineering.* 2005. 352-361.

Seno, M., and G. Karypis. "SLPMiner: An algorithm for finding frequent sequential patterns using length decreasing support constraint." *ICDM'02.* 2002. 418-425.

Simon, H. "The structure of ill-structured problems." *Artificial Intelligence*, 1973: 181-202.

Simon, S. "The impact of culture and gender on web sites: An empirical study." *The Data Base for Advances in Information Systems, 32(1),* 2001: 18-37.

Strauss, A., and J. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory.* Second Edition, Thousand Oaks, CA: SAGE Publications, 1998.

Tan, S., M. Czerwinski, and G. Robertson. "Women go with the (optical) flow." *ACM Conference on Human Factors in Computing Systems.* 2003. 209-215.

Turkle, S. "Computational reticence: Why women fear the intimate machine." In *Technology and Women's Voices*, by C. Kramerae, 41-61. 1988.

Wirth, R., and J. Hipp. "CRISP-DM: Towards a standard process model for data mining." *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining.* 2000. 29-39.

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

APPENDICES

APPENDIX A. (DATA PREPARATION) DATA TABLES

These are what the tables looked like for my data source. We created several data source views which we mined. Those views were created using combinations of subsets of data from the original tables in this Appendix.

Events Table		
Field Name	Description	Notes
EventID	One number for each event.	For faster access.
Event	The name of the event performed in a line of a log file.	This is the primary key for this table. Examples of events: arrows on, cell-FL-changed, HMT, checkmark.
Type	The type of the event. Each type has characteristics particular to it.	The three types: User, System, HMT.

Table 7: Event fields, description, and notes about them.

Cell Data		
Field Name	Description	Notes
CellName	What the cell was named as.	This is the primary key for this table. Examples of cell names: Course_Avg, Quiz2, DentalInsurancePremium, D3, G18.
CellID	This is a unique ID for each cell that gets created. Because the Summer2006 study required the cells to be moved around, each cell with a specific ID might have up to two different cell names.	Examples of cell IDs: CELL32684231-3568, CELL32521762-4768.
OriginalCellID	Unfortunately, when cells got moved around for the Summer2006 data, three of them got recreated, which generated a new ID for them. In this field, the ID of those three cells is the ID that they would have had, had they only been moved, rather than recreated.	The three cells which were recreated instead of being moved: CELL33617219-8238, CELL33617219-8241, and CELL33617219-8239. These three cells were new formula cells that appear only in Summer2006: CELL33617219-9307, CELL33617219-8168, and CELL33617219-8236.
Workbook	The forms spreadsheet that the participant had to debug.	This is either Gradebook or Payroll in all cases.
TypeEdit	When a user “edits” a cell, it is sometimes useful to know whether the cell is a formula cell or simply a value.	The two types of cells are value cells and formula cells. Value cells contain constants, while formula cells contain references to other cells.

WO	The Western Reading Order of the cell in the spreadsheet. If we list the cells starting from the top left, moving right and then down a row, continue right, and so on, this is the place that the cell occupies in this list.	For the Summer2005 and Winter2004 studies, the cells have values between 1 and 19 for Gradebook and 1 and 24 for Payroll. For Summer 2006, the values for Gradebook range from 1 to 22 (because of the three new formula cells) and also 1 to 24 for Payroll.
CO	Column Order is a similar rating. If the list were now made starting from the top left, first going down, then up and right and down the next column, this is the location that the particular cell would fall in that list.	One difference here is that two cells shared the same column in the Payroll spreadsheets used by Winter2004 and Summer2005. There are therefore two cells that placed 7^{th} in that line. So, the values go from 1 to 23 in Payroll for those two studies.
ShortestDist	This is the shortest distance between an input cell and the cell that the value is listed for. It is a measure of how many levels away the inputs are.	The values for shortest distance range from 0 to 3: 0 meaning that this cell is an input cell, 1 meaning it references an input cell, etc.
Buggy	The Buggy field tells us whether or not this cell's formula contained a bug at the beginning of the task.	The possible entries for this field are: TRUE (the cell was buggy) and FALSE (the cell formula did not contain a bug).
Row	The row that the cell is in.	For Summer2005 and Winter2006, there were a total of four rows in Gradebook and six rows in Payroll. In the Summer2006 data, the values range from 1 to 15 (though value/formula cells were only in rows 3, 9, and 15) in Gradebook and 1 through 18 in Payroll (though value/formula cells were

		only in rows 3, 8, 13, and 18).
Column	The column that the cell is in.	There were 6 columns in Gradebook and 6 columns in Payroll for Winter2004 and Summer2005. The Medicare cell in Payroll was listed as in column 2.5, since it shared column 2 with another cell. For Summer2006, there were 13 in Gradebook and 8 in Payroll.
Input Middle Output	Is it an input cell, middle, or output cell?	Input cells are all of the value cells (are not dependent on any other cells). Middle cells are those cells that both reference another cell and are also referenced by at least one other cell. Output cells are those cells that are not referenced by any other cells.
Sink1	Does this cell affect Sink1's value? (The output cells are the sinks.)	We do not know how to use this sink data.
Sink2	Does this cell affect Sink2's value?	
DescriptionO.	The order in which cells are listed on the description handout.	Summer2006 did not have a specific description order, though people following description order could still be identified qualitatively.
ExampleO.	The order in which cells are listed on the example handout.	There is no such order for the Summer2006 study.

Table 8: Cell data fields, descriptions of those fields, and notes about them.

Static Data		
Field Name	Description	Notes
ID	The participant's ID number.	This is the primary key: the ID number is unique to each participant.
Study	The study that the data was collected for.	The data comes from three studies: Winter2004, Summer2005, and Summer1006.
Gender	The participant's gender.	M or F.
Major	The participant's major.	Popular majors were Business, Animal Science, and Forestry.
YearInSchool	The participant's year in college or graduate school.	Possible values: Freshman, Sophomore, Junior, Senior, Post Bac., Graduate.
GPA	Participant's current cumulative grade point average.	Range: 2 to 4.
CSExpRating	This is a rating of the amount of programming experience that the participant has.	The raw data was a set of three numbers for the amount of years that the participant programmed for (in high school, college, or professionally). This number ranged from 0 to 12.

		We gave the participant a rating based on that number. The possible ratings were as follows: None (0), Some (>0 but <4), and Many (>=4).
ProgLang	The programming languages that some participants used in the past.	This data is not available for Winter2004 entries. Examples: HTML, VB, C, C++.
ProfEUProgOrProgExp	Did the participant state that they had professional experience as an end-user programmer (HTML, MatLab, etc.) or actual programming experience (C++, Java, etc.)?	Yes or No.
SSExpRating	This is a rating of the amount of spreadsheet experience that the participant has.	The raw data was a set of four numbers for the amount of spreadsheets that the participant created (in high school, college, professionally, or for personal use). This number ranged from 0 to 450. We gave the participant a rating based on that number. The possible ratings were as follows: Few (<5), Some (>=5 but <20), Several (>=20 but <50), and Many

		(>50).
ProfSSExp?	Did the participant state that they had professional experience in spreadsheet usage?	Yes or No.
English?	Is English the participant's first language?	Yes or No.
NrEnglishYears	If English is not the participant's first language, this is the number of years that they have spoken English for.	These values range from 1 to 25.
TotalPreSE	The participant's total Self-Efficacy before the first task.	The lowest total pre self-efficacy score was 25 and the highest was 50.
TotalPostSE	The participant's total Self-Efficacy score after the second task.	Winter2004 records do not include this data. The lowest total post self-efficacy score was 23 and the highest 50 again.
E15(Curved_Midterm3)Found	Did the participant find the curved midterm3 bug?	1 if Yes, 0 if No.
E15(Curved_Midterm3)Fixed	Did the participant fix the curved midterm3 bug?	1 if Yes, 0 if No.
F15Found	Did the participant find	1 if Yes, 0 if No.

	the f15 bug?	This bug was in one of the additional cells introduced during the Summer2006 study; it is in neither Summer2005 nor Winter2004.
F15Fixed	Did the participant fix the f15 bug?	1 if Yes, 0 if No. This bug was in one of the additional cells introduced during the Summer2006 study; it is in neither Summer2005 nor Winter2004.
I15(Course_Avg)Found	Did the participant find the course average bug?	1 if Yes, 0 if No.
I15(Course_Avg)Fixed	Did the participant fix the course average bug?	1 if Yes, 0 if No.
K9(Quiz_Avg)Found	Did the participant find the quiz average bug?	1 if Yes, 0 if No.
K9(Quiz_Avg)Fixed	Did the participant fix the quiz average bug?	1 if Yes, 0 if No.
L9(Midterm_Avg)Found	Did the participant find the midterm average bug?	1 if Yes, 0 if No.
L9(Midterm_Avg)Fixed	Did the participant fix the midterm average bug?	1 if Yes, 0 if No.
M9(Exam_Avg)Found	Did the participant find the exam average bug?	1 if Yes, 0 if No.
M9(Exam_Avg)Fixed	Did the participant fix	1 if Yes, 0 if No.

	the exam average bug?	
GBFound	The total number of bugs found by the participant in the Gradebook spreadsheet.	These values range from 0 to 6. Only the Summer2006 Gradebook spreadsheet contained the sixth bug, however.
GBFixed	The total number of bugs fixed by the participant in the Gradebook spreadsheet.	These values range from 0 to 6. Only the Summer2006 Gradebook spreadsheet contained the sixth bug, however.
B8-1(SocSec-1)Found	Did the participant find the first social security bug?	1 if Yes, 0 if No.
B8-1(SocSec-1)Fixed	Did the participant fix the first social security bug?	1 if Yes, 0 if No.
B8-2(SocSec-2)Found	Did the participant find the second social security bug?	1 if Yes, 0 if No.
B8-2(SocSec-2)Fixed	Did the participant fix the second social security bug?	1 if Yes, 0 if No.
C18(AdjustedGrossPay)Found	Did the participant find the adjusted gross pay bug?	1 if Yes, 0 if No.
C18(AdjustedGrossPay)Fixed	Did the participant fix the adjusted gross pay	1 if Yes, 0 if No.

	bug?	
D3(SingleWithHold)Found	Did the participant find the single withhold bug?	1 if Yes, 0 if No.
D3(SingleWithHold)Fixed	Did the participant fix the single withhold bug?	1 if Yes, 0 if No.
E3(MarriedWithhold)Found	Did the participant find the married withhold bug?	1 if Yes, 0 if No.
E3(MarriedWithhold)Fixed	Did the participant fix the married withhold bug?	1 if Yes, 0 if No.
PRFound	The total number of bugs found by the participant in the Payroll spreadsheet.	Ranges between 0 and 5.
PRFixed	The total number of bugs fixed by the participant in the Payroll spreadsheet.	Ranges between 0 and 5.
TotalFound	The total number of bugs found by the participant over both tasks.	Ranges between 0 and 11. Only the Summer2006 students could have gotten 11. The others had a maximum of 10 bugs found or fixed. In particular, the values ranged from 0 to 10 for the Summer2005 study, from 1 to 11 for the Summer2006 study, and from 3 to 10 for the

		Winter2004 study.
TotalFixed	The total number of bugs fixed by the participant over both tasks.	Ranges between 0 and 11. Only the Summer2006 students could have gotten 11. The others had a maximum of 10 bugs found or fixed. In particular, the values ranged from 0 to 10 for the Summer2005 study, 0 to 11 for the Summer2006 study, and from 0 to 9 for the Winter2004 study.
Task1	This is the first task that the participant was given.	Possible values are Gradebook and Payroll.

Table 9: Static data fields, description, and notes about them.

Log Files Data	
Field Name	Description
Index	This is particular to the database use. An automatically generated number that we used as the primary key for the table. This means that each one of the records had a unique index.
Study	Which study did the log files come from initially: Winter2004, Summer2005, or Summer2006? This information was added in by me.
Subject	The participant's subject number was also added in by me. These ID's are the same ones that appear in the background data table.
Workbook	What workbook were they working on: Gradebook or Payroll? This information comes directly from the log files.
Seconds	The time in the log files is saved in the form: hh:mm:ss. We changed it into seconds and subtracted the first time from all of them to be able to compare the progress of different participants, even if they did not start at the same time.
CellName	The cell name was taken directly from the log file.
CellID	The cell ID was taken directly from the log file.
Event	The event is the action performed by the user, this is also taken directly from the log file. System events, however, we took out of the records and stuck them into their own field, as we will see in a bit.
PercentTested	This used to be one of the system events. Whenever a change was made, this information would come up as a new record. In order to be able to best analyze how the

	users' events related to this change, we decided to stick this information in a field. This way, we know what the spreadsheet testedness was when the participant performed certain actions.
CTC	This is the cell testedness for the particular cell that the user event was performed on. Turning the system events into additional fields for each user event becomes even more important for cases like this one and the cell fault-likelihood. It might be interesting, for example to know whether a checkmark was placed on a cell that was the least tested.
CFC	This is the cell's fault-likelihood when a particular user event was performed on the cell. Was the darkest cell's formula edited, for example? This information was also taken from the log file, though we had to write a script to turn it into a field for all user events.
FormulaStatus	When this particular event was performed, was the formula for this cell open or closed? This is one of the user events from the log files, but it needed a bit of massaging to get this information at any point in time.
OpenAt	If the status of the formula is "Open", then when was that formula opened? We also massaged data from the log files a bit to get this.
CellWO	For the particular cell that is being touched, what does the cell data table say its Western Reading Order (not score) is? We added this field to make it easy to debug the Western Reading Order score calculation. We used information from the log files and the cell data table to get this information.
CellCO	For the particular cell that is being touched, what does the cell data table say its Column Order (not score) is? We added this field to make it easy to debug the Column Order score calculation. We used information from the log files

	and the cell data table to get this information.
Buggy	In the beginning of the task, did this cell contain a bug? We used information from the log files and the cell data table to get this information.
ShortestDistance	What level is that cell on in dataflow order? We used information from the log files and the cell data table to get this information.

Table 10: Log file data fields, descriptions of those fields, and notes about them.

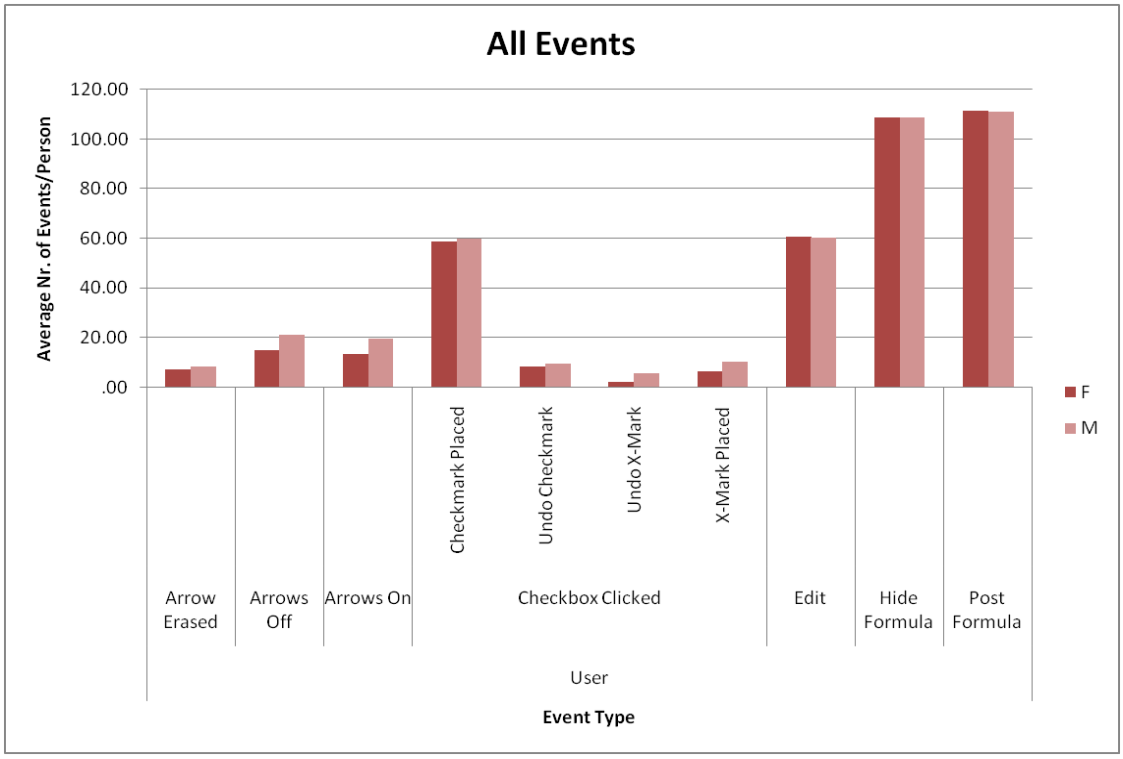
APPENDIX B. (DATA UNDERSTANDING) DESCRIPTIVE STATISTICS AND REGRESSION ANALYSES

The research questions we had in mind while conducting this data exploration were:

1. What do females' and males' use of the Forms/3 features look like over time?
2. What happens when testing features, formula operations, and value operations are looked at separately?
3. How similar are these feature usage profiles between the two tasks (Gradebook and Payroll)?
4. How does feature usage relate to bugs fixed?

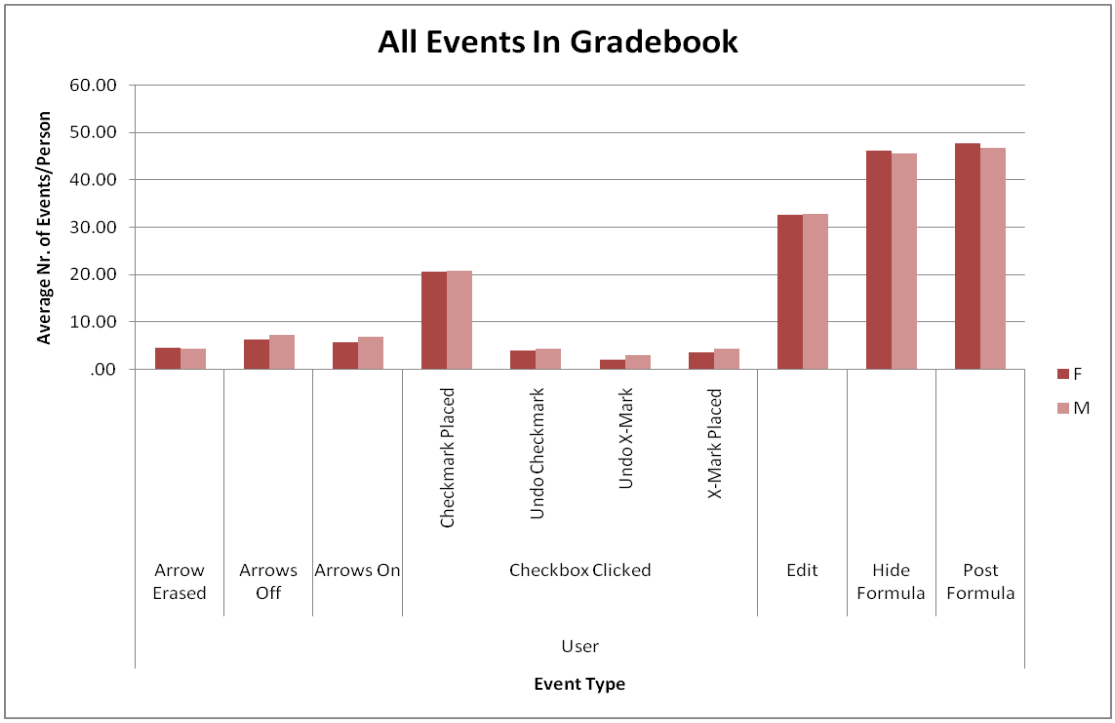
This exploration involved a series of calculations and data visualization iterations to get a better understanding of successful male and female behavior. In addition to helping with the understanding of successful behavior, calculations like counts of events by time period also help with the understanding of the data itself.

Results about All Events



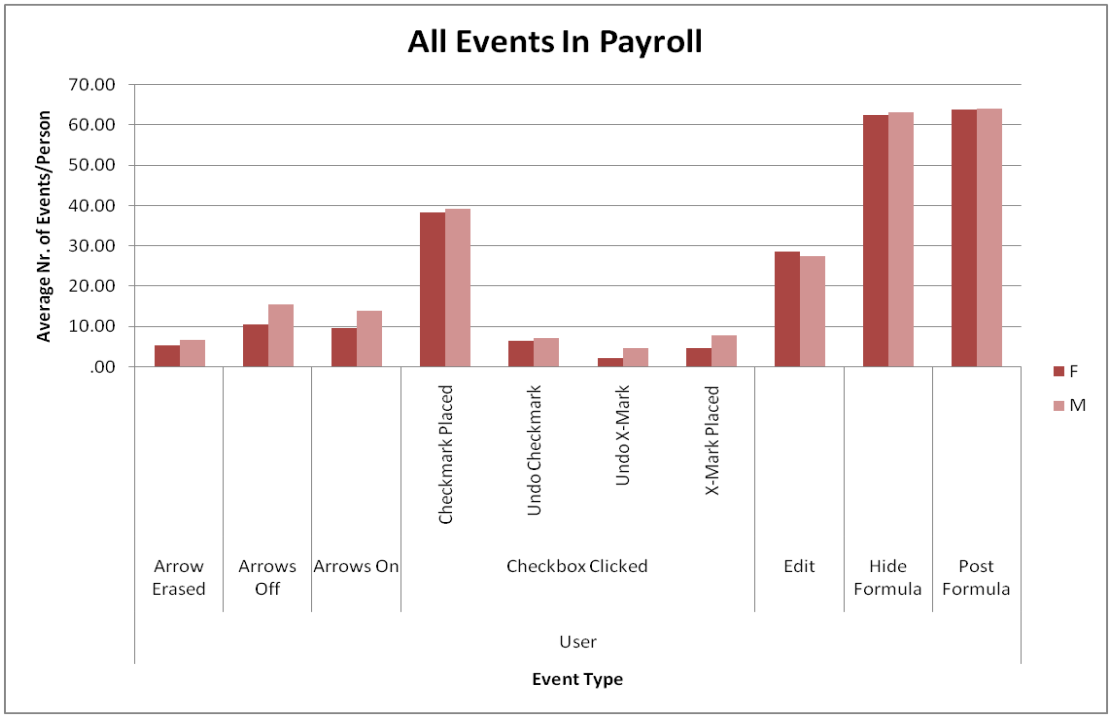
Workbook	All		
EventCount	**Column Labels**		
Row Labels	**F**	**M**	**Grand Total**
User	**382.38**	**405.30**	**393.30**
Arrow Erased	**7.31**	**8.33**	**7.87**
Arrows Off	**15.18**	**21.40**	**18.29**
Arrows On	**13.72**	**19.66**	**16.66**
Checkbox Clicked	**72.12**	**80.56**	**76.14**
Checkmark Placed	58.84	59.93	59.36
Undo Checkmark	8.58	9.77	9.14
Undo X-Mark	2.49	5.96	4.47
X-Mark Placed	6.48	10.46	8.54
Edit	**60.88**	**60.48**	**60.69**
Hide Formula	**108.95**	**108.80**	**108.88**
Post Formula	**111.57**	**111.05**	**111.32**
Grand Total	**382.38**	**405.30**	**393.30**

Figure 23: Counts of all events in both workbooks.



Workbook	GRADEBOOK		
EventCount	**Column Labels**		
Row Labels	**F**	**M**	**Grand Total**
User	**160.52**	**166.01**	**163.14**
Arrow Erased	**4.47**	**4.19**	**4.29**
Arrows Off	**6.25**	**7.12**	**6.72**
Arrows On	**5.57**	**6.77**	**6.21**
Checkbox Clicked	**25.08**	**27.22**	**26.10**
Checkmark Placed	20.66	20.84	20.75
Undo Checkmark	3.92	4.34	4.13
Undo X-Mark	1.92	3.00	2.68
X-Mark Placed	3.57	4.25	3.93
Edit	**32.55**	**32.82**	**32.68**
Hide Formula	**46.31**	**45.65**	**45.99**
Post Formula	**47.76**	**46.81**	**47.31**
Grand Total	**160.52**	**166.01**	**163.14**

Figure 24: Counts of all events in Gradebook.



Workbook	PAYROLL		
EventCount	**Column Labels**		
Row Labels	**F**	**M**	**Grand Total**
User	**221.86**	**239.29**	**230.16**
Arrow Erased	**5.36**	**6.84**	**6.11**
Arrows Off	**10.72**	**15.67**	**13.18**
Arrows On	**9.66**	**14.02**	**11.80**
Checkbox Clicked	**47.29**	**53.64**	**50.31**
Checkmark Placed	38.39	39.32	38.83
Undo Checkmark	6.48	7.13	6.79
Undo X-Mark	2.23	4.73	3.69
X-Mark Placed	4.64	7.90	6.38
Edit	**28.62**	**27.66**	**28.16**
Hide Formula	**62.64**	**63.15**	**62.88**
Post Formula	**63.81**	**64.24**	**64.02**
Grand Total	**221.86**	**239.29**	**230.16**

Figure 25: Counts of all events in Payroll.

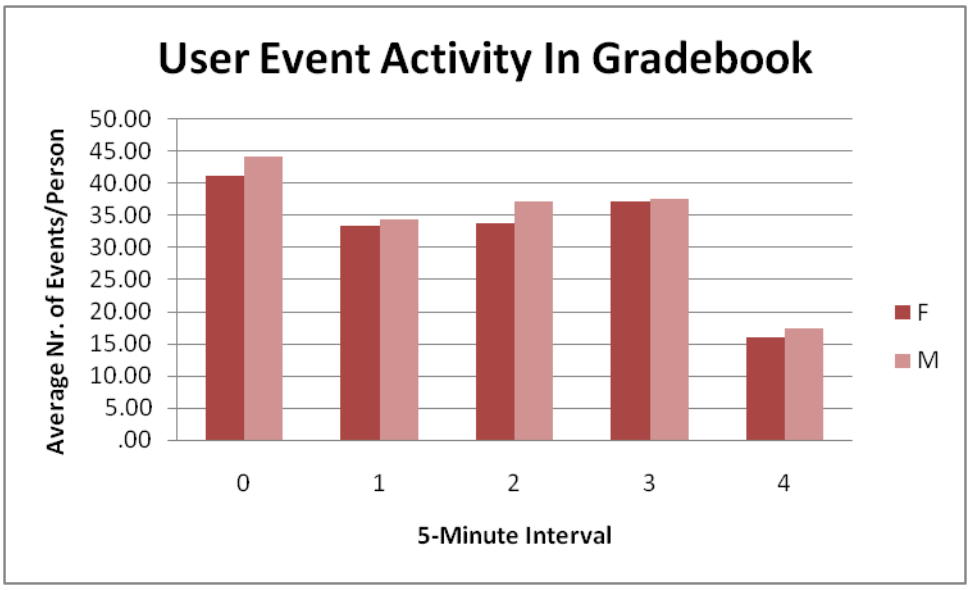


Figure 26: Graph of event counts over time in Gradebook for males and females.

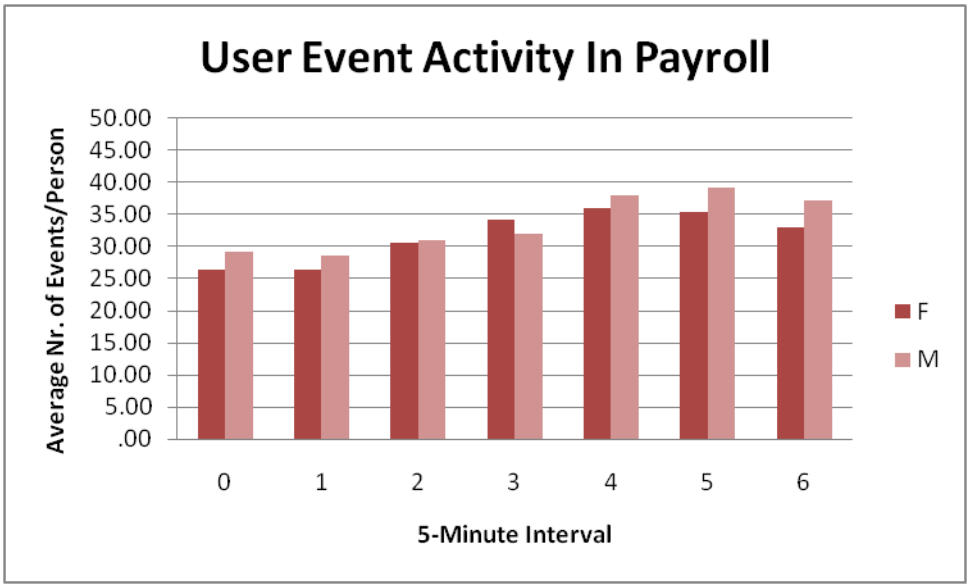


Figure 27: Graph of event counts over time in Payroll for males and females.

Workbook Hierarchy	GRADEBOOK (Multiple Items)		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	41.27	44.10	42.62
1	33.36	34.48	33.89
2	33.78	37.16	35.38
3	37.13	37.60	37.35
4	16.07	17.43	16.69
Grand Total	160.00	165.87	162.80

Workbook Hierarchy	PAYROLL (Multiple Items)		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	26.51	29.27	27.83
1	26.45	28.56	27.46
2	30.53	30.91	30.71
3	34.21	32.07	33.19
4	35.97	38.01	36.94
5	35.41	39.12	37.20
6	32.92	37.14	34.97
Grand Total	219.64	234.68	226.81

Figure 28: Counts of events in Gradebook and Payroll by males and females.

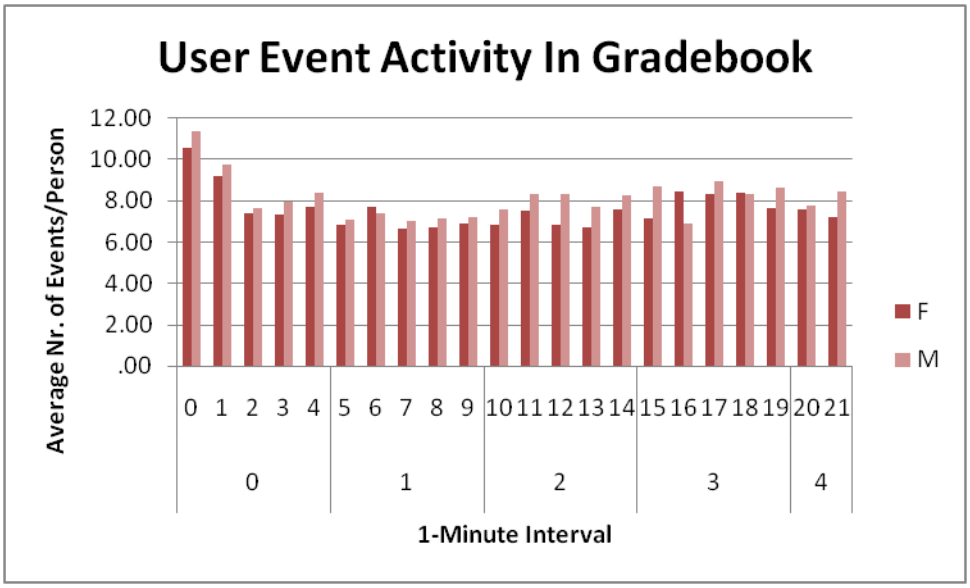


Figure 29: User activity in Gradebook per minute by males and females.

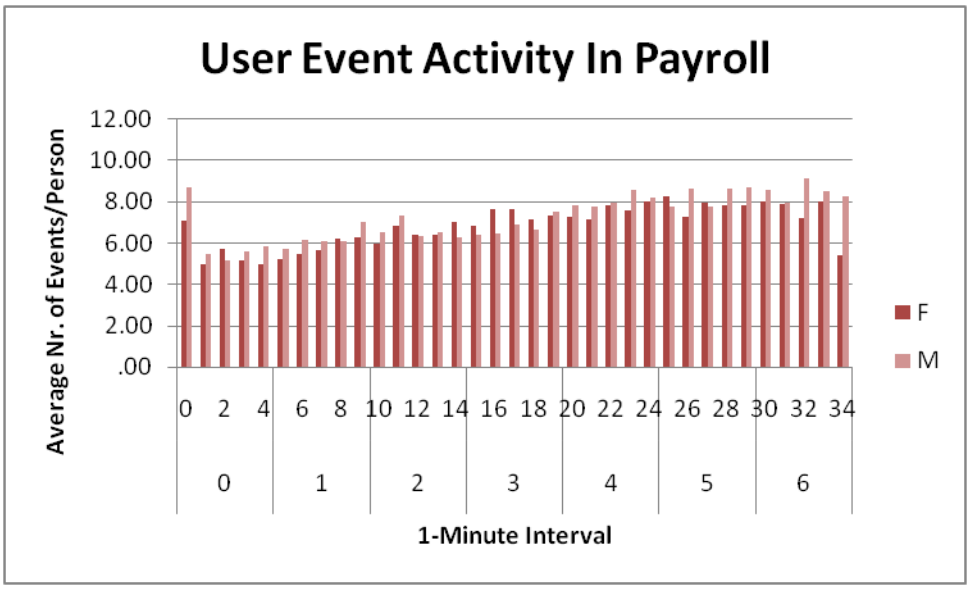


Figure 30: User activity in Payroll per minute by males and females.

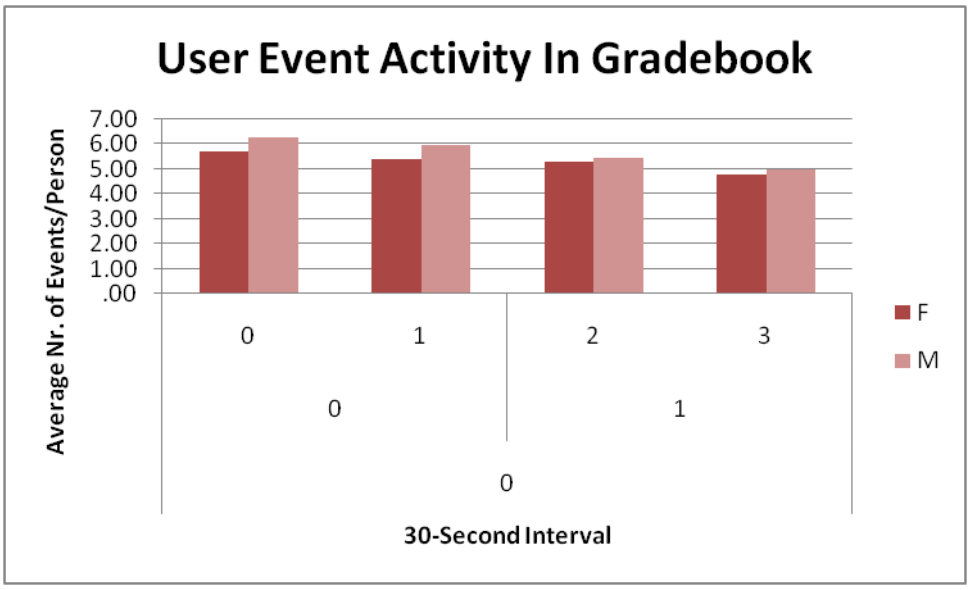


Figure 31: User activity in Gradebook in the first two minutes by males and females.

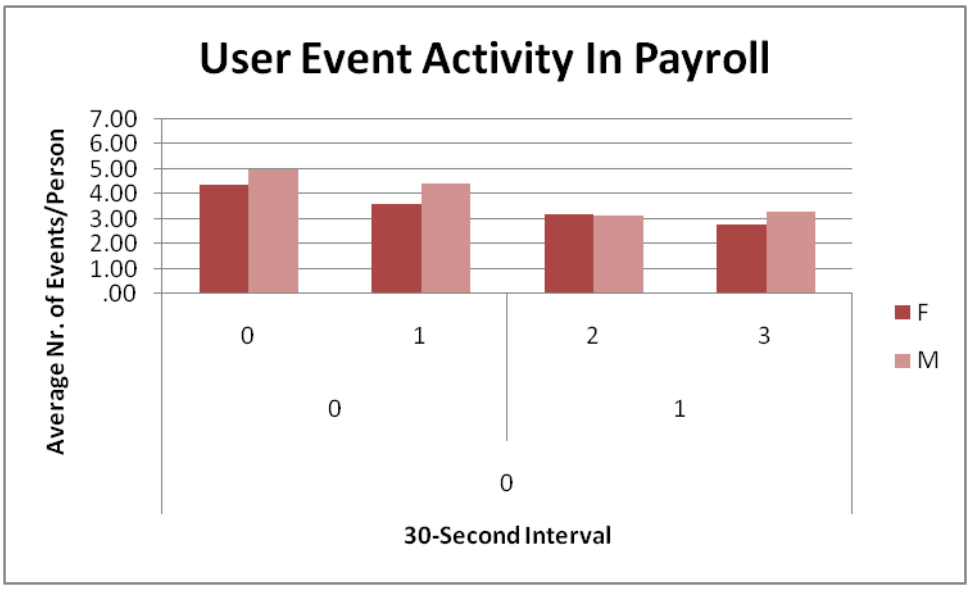


Figure 32: User activity in Payroll in the first two minutes by males and females.

Results About Testing Features

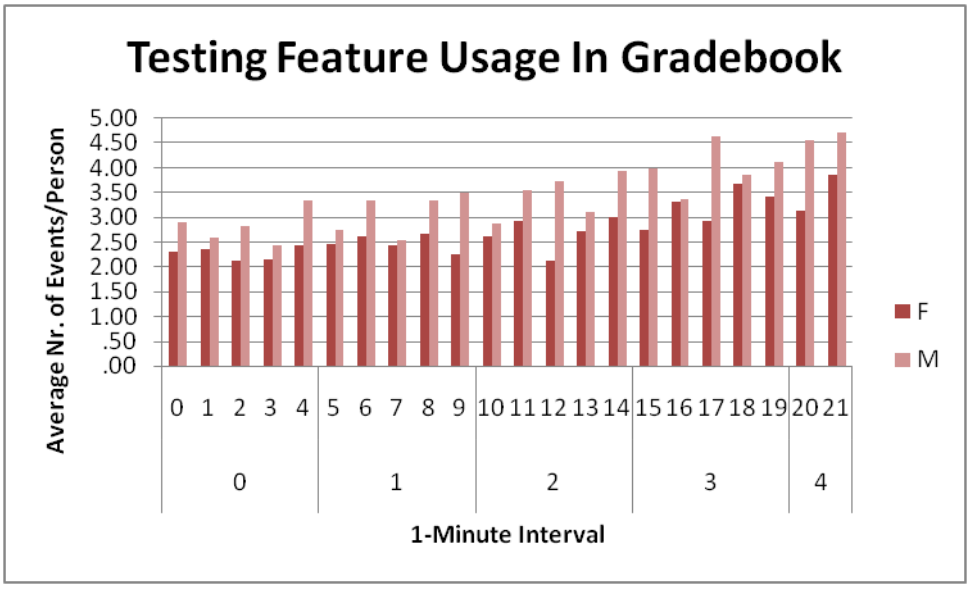


Figure 33: Testing feature activity in Gradebook by males and females.

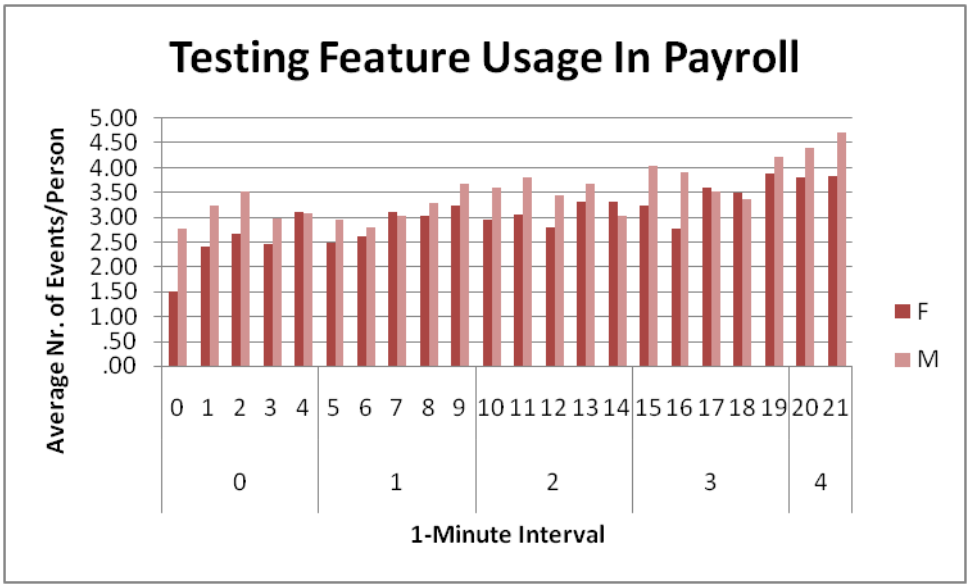


Figure 34: Testing feature activity in Payroll by males and females.

Workbook	GRADEBOOK		
Hierarchy	(Multiple Items)		
EventCount	**Column Labels**		
Row Labels	**F**	**M**	**Grand Total**
⊞0	6.00	7.85	75.93
⊞1	8.40	9.80	19.95
⊞2	8.20	10.64	50.97
⊞3	10.06	12.59	53.83
⊞4	5.31	6.75	5.96
Grand Total	**32.78**	**39.87**	**36.16**
Workbook	PAYROLL		
Hierarchy	(Multiple Items)		
EventCount	**Column Labels**		
Row Labels	**F**	**M**	**Grand Total**
⊞0	6.50	8.10	7.30
⊞1	8.51	9.76	9.16
⊞2	10.24	11.54	10.87
⊞3	11.33	11.93	11.63
⊞4	12.38	15.26	13.76
⊞5	13.19	15.89	14.50
⊞6	12.30	15.52	13.83
Grand Total	**66.17**	**82.14**	**73.78**

Figure 35: Gradebook and Payroll testing feature activity in 5-minute intervals by males and females.

Workbook	GRADEBOOK		
Hierarchy	Checkmark Placed		
CellType	All		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	4.01	4.30	4.15
1	4.91	4.97	4.94
2	5.31	6.70	5.96
3	6.34	6.43	6.38
4	4.20	4.27	4.23
Grand Total	20.60	20.78	20.68

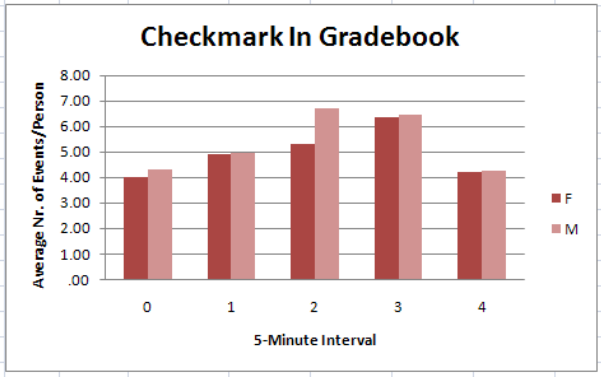


Figure 36: Checkmarks in Gradebook.

Workbook	PAYROLL		
Hierarchy	Checkmark Placed		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	4.17	3.95	4.07
1	5.34	5.18	5.25
2	6.71	5.86	6.31
3	6.42	6.09	6.26
4	7.56	8.10	7.82
5	7.87	7.92	7.90
6	6.75	7.37	7.03
Grand Total	37.90	38.11	38.00

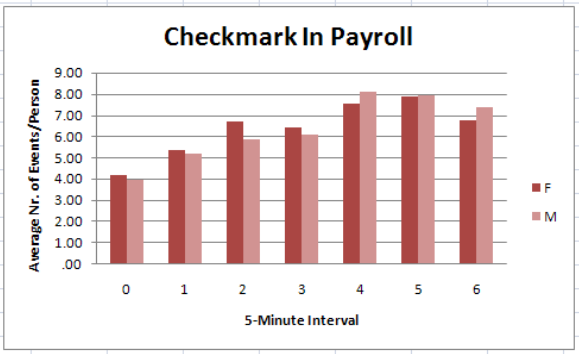


Figure 37: Checkmarks in Payroll.

Workbook	GRADEBOOK
Hierarchy	Undo Checkmark
CellType	All

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	2.00	2.33	2.16
1	2.08	1.96	2.02
2	1.80	2.16	1.98
3	1.92	2.52	2.18
4	1.78	2.71	2.19
Grand Total	**3.89**	**4.34**	**4.11**

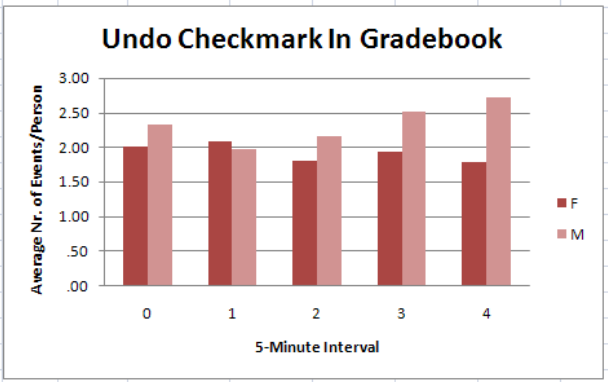


Figure 38: Undo checkmarks in Gradebook.

Workbook	PAYROLL
Hierarchy	Undo Checkmark

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	1.44	1.33	1.39
1	2.36	2.05	2.22
2	2.22	2.09	2.16
3	1.93	2.78	2.30
4	2.60	2.28	2.45
5	2.69	3.22	2.96
6	2.13	2.79	2.47
Grand Total	**6.41**	**7.03**	**6.70**

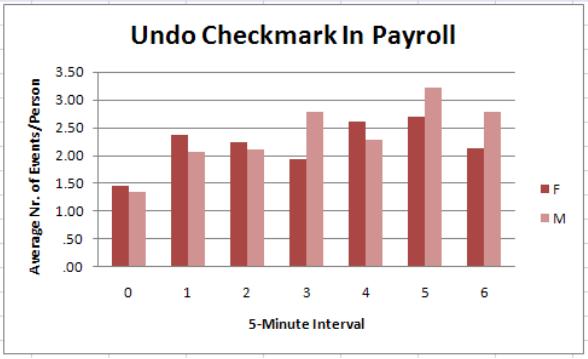


Figure 39: Undo checkmarks in Payroll.

Workbook	GRADEBOOK
Hierarchy	X-Mark Placed
CellType	All

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	1.67	1.40	1.54
1	2.00	2.40	2.22
2	1.83	2.17	2.02
3	1.60	1.92	1.76
4	1.45	2.07	1.81
Grand Total	**3.57**	**4.25**	**3.93**

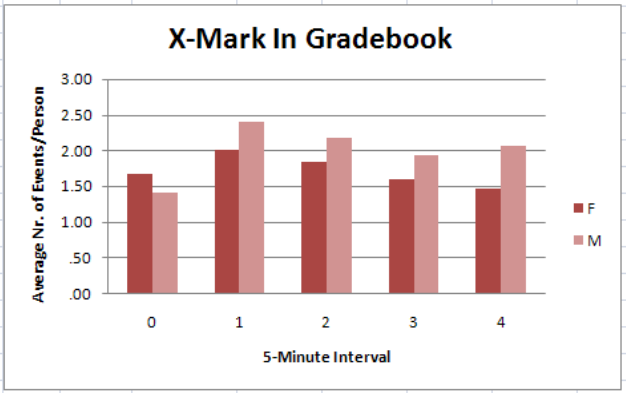


Figure 40: X-marks in Gradebook.

Workbook	PAYROLL
Hierarchy	X-Mark Placed

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	2.15	2.43	2.32
1	1.67	2.52	2.12
2	2.14	3.07	2.67
3	1.76	1.96	1.87
4	1.70	2.77	2.35
5	2.06	3.52	2.98
6	1.65	2.59	2.19
Grand Total	**4.61**	**7.87**	**6.35**

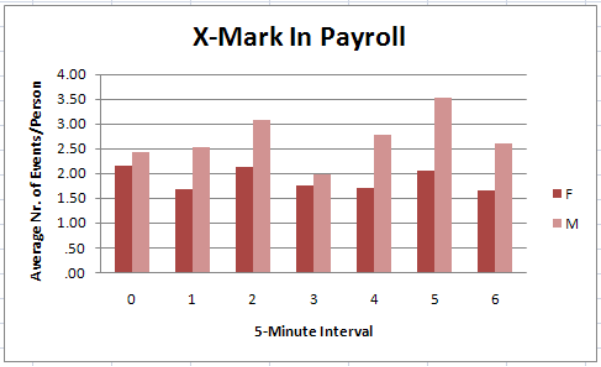


Figure 41: X-marks in Payroll.

Workbook	GRADEBOOK
Hierarchy	Undo X-Mark
CellType	All

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	1.50	1.25	1.33
1	1.40	2.25	1.92
2	1.40	2.08	1.89
3	1.00	1.67	1.50
4	1.00	1.75	1.60
Grand Total	1.92	3.00	2.68

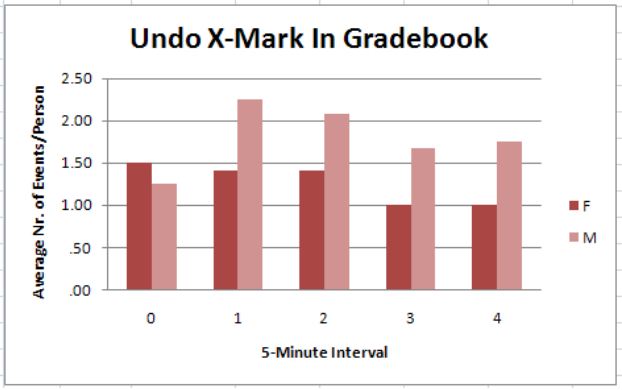


Figure 42: Undo X-marks in Gradebook.

Workbook	PAYROLL
Hierarchy	Undo X-Mark

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	1.60	1.57	1.58
1	1.00	1.63	1.43
2	1.38	2.85	2.29
3	1.00	1.62	1.44
4	1.67	1.65	1.65
5	1.67	3.13	2.60
6	1.71	2.06	1.96
Grand Total	2.19	4.68	3.65

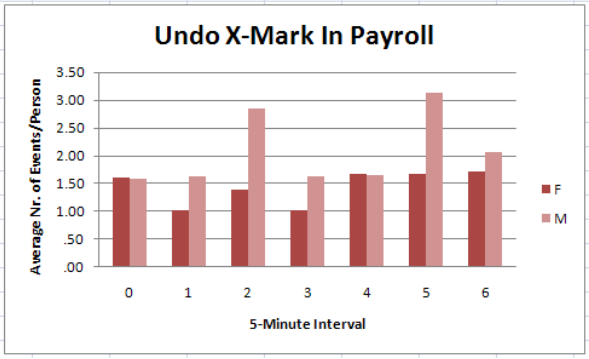


Figure 43: Undo X-marks in Payroll.

Workbook	GRADEBOOK		
Hierarchy	Arrow Erased		
CellType	All		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	3.25	2.40	2.64
1	2.83	3.06	2.97
2	2.23	2.25	2.24
3	2.00	2.48	2.31
4	3.10	2.56	2.84
Grand Total	4.43	4.19	4.28

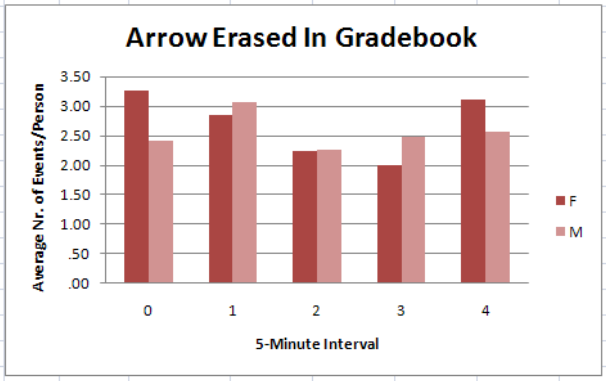


Figure 44: Arrow Erased in Gradebook.

Workbook	PAYROLL		
Hierarchy	Arrow Erased		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	3.50	2.25	2.50
1	2.44	1.71	2.13
2	4.00	3.18	3.44
3	3.13	2.64	2.92
4	2.22	5.20	3.79
5	2.17	3.05	2.57
6	6.27	3.45	4.45
Grand Total	5.34	6.75	6.05

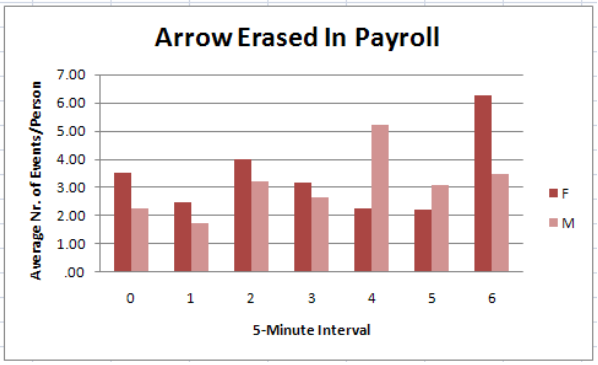


Figure 45: Arrow Erased in Payroll.

Results about Event Activity on Value Cells

Workbook	GRADEBOOK
Hierarchy	Edit
CellType	Value

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	8.00	7.99	7.99
1	5.25	5.64	5.43
2	6.71	7.16	6.93
3	7.88	8.26	8.05
4	3.76	6.04	4.67
Grand Total	21.37	21.84	21.59

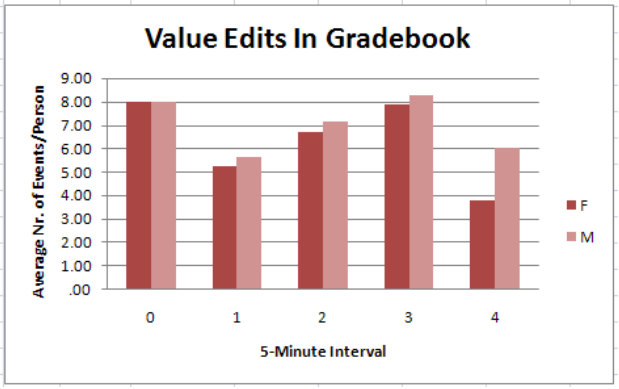


Figure 46: Value Edits in Gradebook.

Workbook	PAYROLL
Hierarchy	Edit
CellType	Value

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	3.69	4.04	3.86
1	3.51	4.18	3.79
2	3.70	3.95	3.81
3	4.63	3.86	4.28
4	4.48	4.96	4.69
5	4.44	5.58	4.96
6	4.78	6.78	5.69
Grand Total	18.33	19.08	18.69

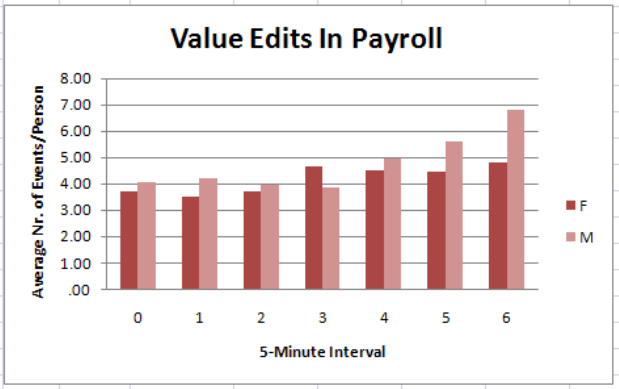


Figure 47: Value Edits in Payroll.

Workbook	GRADEBOOK
Hierarchy	Post Formula
CellType	Value

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	8.33	8.71	8.51
1	4.64	4.58	4.61
2	5.58	6.53	6.03
3	6.89	6.73	6.82
4	3.20	4.07	3.55
Grand Total	20.97	21.16	21.06

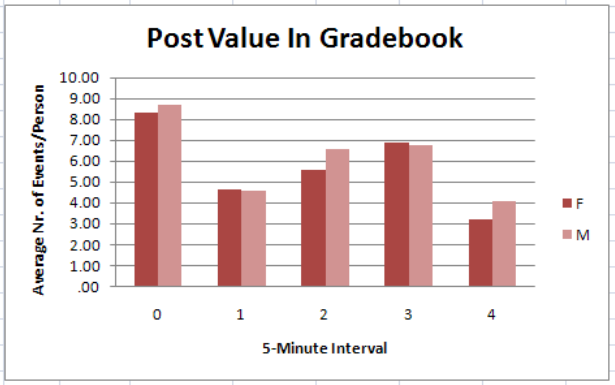


Figure 48: Post Value in Gradebook.

Workbook	PAYROLL
Hierarchy	Post Formula
CellType	Value

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	4.02	4.29	4.16
1	2.81	3.52	3.13
2	3.17	3.28	3.22
3	4.03	3.37	3.73
4	3.59	4.12	3.82
5	3.43	4.05	3.74
6	3.30	5.00	4.08
Grand Total	16.35	17.63	16.96

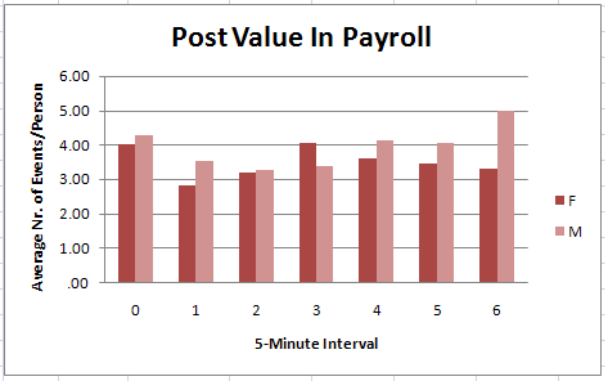


Figure 49: Post Value in Payroll.

Workbook	GRADEBOOK
Hierarchy	Hide Formula
CellType	Value

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	7.98	8.00	7.99
1	4.24	4.57	4.40
2	5.28	6.61	5.90
3	6.36	6.89	6.60
4	3.58	3.66	3.61
Grand Total	**20.15**	**20.31**	**20.23**

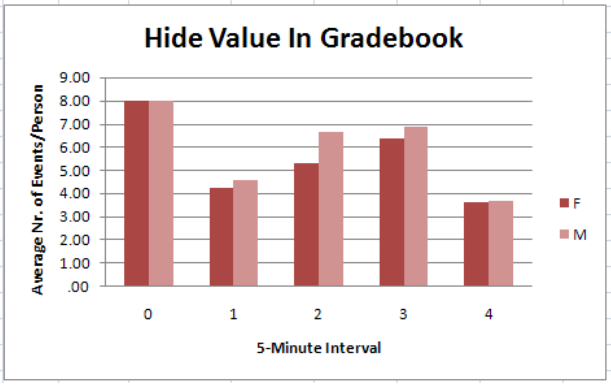


Figure 50: Hide Value in Gradebook.

Workbook	PAYROLL
Hierarchy	Hide Formula
CellType	Value

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	3.73	3.95	3.84
1	2.72	3.76	3.16
2	3.36	3.38	3.37
3	3.88	3.57	3.74
4	3.61	4.16	3.85
5	3.44	4.13	3.76
6	3.64	4.84	4.19
Grand Total	**15.79**	**16.97**	**16.35**

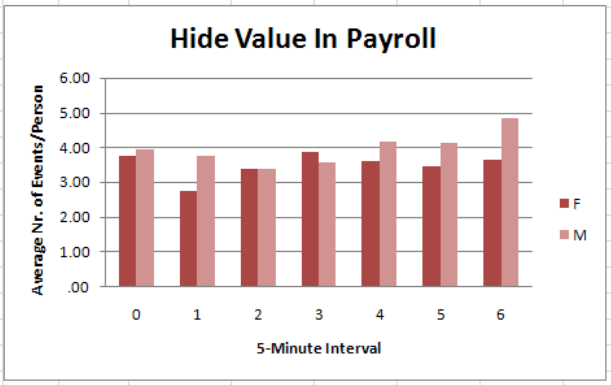


Figure 51: Hide Value in Payroll.

Results about Event Activity on Formula Cells

Workbook	GRADEBOOK		
Hierarchy	Edit		
CellType	Formula		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	2.34	2.32	2.33
1	3.29	3.73	3.49
2	3.51	3.14	3.34
3	3.31	3.39	3.34
4	2.17	2.16	2.16
Grand Total	11.50	11.10	11.31

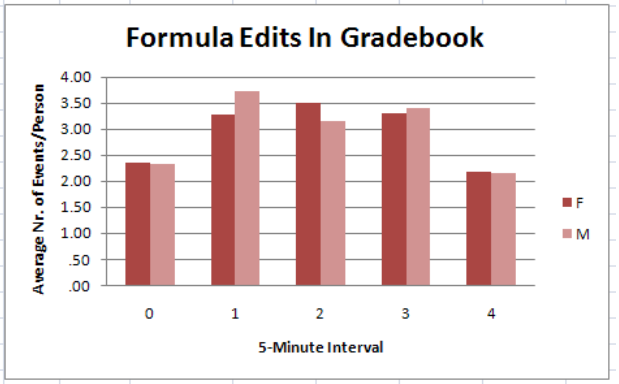


Figure 52: Formula Edits in Gradebook.

Workbook	PAYROLL		
Hierarchy	Edit		
CellType	Formula		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	1.90	1.77	1.85
1	2.20	1.98	2.11
2	2.57	1.95	2.26
3	1.84	2.22	2.03
4	2.68	2.19	2.44
5	2.51	1.83	2.20
6	2.63	2.23	2.45
Grand Total	10.02	8.29	9.20

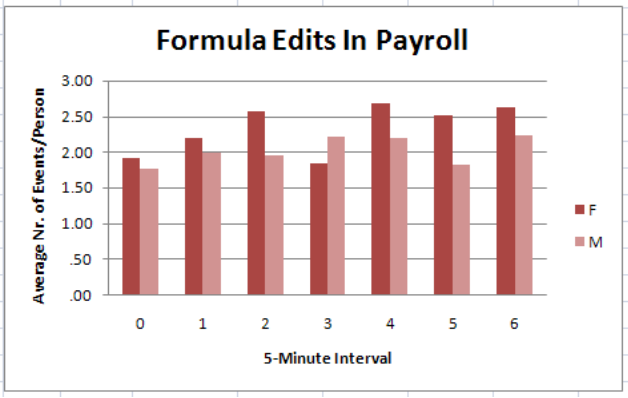


Figure 53: Formula Edits in Payroll.

Workbook	GRADEBOOK		
Hierarchy	Post Formula		
CellType	Formula		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	6.22	7.05	6.61
1	7.00	6.44	6.74
2	6.16	5.63	5.91
3	5.55	5.26	5.41
4	2.91	3.06	2.98
Grand Total	26.80	25.62	26.24

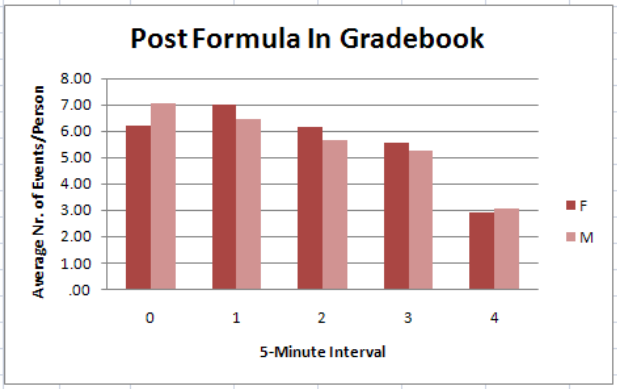


Figure 54: Post Formula in Gradebook.

Workbook	PAYROLL		
Hierarchy	Post Formula		
CellType	Formula		
EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	6.17	5.99	6.08
1	6.69	6.62	6.66
2	7.20	6.85	7.03
3	7.76	6.74	7.27
4	7.05	6.81	6.94
5	6.98	6.93	6.96
6	6.01	6.27	6.14
Grand Total	46.88	45.78	46.36

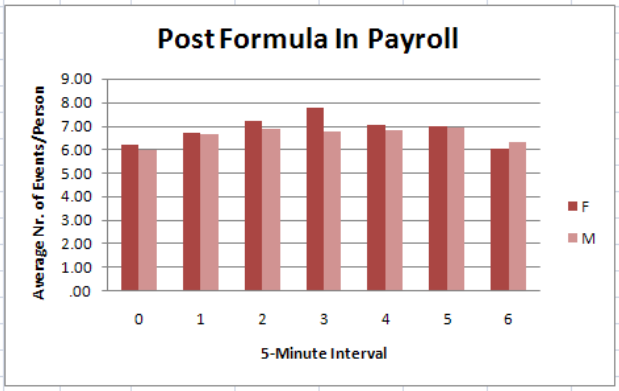


Figure 55: Post Formula in Payroll.

Workbook	GRADEBOOK
Hierarchy	Hide Formula
CellType	Formula

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	5.50	6.39	5.91
1	6.74	6.29	6.53
2	6.03	5.61	5.83
3	5.59	5.33	5.46
4	3.18	3.34	3.26
Grand Total	26.28	25.31	25.82

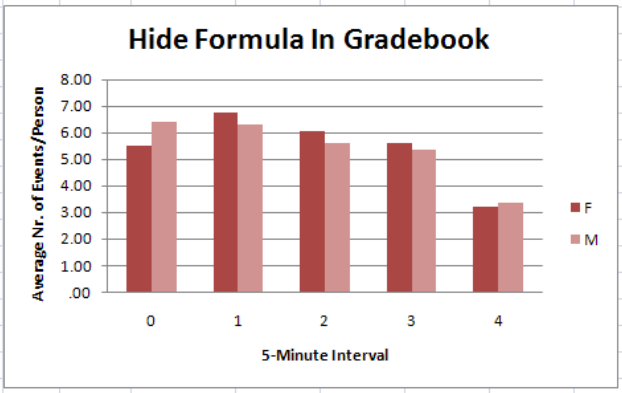


Figure 56: Hide Formula in Gradebook.

Workbook	PAYROLL
Hierarchy	Hide Formula
CellType	Formula

EventCount	Column Labels		
Row Labels	F	M	Grand Total
0	5.55	5.60	5.57
1	6.59	6.42	6.51
2	7.03	6.91	6.97
3	7.87	6.69	7.31
4	7.31	6.95	7.13
5	7.09	6.72	6.91
6	6.34	6.40	6.37
Grand Total	46.38	45.10	45.77

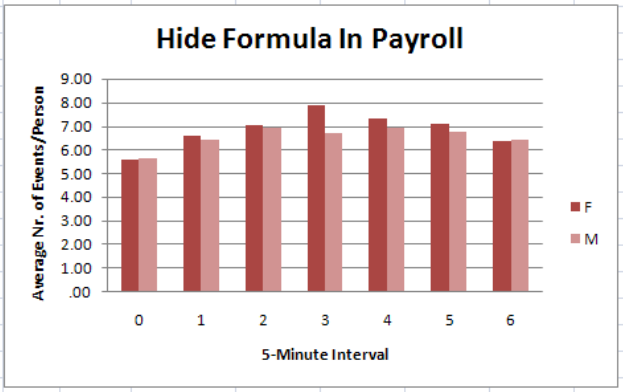


Figure 57: Hide Formula in Payroll.

Statistically Significant From Regression Analysis

The traditional way we have used to measure participants' success is by seeing how many bugs they fixed. This is the most important measure of success, since the ultimate goal of each user in our task is to find and fix the bugs that were planted into their spreadsheets. Thus, for males and females, what background data and feature usage data was predicted with the number of bugs fixed?

	Gradebook	Payroll
Males	Arrow Erased (p=0.023) "-" Edit Value (p=0.000) "+" Hide Value (p=0.003) "+" Post Value (p=0.001) "+"	Professional End-User Programming or Programming Experience (p=0.037) "+" Total Pre Self-Efficacy (p=0.001) "+" The First Task (p=0.025) X-Mark Placed (p=0.009) "-" Undo X-Mark (p=0.004) "-" Edit Value (p=0.034) "+" Hide Value (p=0.036) "+" Post Value (p=0.026) "+"
Females	Study (p=0.001) Major (p=0.010) GPA (p=0.041) "+" Spreadsheet Experience (p=0.039) "+" Total Pre Self-Efficacy (p=0.014) "+" Arrows Off (p=0.026) "-" Undo Checkmark (p=0.035) "-" Edit Value (p=0.000) "+" Hide Value (p=0.001) "+" Post Value (p=0.001) "+"	Year In School (p=0.015) "+" GPA (p=0.007) "+" Total Pre Self-Efficacy (p=0.017) "+" The First Task (p=0.020) Arrow Erased (p=0.005) "-" Undo X-Mark (p=0.009) "-"

Table 11: The variables that were had significant p-values, using regression analysis, in predicting the number of bugs fixed in Gradebook and Payroll by males and females. "+" means that the factor positively predicted the number of bugs fixed in Gradebook or Payroll (column headings) by Males or Females (row headings).

APPENDIX C. (MODELING) INITIAL MODELS OF STATIC DATA

Before beginning our full-scale model-building effort, we built models for the purposes of exploring the data and finding homogeneous groupings in the static data alone (background, self-efficacy scores, and success). Input variables and whether they were set as "Key", "Input", or "PredictOnly", can be seen in Figure 58 below. These were the same for all four models. The one thing that did differ among them was the parameter setting of the number of clusters. For CA, it was the default setting of "10". For CA2, we changed that value to 0 (see Figure 59). A value of 0 for the cluster count parameter divides the data up into its natural number of groupings. This natural number of clusters was 4. We therefore also created CA3 (with Cluster_Count set to 3) and CA 4 (with Cluster_Count set to 5), to see if either increasing or decreasing the natural number of clusters would lead to a better statistical fit for the testing
set.

Structure	CA_DiscTrainBackgroundPerc	CA2_DiscTrainBackgroundPerc	CA3_DiscTrainBackground	CA4_DiscTrainBackground
	Microsoft_Clustering	Microsoft_Clustering	Microsoft_Clustering	Microsoft_Clustering
CS Exp Rating	Input	Input	Input	Input
Gender	Input	Input	Input	Input
GPA	Input	Input	Input	Input
Major Type	Input	Input	Input	Input
SS Exp Rating	Input	Input	Input	Input
Std Comprehension	PredictOnly	PredictOnly	PredictOnly	PredictOnly
Study	Input	Input	Input	Input
Subject ID	Key	Key	Key	Key
Total Fixed Perc	PredictOnly	PredictOnly	PredictOnly	PredictOnly
Total Found Perc	PredictOnly	PredictOnly	PredictOnly	PredictOnly
Total Pre SE	Input	Input	Input	Input
Year In School	Input	Input	Input	Input

Figure 58: The four competing clustering models built using only static data (background and self-efficacy scores).

Parameters:

Parameter	Value	Default	Range
CLUSTER_COUNT	0	10	[0,...)
CLUSTER_SEED		0	[0,...)
CLUSTERING_METHOD		1	1,2,3,4
MAXIMUM_INPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_STATES		100	0,[2,65535]
MINIMUM_SUPPORT		1	(0,...)
MODELLING_CARDINALITY		10	[1,50]
SAMPLE_SIZE		50000	0,[100,...)
STOPPING_TOLERANCE		10	(0,...)

Figure 59: The parameter settings for the CA2 cluster analysis model. The CLUSTER_COUNT parameter was the only one that differed in these first four competing models.

Unlike in the models built in Study 2, where "successful" meant above the median in bug fixing, the participants were split into five equally-sized buckets for these models. The top performing model for predicting success at fixing 80%-100% of bugs, finding 90%-100% of bugs, and scoring 75%-100% on the comprehension test was CA2. Recall that CA2 was the one with the Cluster_Count parameter set to "0" for the natural number of groupings.

The reason why we decided to divide the participants into only two groups based on success in Study 2 (rather than the five groups like in this attempt) was that the datasets were too small to dependably build models predicting success in bug fixing at this level of detail. The training set highest-success group had 25 participants (12 females and 13 males, out of a total of 139) and there were only 8 such participants (3 females and 5 males) in the testing set (out of a total of 52). This is why the results from this chapter were only used as hypotheses to inform our design of possible competing models in Study 2.

CA2 divided the population into four homogeneous groupings (or clusters). There were high performers in each of these clusters. The attributes for each of those clusters are in Figure 60.

Figure 60: The distributions of the values of the static characteristics (row headings) for each one of the clusters in CA2 (column headings).

Since including variables that do not add anything to the model's efficiency only weaken it, we decided to further trim the number of input variables used. From previous Gender HCI studies, we already know that gender and self-efficacy are important background factors. We therefore created a cluster analysis model that only took these two input variables into account when predicting bugs fixed, bugs found, and comprehension scores (CA5). Using only "Gender" and "Total Pre Self-Efficacy", the model was able to predict 75% of the target group (80%-100% of bugs fixed). Adding "GPA" to the mix increases the accuracy of a new model, CA6, to CA2's 88% prediction. See Figure 61 for the distribution of these values among the different clusters. Furthermore, adding any other one input variable to "Gender", "Self-Efficacy", and "GPA" reduces the accuracy.

Hypothesis: For optimal prediction of high success at bug fixing, the only static input variables should be "Gender", "Self-Efficacy", and "GPA". Adding others weakens the model.

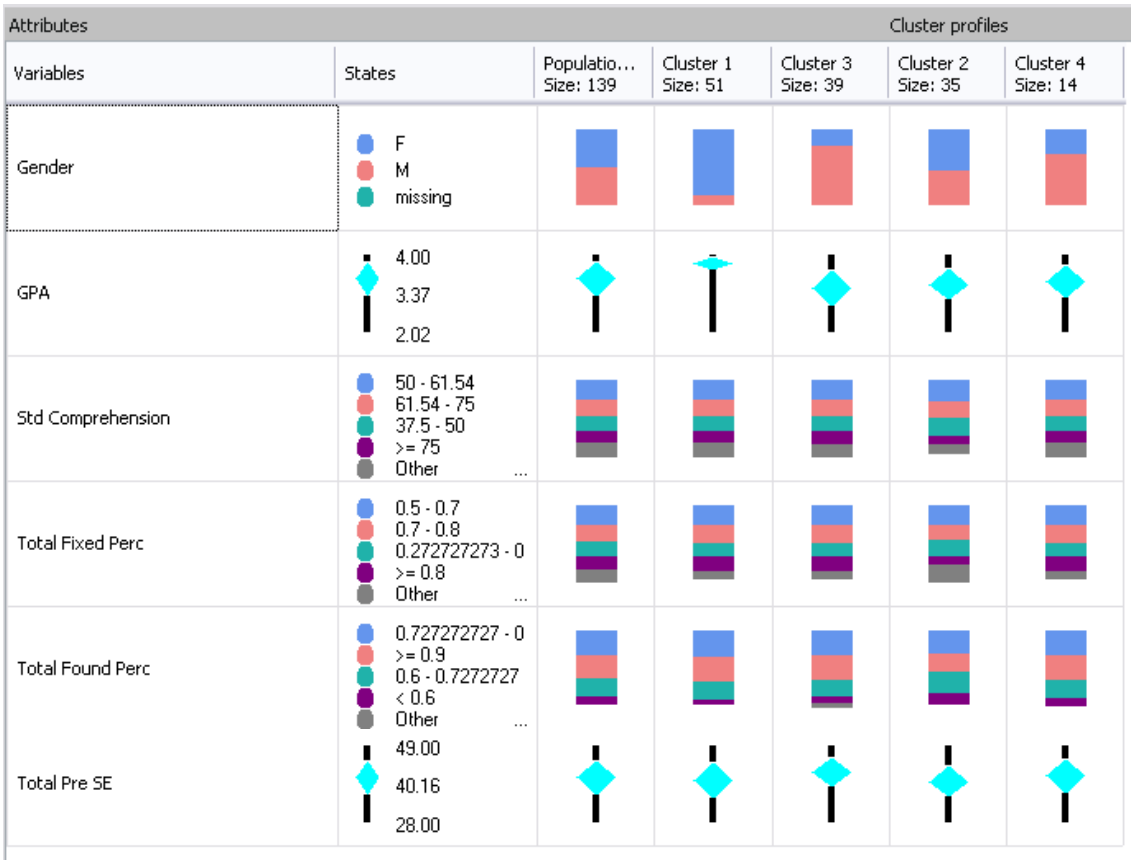


Figure 61: Cluster characteristics of four groupings in the population, for the top performing and most efficient model, CA6.

Recall that bugs fixed was only used as a measure after the models were built – not in building the models. One interesting observation is that the two mostly female clusters are very different in terms of bugs fixed (cluster 2 is above average, while cluster 1 is below average). The two male clusters, on the other hand, were similar in terms of success. Thus, females with similar background characteristics do similarly well at fixing bugs, while, just because males have similar patterns in background characteristics, it does not mean that they will fix a similar number of bugs.

Hypothesis: Background data is important in predicting female success at bug fixing. For males, other factors (other background data, behavioral data, situational data, or maybe something else altogether) play a more important role.

Since the sample population was too small to reliably verify the validity of these hypotheses, we do not consider them to be findings of this study. Instead, we used them to build a competing model for predicting success at fixing bugs in Study 2.

APPENDIX D. (MODELING) PARAMETER SETTING FOR THE MODELS

The default values provided by SSAS 2005 are conservative. We therefore only changed one of the parameters: for a cluster analysis model, we changed the number of clusters to "0", which provides the natural number of clusters that the data splits up into. My complete parameter settings are in the figures that follow.

Parameter	Value	Default	Range
MAXIMUM_ITEMSET_COUNT		200000	[1,...)
MAXIMUM_ITEMSET_SIZE		3	[0,500]
MAXIMUM_SUPPORT		1.0	(0.0,...)
MINIMUM_IMPORTANCE		-999999999	(...,...)
MINIMUM_ITEMSET_SIZE		1	[1,500]
MINIMUM_PROBABILITY		0.4	[0.0,1.0]
MINIMUM_SUPPORT		0.03	[0.0,...)

Figure 62: Association Rules parameters.

Parameter	Value	Default	Range
CLUSTER_COUNT	0	10	[0,...)
CLUSTER_SEED		0	[0,...)
CLUSTERING_METHOD		1	1,2,3,4
MAXIMUM_INPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_STATES		100	0,[2,65535]
MINIMUM_SUPPORT		1	(0,...)
MODELLING_CARDINALITY		10	[1,50]
SAMPLE_SIZE		50000	0,[100,...)
STOPPING_TOLERANCE		10	(0,...)

Figure 63: Clustering parameters.

Parameter	Value	Default	Range
HIDDEN_NODE_RATIO		4.0	[0,...)
HOLDOUT_PERCENTAGE		30	(0,100)
HOLDOUT_SEED		0	(...,...)
MAXIMUM_INPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_OUTPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_STATES		100	0,[2,65535]
SAMPLE_SIZE		10000	[0,...)

Figure 64: Neural Network parameters.

Parameter	Value	Default	Range
COMPLEXITY_PENALTY			(0.0,1.0)
FORCE_REGRESSOR			
MAXIMUM_INPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_OUTPUT_ATTRIBUTES		255	[0,65535]
MINIMUM_SUPPORT		10.0	(0.0,...)
SCORE_METHOD		4	1,3,4
SPLIT_METHOD		3	[1,3]

Figure 65: Decision Tree parameters.

Parameter	Value	Default	Range
MAXIMUM_INPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_OUTPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_STATES		100	0,[2,65535]
MINIMUM_DEPENDENCY_PROBABILITY		0.5	(0,1)

Figure 66: Naïve Bayes parameters.

Parameter	Value	Default	Range
HOLDOUT_PERCENTAGE		30	(0,100)
HOLDOUT_SEED		0	(...,...)
MAXIMUM_INPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_OUTPUT_ATTRIBUTES		255	[0,65535]
MAXIMUM_STATES		100	0,[2,65535]
SAMPLE_SIZE		10000	[0,...)

Figure 67: Logistic Regression parameters.